МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський національний університет залізничного транспорту імені академіка В. Лазаряна

Кафедра Комп'ютерні інформаційні технології

«ДО ЗАХИСТУ»

Завідувач кафедри

_____ /В. І. Шинкаренко/

«17» грудня 2020 р.

ДИПЛОМНА РОБОТА

на здобуття освітнього ступеня «магістр»

Галузь знань **12 Інформаційні технології**

Спеціальність **121 Інженерія програмного забезпечення**

Тема **Конструктивно - продукційне моделювання молнієвої активності**

Theme **Constructive-synthesizing modeling of lightning activity.**

Керівник дипломної роботи проф. _____ В. І. Шинкаренко

Нормоконтролер доц. _____ О. С. Куроп'ятник

Студентка групи П31921 _____ І. М. Нікітіна

Student Nikitina Iryna

Дніпро – 2020

Дніпровський національний університет залізничного транспорту
імені академіка В. Лазаряна

Факультет Комп'ютерні технології і системи
Кафедра Комп'ютерні інформаційні технології
Спеціальність Інженерія програмного забезпечення

«ЗАТВЕРДЖУЮ»
Завідувач кафедри
_____ проф. Шинкаренко В.І.
(підпис)
«10» жовтня 2020 р.

ЗАВДАННЯ

до дипломної роботи на здобуття ОС Магістр
(освітньо-кваліфікаційнийрівень)

студентки групи (ПЗ1921) 961-М Нікітіної Ірини Михайлівни
(номер групи) (ПІБ)

1 Тема дипломної роботи: Конструктивно-продукційне моделювання молнієвої активності

затверджена наказом по університету від «10» жовтня 2019 р. № 779ст.

2 Термін подання студентом закінченої роботи 16 грудня 2020 р.

3 Вихідні дані до дипломної роботи _____

4 Зміст пояснювальної записки (перелік питань до розробки) Призначення, постановка задачі та огляд аналогів, методика проведення дослідження, проектування і розробка інструментального програмного забезпечення, аналіз результатів, охорона праці та безпека в надзвичайних ситуаціях, висновки.

5 Перелік демонстраційного матеріалу Презентація на тему «Конструктивно-продукційне моделювання молнієвої активності», відео, що демонструє процес роботи програми «Автоматизована система конструктивно-продукційного моделювання молнієвої активності» у форматі mp4.

6. Консультанти (з назвами розділів):

Розділ	Консультант	Підпис, дата	
		завдання видав	завдання прийняв
Техніко-економічні розрахунки	доц. Гненний М.В.	30.10.20	02.12.20
Охорона праці та безпека в надзвичайних ситуаціях	ст. викл.. Музикін М.І.	15.10.20	02.11.20

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва розділів дипломної роботи	Термін виконання розділів роботи	Примітка
1	Вступ	04.11.2019 – 08.11.2019	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	11.11.2019 – 06.12.2019	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення при вирішенні проблем дослідження	09.12.2019 – 20.12.2019	30%
4	Постановка задачі, технічне завдання	23.12.2019 – 27.12.2019	
5	Техніко-економічні показники	13.01.2020 – 24.01.2020	
6	Розробка інструментальних засобів дослідження	03.02.2020 – 03.07.2020	60%
7	Проведення досліджень	08.07.2020 – 16.08.2020	
8	Оформлення пояснювальної записки	19.08.2020 – 04.12.2020	
9	Розробка демонстраційних матеріалів	08.12.2020 – 16.12.2020	100%

Дата видачі завдання «10» жовтня 2020 р.

Керівник дипломної роботи _____ Шинкаренко В.І.
(підпис) (ПІБ)

Завдання прийняв до виконання _____ Нікітіна І.М.
(підпис) (ПІБ)

РЕФЕРАТ

Об’єктом дослідження є комп'ютерне моделювання процесів розповсюдження грозового фронту та розрядів блискавок хмара-земля у атмосфері.

Предметом дослідження є методи виокремлення блискавок з відео з супутників NASA з подальшим математичним моделюванням розповсюдження грозових фронтів та розрядів блискавок у атмосфері.

Мета роботи полягає у визначенні засобів для виокремлення блискавок з відео з супутників NASA та у пошуці конструкторів, які б описували закономірності вертикального поширення електричних розрядів у атмосфері.

Методи дослідження: порівняння кольору пікселя на основі різноманітних моделей кольору, моделювання за допомогою конструкторів.

Результати та їх новизна: вперше використані моделі кольору Lab та LCH для пошуку блискавок на кадрах відео, побудовані моделі їх вертикального поширення з використанням L-систем.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, бібліографічного списку та 6 додатків:

– у вступі описується сутність розробки, її актуальність. Складається з 2 сторінок;
– у першому розділі висвітлено аналіз сучасного стану дослідження проблеми за науковими літературними джерелами також проаналізовано сучасний стану програмно-апаратного забезпечення. Складається з 15 сторінок;
– у другому розділі надано обґрунтування експериментального методу дослідження. Складається з 28 сторінок;
– у третьому розділі представлене проектування і розробка інструментального забезпечення для дослідження. Складається з 28 сторінок;
– у четвертому розділі описані виконані дослідження. Складається з 11 сторінок;
– у п’ятому розділі розкриті питання охорона та безпеки праці. Складається з 10 сторінок;
– додатки містять технічне завдання, робочий проект, дві статті, тези двох доповідей та інформаційну частину НДР.

Таблиць – 5, рисунків – 51, бібліографія – 65 джерел.

Ключові розпізнавання образів, кольорові, схема програми, L-системи, конструктивно-продукційне моделювання.

ЗМІСТ

ВСТУП

Актуальність роботи. На сьогодні блискавка продовжує складати суттєву загрозу для людей та обладнання. Прогнозування місць вірогідного влучення блискавок дозволило б завчасно провести профілактичні роботи на обладнані по захисту від грозового імпульсу та бути готовими прийняти заходів по ліквідації ймовірних руйнівних наслідків від ударів блискавок шляхом:

- передбачення резервних джерел електропостачання у випадках знаходження електричних підстанцій та електростанцій в зонах ризику ураження;
- посилення протипожежних заходів на об'єктах з підвищеною вогненебезпекою (газосховища, нафтосховища, ліси, електричні підстанції).

Пошук зони найбільш ймовірного ураження виконується на основі математичних моделей розповсюдження грозового фронту та поширення блискавки від хмари до землі. Для побудування моделі розповсюдження фронту необхідно проаналізувати характер молнієвої активності, використовуючи дані з погодних супутників, що дозволить розрахувати координати виникнення розряду. Подальше моделювання блискавки у вертикальному напрямку дозволить розрахувати координати її влучення.

Тема роботи: «Конструктивно-продукційне моделювання молнієвої активності».

Об’єктом дослідження є комп'ютерне моделювання процесів розповсюдження грозового фронту та розрядів блискавок хмара-земля у атмосфері.

Предметом дослідження є методи виокремлення блискавок з відео з супутників NASA з подальшим математичним моделюванням розповсюдження грозових фронтів та розрядів блискавок у атмосфері.

Мета роботи полягає у визначенні засобів для виокремлення блискавок з відео з супутників NASA та у пошуці конструкторів, які б описували закономірності вертикального поширення електричних розрядів у атмосфері.

Поставлена мета зумовила необхідність вирішення наступного комплексу взаємопов’язаних **завдань**:

- виявити особливості представлення блискавок на кадрах з відео метрологічних супутників;

- реалізувати інструментальне програмне середовище, здатне за виявленими особливостями визначати блискавки на кадрах відео з подальшим розрахунком їх радіусу та координат центру;
- дослідити здатність розроблених раніше моделей об’єктивно описувати характер поширення блискавок у грозовому фронті;
- реалізувати інструментальне програмне середовище, здатне за заданим конструктором моделювати поширення електричного заряду у атмосфері;
- знайти конструктори, які б описували реальні блискавки на основі візуального порівняння з фотографіями гроз.

Методи дослідження. Для визначення особливостей блискавок були застосовані методи обробки фотографій (видалення каналу кольору, виділення контурів методом різниці Гаусса) та порівняння кольору пікселя на основі різноманітних моделей кольору. Побудова поширення електричного заряду у атмосфері відбувалась на основі механізму перезапису ребер L-системи.

Наукова новизна. Вперше були використані моделі кольору Lab та LCH для пошуку блискавок на кадрах відео та побудовані моделі їх вертикального поширення з використанням L-систем. На відміну від інших досліджень, було зроблено акцент на можливість автоматичного розпізнавання блискавок в умовах сильної динамічної хмарності, реалізованого на сучасних мовах програмування.

Практичне значення. Результати даної роботи покликані зробити внесок у розробку систем прогнозування та виявлення місць ймовірного уражання блискавками. Результати роботи використані при розробці НДР «Конструктивно-продукційне моделювання фракталів», № державної реєстрації 0118U004215.

Апробація результатів дослідження. Основні положення магістерської роботи доповідалися та були схвалені на конференціях (19.09.2019 та 26.09.2020 на 14-й та 15-й «International Conference on Computer Sciences and Information Technologies (CSIT)») та семінарах кафедри КІТ (28.11.2019, 13.02.2020, 16.09.2020).

Публікації за темою роботи. За результатами роботи опубліковано чотири наукові праці, у тому числі: дві наукові статті [1], [2] (індексуються у базі даних SCOPUS) та дві доповіді на наукових конференціях [3], [4] (індексуються у базі даних SCOPUS).

1 ПРИЗНАЧЕННЯ, ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД АНАЛОГІВ

1.1 Опис проблеми. Актуальність дослідження

Через суттєву загрозу для людства молнієвій активності присвячується багато наукових праць. Так, у науковому дослідженні [5] узагальнюється і описується діяльність гроз і блискавок при розглядані їх як явищ несприятливої погоди на певних територіях і як елементів глобальної електричної мережі. Також, активно проводиться пошук та аналіз залежностей особливостей та характеру гроз від географічного їх положення та атмосферних умов; вже достатньо добре вивчені комплексні взаємозв'язки між блискавкою та дощем у конвективних бурях [6]. Багато досліджень присвячено дослідженню впливу параметрів сонячної мінливості та інших метеорологічних параметрів на грози та конвективні дощі в конкретних регіонах [7], пошуку зон активності у заданих географічних районах [8] та їх впливу на виникнення пожеж [9]. Для вивчення блискавичної активності використовуються різноманітні методи, так, наприклад глобальна грозова активність вивчалась за шкалою часу ENSO (Південне коливання Ель-Ніньо) на основі запису резонансів Шумана Землі в Надьченці (NCK), Угорщина, а також спостереження за допомогою оптичного детектору перехідних процесів (OTD) та датчика зображення блискавок (LIS) супутників [10]. Сфера досліджень в даних областях значно розширилась завдяки спостереженням блискавки за метеосупутниками, які наразі є одним із інструментів спостерігання за грозовим фронтом як у загальноземному, так і територіальному масштабах [11]. Дані, отримані від супутників дозволяють отримати знання, щодо поведінки блискавки у грозовому фронті, які необхідні для її моделювання. Однак, проблематиці моделювання блискавки у грозовому фронті, на відміну від робіт, що вивчають її характеристики та досліджують взаємозв'язки з метеорологічними умовам, присвячена значно менша кількість робіт, що зумовлено насамперед складністю: зазвичай такі роботи обмежуються виділенням компактних зон (скупчень) утворення блискавки [12].

Стосовно моделювання поширення блискавки від хмари до землі: хоча складність такого моделювання зумовлена стохастичністю блискавки, даній роботі

присвячена достатня кількість робіт, які використовують різноманітні методи: центральний метод скінченної різниці [13], ґраткову фрактальну модель для комп'ютерного моделювання DBM (модель діелектричного пробою) [14]. Проводяться вдосконалення раніше створених методів з метою врахування будівель, рельєфу та інших факторів, що впливають на характер поширення блискавок, під час моделювання [15]. Однак, всі описані методи розглядають лише моделювання блискавки у площині, хоча електричний розряд є тримірним явищем. Також для моделювання, як правило, використовуються матриці, які генеруються випадковим чином, що, по-перше, значно ускладнює перевірку адекватності моделей, а, по-друге, потребують значної кількості обчислень, роблячи перехід до тривимірного моделювання неможливим, через значне зростання складності алгоритму. Використання L-систем для моделювання розряду від хмари до землі значно знизить складність обчислень, а також просто дозволить виконати перехід до тривимірних моделей. У цей час моделювання активності блискавки на основі супутникового моніторингу у комбінації з моделюванням розрядів типу хмара-земля допоможе підвищити точність прогнозування штормової погоди, що, у свою чергу, допоможе уникнути людських та матеріальних втрат.

1.2 Огляд існуючих досліджень і публікацій щодо математичного моделювання розповсюдження грозових фронтів та розрядів блискавок у атмосфері

Національна лабораторія дослідження екстремальних штормів (NSSL) визначає погодні супутники, як один з інструментів спостерігання за грозами [11], оскільки за допомогою них можна побачити більшість районів Землі. Серія геостаціонарних оперативних супутників спостереження за навколишнім середовищем-R (GOES-R) – це нове покоління геостаціонарних метеорологічних супутників. Перший супутник GOES-R, GOES-16, який покриває східну половину США, був запущений в 2016 році; GOES-17 покриває західну половину і був запущений в 2018 році. Кожен з цих супутників оснащений геостаціонарним картографом блискавок (GLM), який виявляє світлове випромінювання як від хмар до землі, так і від блискавок типу хмара-хмара. Супутники фотографують Землю

через рівні проміжки часу з космосу, а метеорологи, у свою чергу, у реальному часі спостерігають за цими знімками з метою виявлення хмар, які швидко збільшуються, тобто ймовірним місцем виникнення грози. Далі відстежується напрямок руху таких хмар, щоб визначити місце виникнення грози. Також відбувається збір даних щодо середовища перед бурею та після неї. Крім того, GLM забезпечує моніторинг блискавок на великій території, який вчені з NSSL та інших організацій можуть використовувати для вирішення дослідницьких питань, які вони не могли вирішити раніше. Це допомагає синоптикам швидко виявляти грози, щоб вони могли видавати точні і своєчасні попередження про сильні грози та торнадо.

Шляхом польових експериментів вивчається, як грози генерують блискавки та як поширюється сама блискавка у атмосфері. Розглядаються дані з карт блискавок (рис. 1.1), щоб щоб дізнатися, як зміни в їх поведінці можуть бути пов'язані з різними типами штормів.

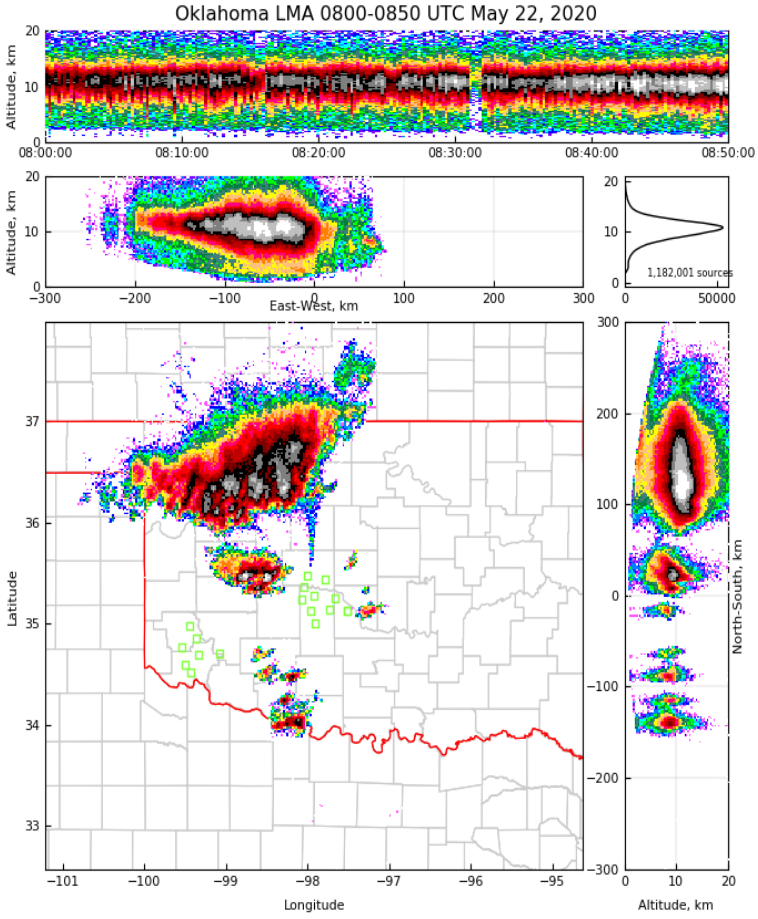

Рисунок 1.1 – Дані за певний момент часу з карти блискавок [16]

Наприклад, деякі грози мають «молнієві дірки», де збільшується кількість висхідних потоків і зменшуються кількість опадів, безпосередньо перед тим, як шторм стане більш сильним. Також було виявлено, що поблизу висхідних потоків грози спалахів блискавок, як правило, менше, що може бути корисно для синоптиків при швидкому виявленні частин штормових комплексів, які необхідно відстежувати.

На основі отриманих даних дослідники NSSL створили комп’ютерну модель, що імітує грозу в тривимірному просторі (рис. 1.2), яка використовується для вивчення поведінки грози в залежності від різних погодних умов. На рис. 1.2 показано край хмари (сірий колір), вертикальну завихреність (синій), блискавки (білі та жовті простори), вектори вітру та поверхню, яка імітує відбивну здатність радару. Дослідники використовують цю модель для вивчення мікрофізичної структури шторму та етапів його розвитку та взаємозв’язку між мікрофізикою та електричними явищами шторму, а також для моделювання різних фаз стихійних лих. Ця робота призвела до поліпшення прогнозування та попереджень, покращила розуміння середовищ, сприятливих для утворення гроз, дозволила визначити явища, які сигналізують про грозу, що розвивається.

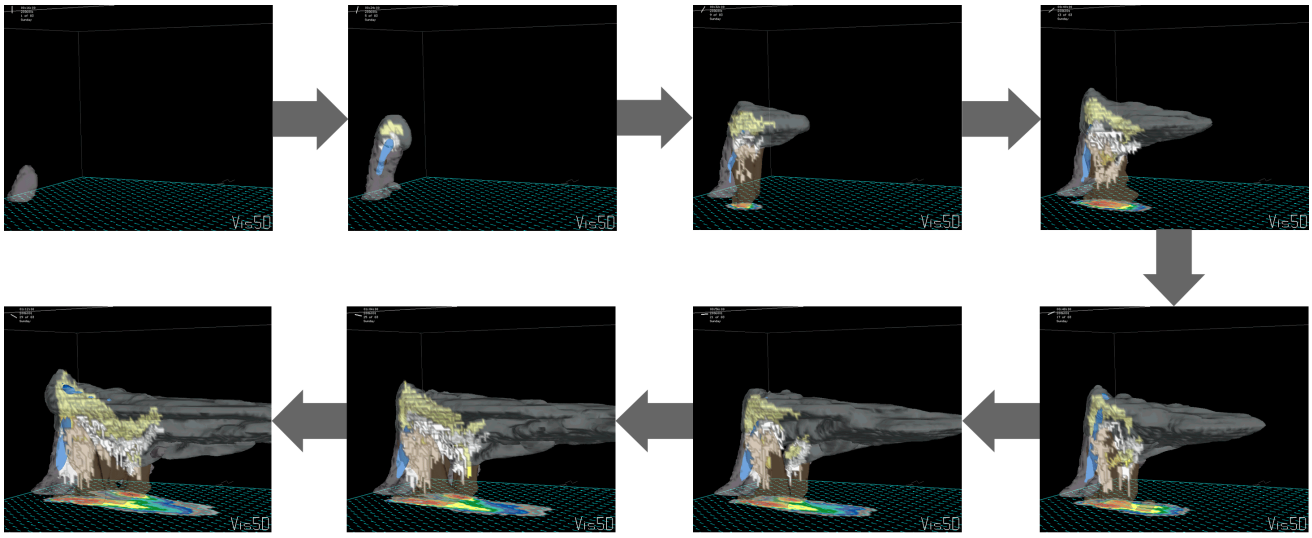

Рисунок 1.2 – Комп’ютерна модель, що імітує поведінку грози у 3-D режимі

До того ж NSSL працює над розробкою ансамблів моделей короткочасних прогнозів (від 0 до 60 хвилин) суворих погодних явищ. Ансамблі – це групи комп'ютерних моделей прогнозу, які здатні приймати дані від допплерівського радару або метеорологічного супутника, щоб забезпечити формування більш точних

прогнозів гроз та пов'язаних з ними важких погодних умов (рис. 1.3). Вчені NSSL вивчають, як може молнієва активність допомогти передбачити поведінку самих гроз. Збільшення частоти спалахів може означати посилення шторму в найближчому майбутньому. Для вивчення поведінки шторму можна використовувати дані супутника в поєднанні з даними радара, щоб дати комп'ютерної моделі більше інформації про початкові атмосфері умови і, власне, грозу, що розвивається. NSSL в даний час співпрацює з іншими групами для тестування різних способів включення інформації GLM в моделі прогнозів, один з яких – метод асиміляції даних.

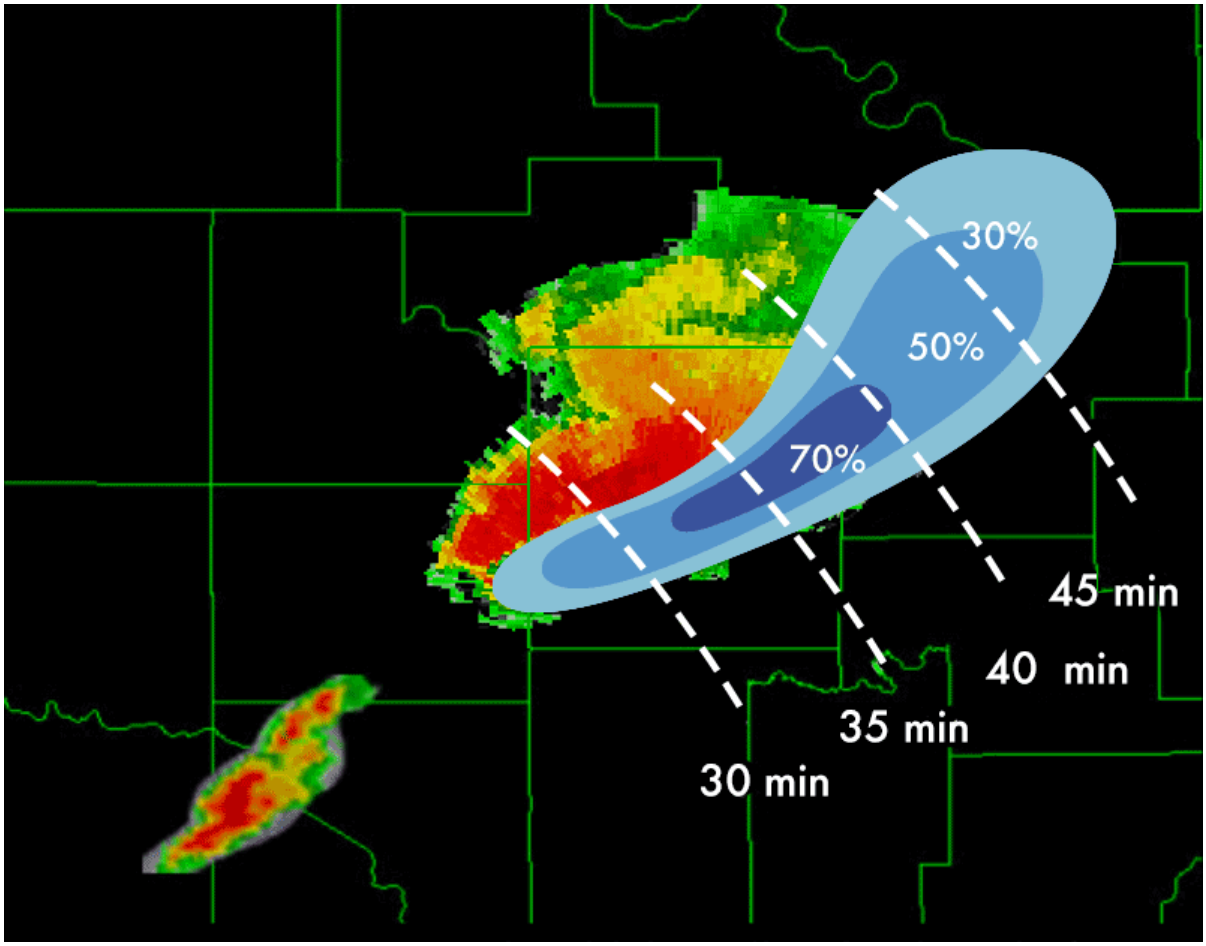

Рисунок 1.3 – Концептуальна модель Warn-on-Forecast, яка передбачає шлях потенційно штормової хмари протягом наступної години

Що стосується безпосередньо самої блискавки як електричного розряду, що поширюється в атмосфері, то дослідники NSSL вивчають ці явища вже майже півстоліття, досліджуючи її структуру та поведінку, розробляючи при цьому методи використання отриманих даних для покращення прогнозів штормової погоди та попереджень [17].

Вчені NSSL/CIMMS змоделювали реалістичні розряди блискавок між хмарами і землею (рис. 1.4), використовуючи тривимірну модель хмар, яка генерує складні опади, такі як граупель (м'який град), який, як відомо, впливає на утворення блискавок. Після чого, отримані моделі спалахів блискавок були порівняні із дійсно спостережуваними. Виявлені подібності та відмінності між спалахами були ретельно проаналізовані, деякі результати аналізу були додані до моделі дослідження і прогнозування погоди (WRF), з метою більш якісного прогнозування і оцінювання виникнення блискавок.

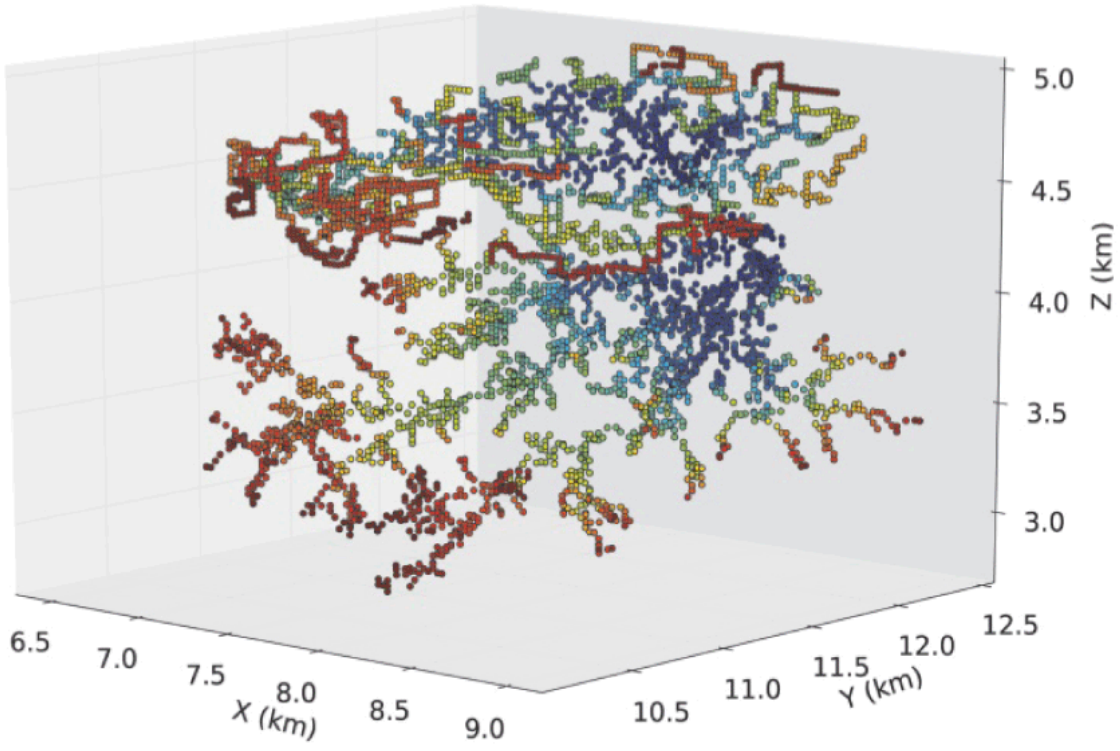

Рисунок 1.4 – 3D-модель, яка імітує спалах блискавки під час модельованих штормів

Хоча лабораторію NSSL проведено багато досліджень, всі матеріали складають комерційну таємницю, їх немає у вільному доступі. Однак, описана раніше інформація допоможе розробити алгоритм виконання даної роботи та розділити її на певні етапи, допоможе більш чітко зрозуміти необхідність виконання певних робіт у дослідженні. Так, наприклад, для перевірки адекватностей моделей,

необхідно мати реальні дані. Оскільки поширення грозового фронту є динамічним, то для перевірки моделей необхідні відео його поширення. NASA представляє [18] серії кадрів, відзнятих метеорологічними супутниками, що містять корисну інформацію для моделювання та валідації моделей. Через те, що серії містять значну кількість кадрів, необхідно розробити алгоритми розпізнавання блискавок на кадрах з супутників.

В даний час для автоматизації процесу аналізу серій знімків розробляються нові алгоритми [19]. Значна частина досліджень, пов'язаних з обробкою даних із погодних супутників, присвячена виявленню [20][21] або класифікації [22][23] типів хмар. Ці дослідження засновані на аналізі форми, кольору та руху хмари, використовуючи різні методи розпізнавання візерунків. Так, у роботі [23] було реалізовано декілька методів відділення хмар від фону. Перший метод – метод однопорогової сегментації, який заснований на тому факті, що висока хмарність виглядає яскравіше, ніж інші частини інфрачервоних супутникових зображень (результат у другому стовпці рис. 1.5). Хоча цей метод дозволяє відокремити високі хмари від фону, він ігнорує межі деяких хмар. Через те, що Земля має періодичні зміни температури, постійно відбувається коливання температури на поверхні землі, на які впливають багато факторів, включаючи рельєф, висоту і географічну широту, єдиний поріг не може застосований і адаптований до всіх цих випадків.

Через це виникає дослідження нового підходу. Загальна ідея нової схеми сегментації описується наступним чином: щоб знати про просторово-часові зміни супутникових зображень, пікселі зображення поділяються на квадрати, а потім моделюються образи кожної одиниці за допомогою GMM (модель суміші Гауса). Після цього відбувається ідентифікація існування компонента, який, відповідає варіаціям високої яскравості хмар і неба. Приклади результатів сегментації високохмарних областей наведені в третій колонці на рис. 1.5. В кінці цього кроку високі хмари відокремлюються з детальною локальною інформацією, тоді як низькі хмари та суша видаляються.

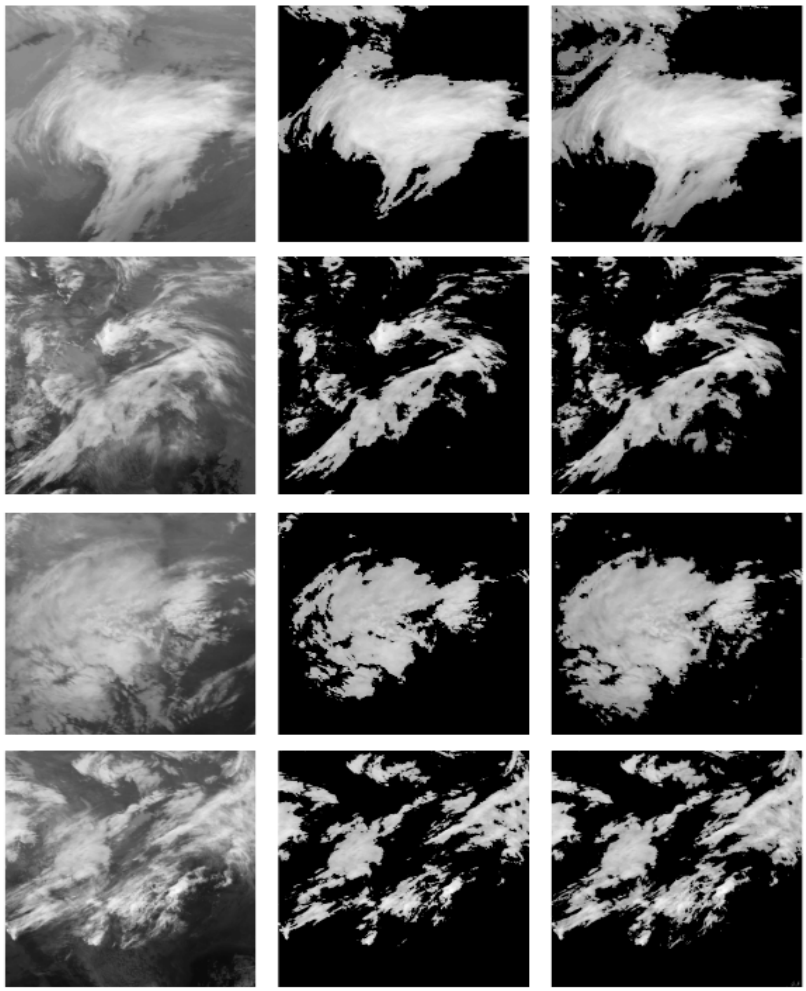

Рисунок 1.5 – Обрізані супутникові зображення. (a) Вихідні дані.
(b) Виділені високі хмари методом однопрохідної сегментації.
(c) Сегментовані високі хмари методом GMM.

Для виявлення спалахів блискавки в динамічно мінливій помутнінні можна застосувати класифікацію змінених пікселів [24], але не завжди пікселі, які є блискавичною частиною, змінюють свій колір (яскраві спалахи центральної частини, як правило, майже однакові під час кілька кадрів). Інший підхід використовує особливі властивості його представлення поточним супутником [25][26], де фільтр виявляє блискавку як горизонтальну яскраву стрічку (рис. 1.6). Освітлення, виявлене блискавкою, утворює горизонтальні стрічки шириною рівно 16 ліній, що відповідає кількості ліній, зібраних за одне сканування датчика. На зразках зображень

зображено три блискавки. Лінія A-B показує положення сканера уздовж рядка, представленого на рисунку 1.7. Алгоритм виявив дві з трьох блискавок.

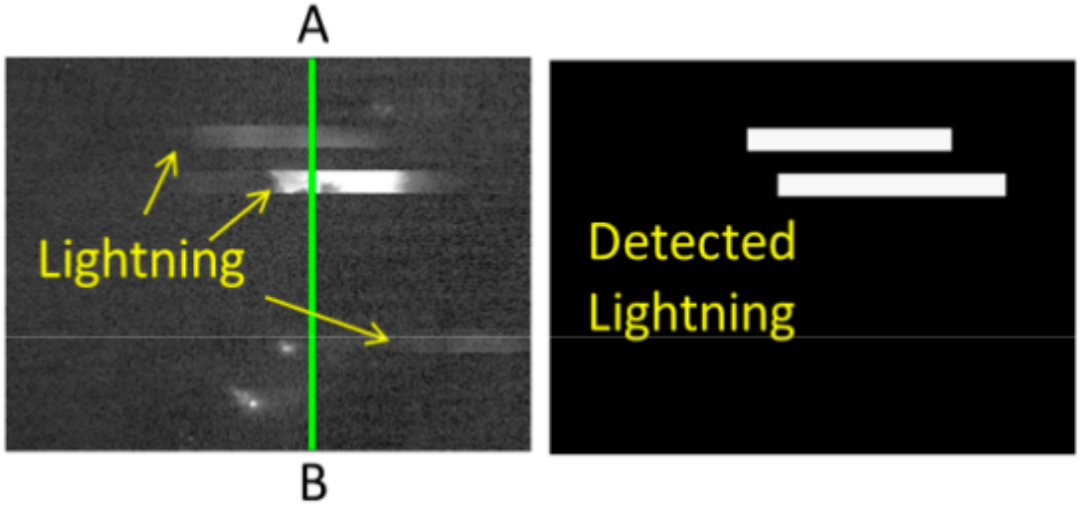

Рисунок 1.6 – Результат роботи алгоритму. Зліва – вхідне зображення. Праворуч – виявлені блискавки.

Ці стрічки мають ширину рівно 16 рядків, що виникає в результаті одночасного збору 16 рядків даних за одне сканування. Алгоритм виявлення освітлення заснований на виявленні просторово великих кроків яскравості вздовж меж між скануванням. Вхідними файлами є попередньо оброблені зображення DNB. Процедура наведена графічно на рисунку 1.7.

Алгоритм виявлення блискавки обчислює різницю сяйва між лініями сканування, що розділяють 16 сканувань. Сегменти сканування з різницею більше 0,1 Log10 DNB, які зберігаються протягом 24 або більше пікселів поспіль, позначаються як блискавка. Пікселі, позначені блискавкою, відкидаються і не записуються у вихідний файл csv (в даній роботі блискавка є стороннім джерелом освітлення і має бути видалена з кадрів).

Незважаючи на те, що цей метод не може бути застосований для виявлення блискавки в кадрах з високою роздільною здатністю, підхід із застосуванням пошуку особливостей зображення блискавки на кадрах з поточного супутника може бути застосований в алгоритмах їх розпізнавання. Даний підхід вважається автором роботи як найбільш перспективний.

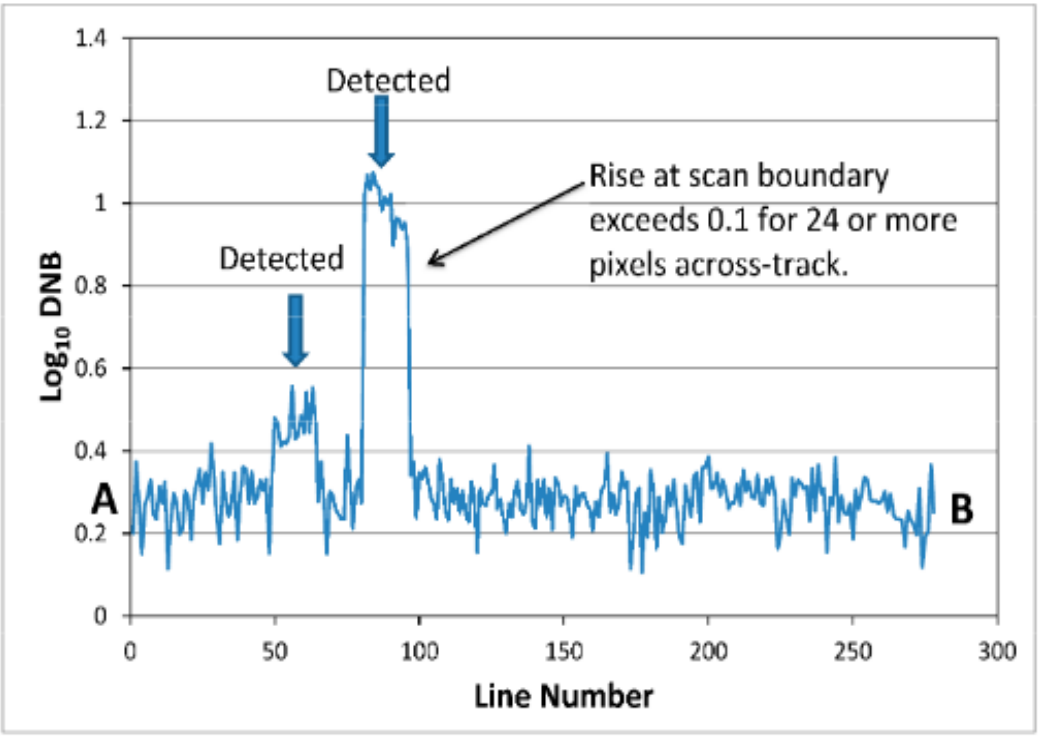

Рисунок 1.7 – Графічне зображення алгоритму пошуку блискавки на основі кольору пікселів вздовж рядка A – B на рисунку 1.6.

1.3 Огляд існуючих програмних інструментів для виокремлення та моделювання блискавок

Для автоматичної процесу виокремлення блискавок з відео з супутників NASA з подальшим математичним моделюванням розповсюдження грозових фронтів та розрядів блискавок у атмосфері наразі існують програмні засоби. У якості прикладу можна розглянути продукти лабораторії NSSL, де вчені працюють над двома інструментами, обидва з використанням машинного навчання, з метою передбачення блискавок в даному місці. Перший – це інструмент інформування ймовірнісної небезпеки блискавок (PHI). Цей інструмент включає дані щодо поточних штормів і минулих спостереженнях, щоб постійно прогнозувати ймовірність ударів блискавки між хмарами і землею протягом наступної години. Другий інструмент – Flash Extent Density [27].

1.3.1 Flash Extent Density

Даний інструмент дозволяє моделювати сильні шторми та спостерігати за тими, що існують наразі. Для того, щоб забезпечити адаптованість до конкретних погодних умов, додаток має інструменти: наприклад, для роботи з моделлю на рис. 1.8 кольорова крива була дещо змінена з метою забезпечення більш чіткого відображення низьких значень, оскільки через це падає ефективність роботи.

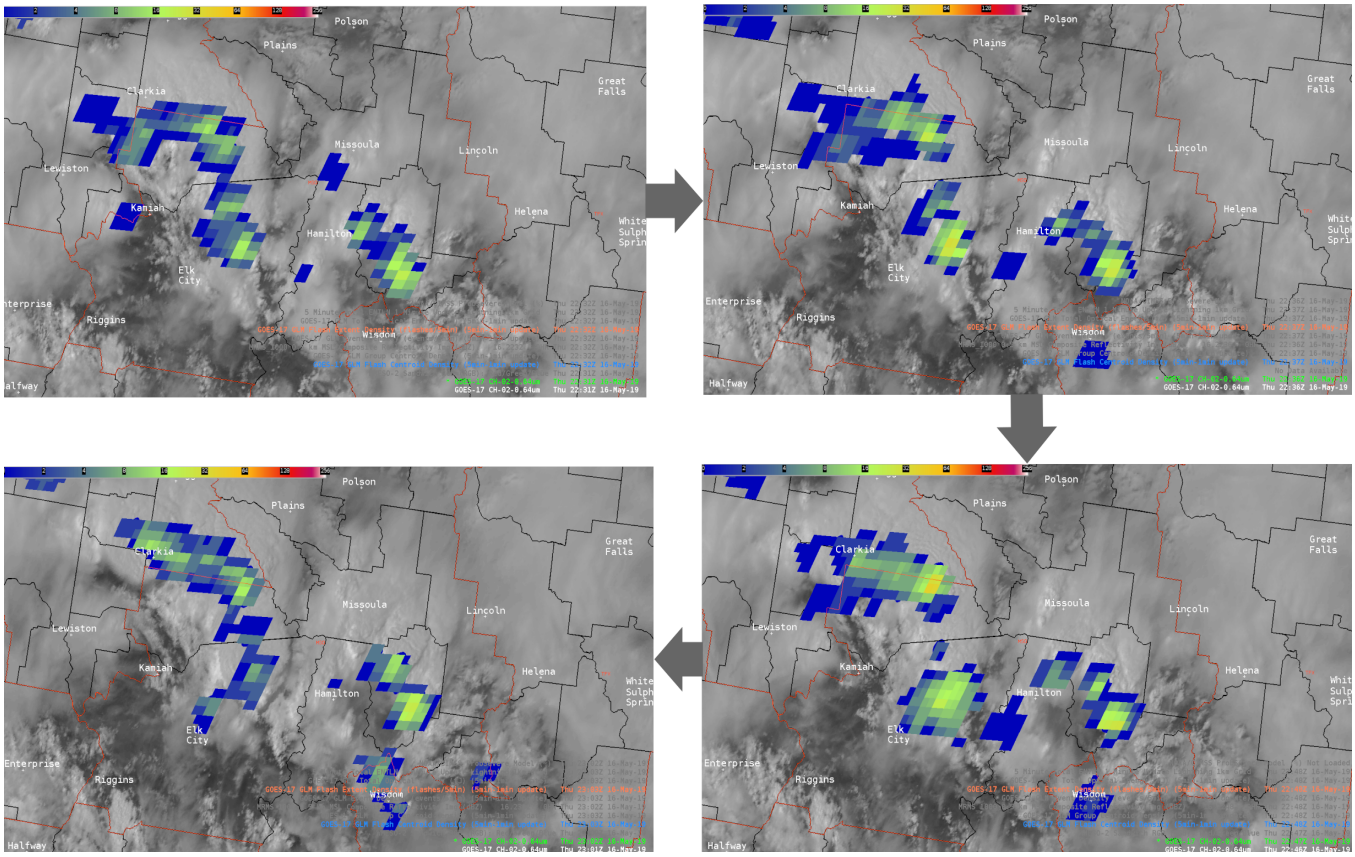

Рисунок 1.8 – Приклад роботи продукту Flash Extent Density

Однак, хоча це і покращує виявлення деяких сильних штормів, великі клітини займають багато місця на екрані під час перегляду фонових супутникових зображень.

1.3.2 Flash Centroid Density Product

Тому для більш точного виявлення блискавок був розроблений дочірній продукт Flash Centroid Density Product (рис. 1.9). При використані даного продукту легше помітити спалах блискавки під час шторму, а також порівняти силу з навколишніми штормами. Також даний продукт потребує менше місця на моніторі.

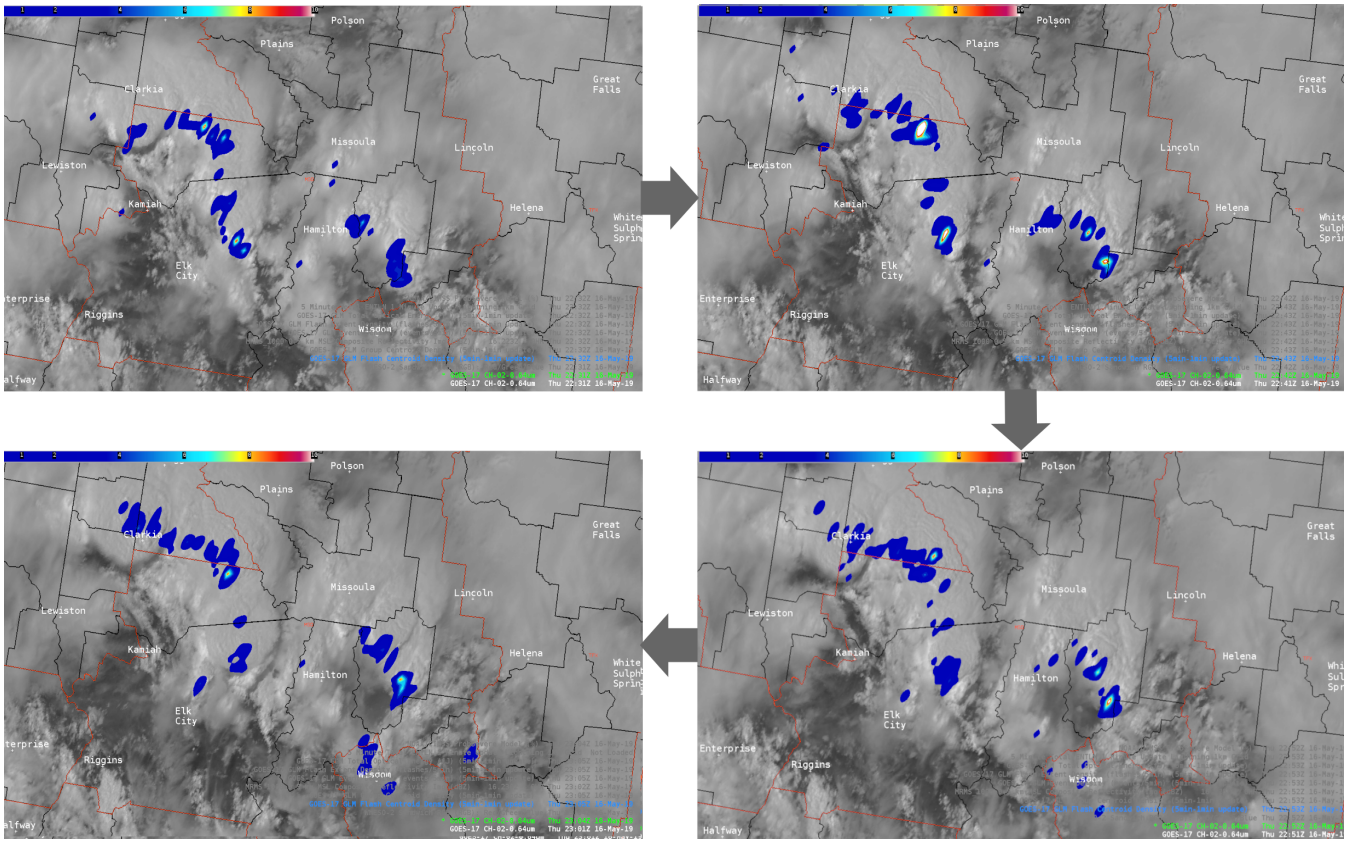

Рисунок 1.9 – Приклад роботи продукту Flash Centroid Density Product

Оскільки дані продукти є розробкою національної лабораторії дослідження екстремальних штормів, то, можливо припустити, що вони достатньо повністю реалізують поставлені перед ним задачі, однак, це неможливо перевірити через закритість даних додатків (в широкому доступі оприлюднені деякі особливості їх роботи), а самі продукти складають комерційну таємницю.

1.3.3 Modelling and Simulation of Lightning Discharge Patterns

Іншим покладом є програма «Modelling and Simulation of Lightning Discharge Patterns» [28], яка призначена для пошуку та обробки блискавок на знімках з космосу. Хоча дана програма чітко розпізнає блискавки (рис. 1.10), дозволяє користувачеві вводити і комбінувати різні параметри, її основним недоліком є те, що програма не має графічного інтерфейсу (рис. 1.11).

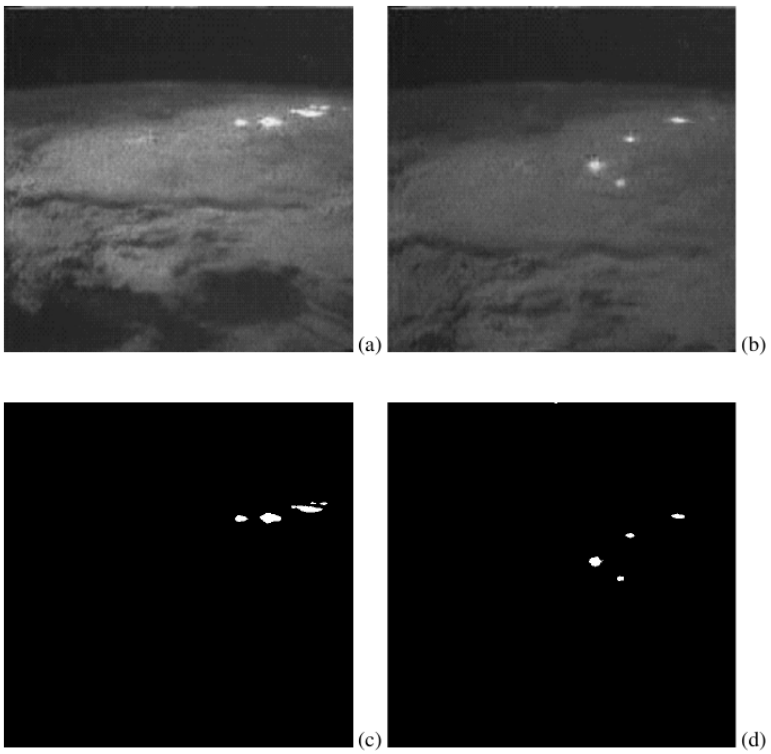

Рисунок 1.10 – Вхідні зображення з шатлу та результат їх обробки

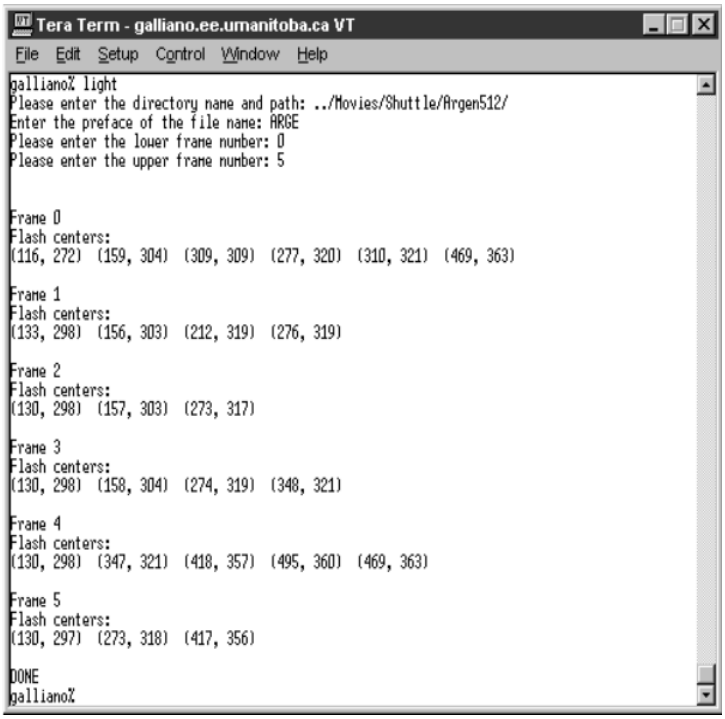

Рисунок 1.11 – Приклад роботи програми «Modelling and Simulation of Lightning Discharge Patterns»

1.3.4 L-SYSTEMS GENERATOR

Наразі існує багато додатків, які дозволяють генерувати конструкції, засновані на L-системах. Один таких: *L-SYSTEMS GENERATOR* [29], що є

інструментом 2D/3D моделювання для Windows, заснований на L-SYSTEMS та клітинних автоматах, написаний на мові C# (рис. 1.12). *L-SYSTEMS GENERATOR* включає набір експериментальних інструментів для редагування, перетворення та комбінування моделей L-SYSTEMS для дослідження генеративного мистецтва, фракталів, природних форм, модульних конструкцій, шаблонів, алгоритмічних архітектур та музики.

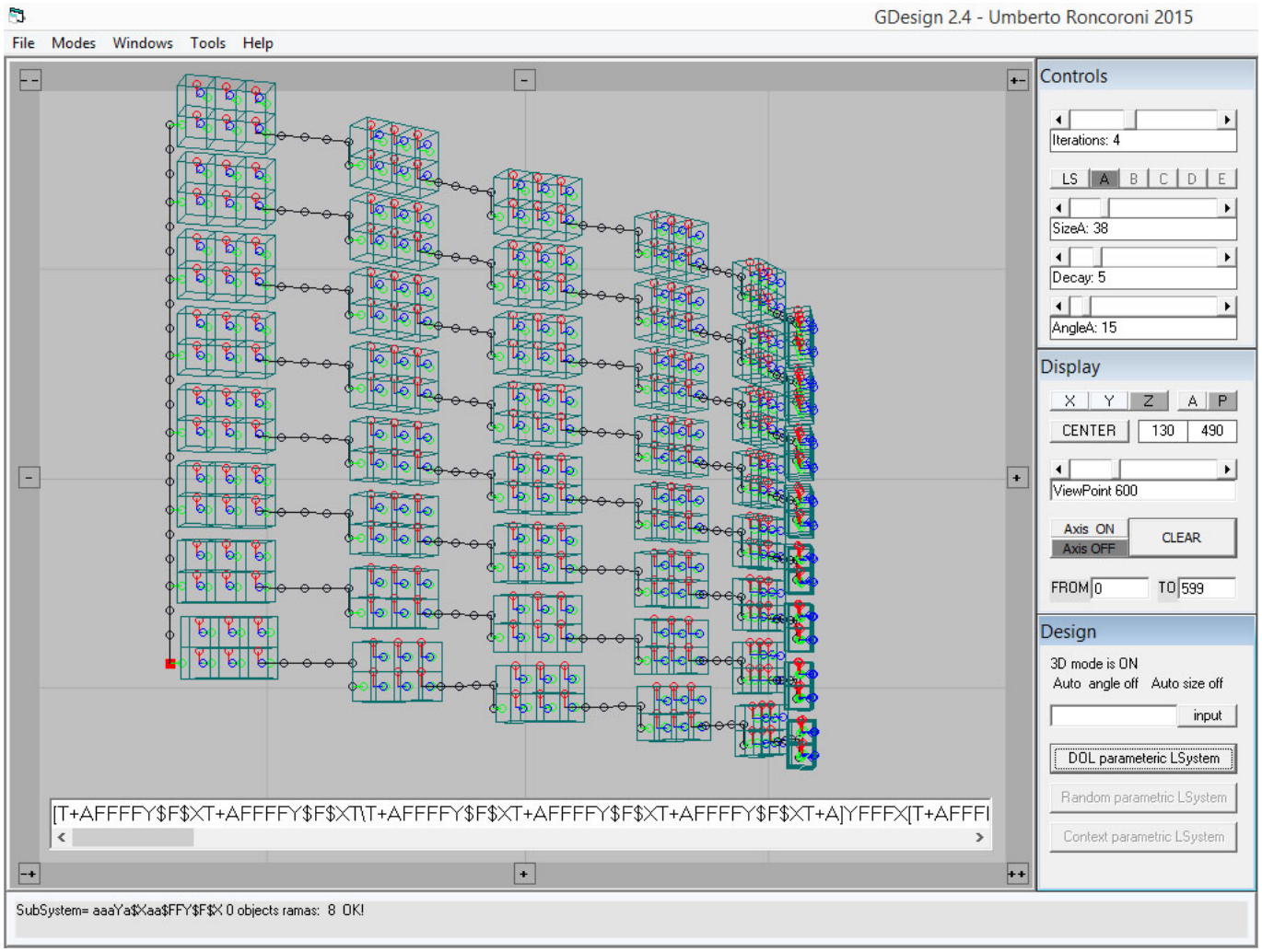

Рисунок 1.12 – Приклад роботи програми L-SYSTEMS GENERATOR

Даний продукт дозволяє створювати й стохастичні конструкції, що може бути використано під час моделювання розрядів блискавки типу хмара-земля. Можливе редагування граматик, аксіом, правил та за допомогою інтерактивних інструментів та простого у використанні інтерфейсу, що дозволяє змінювати параметри в режимі реального часу. Користувачеві доступні інструменти удосконалення функціональності L-Systems та використання спеціальних символів,

макросів, функцій, масок, автоматичної поведінки для створення та редагування складних 2D/3D об’єктів. До того ж отримані результати, можна експортувати у файли в Rhino, STL, POVRay та 3DMax. Недоліком даного продукту є його платформозалежність (працює лише для Windows).

Висновки до розділу 1

Аналіз проблеми показав відсутність повноцінних методів виокремлення спалахів блискавки на кадрах супутників з подальшим математичним моделюванням розповсюдження грозових фронтів та розрядів блискавок у атмосфері у вільному доступі. У рамках дослідження необхідно поєднати та адаптувати під поставлену задачу рішення, які існують наразі щодо виокремлення динамічних та статичних об'єктів з кадрів супутників.

Усі розглянуті існуючи програмні інструменти, що є у вільному доступі, потребують допрацювання з метою поєднання моделювання грозового фронту та вертикального розповсюдження.

2 МЕТОДИКА ВИОКРЕМЛЕННЯ ТА МОДЕЛЮВАННЯ БЛИСКАВОК

2.1 Методика виокремлення блискавок на кадрах з відео зі статичним фоном

2.1.1 Виділення об'єктів переднього плану з послідовності кадрів, отриманих зі статичної камери

Задачу виявлення блискавки на кадрах з відео метеорологічних супутників, що демонструють статичний фон можна привести до задачі «виділення об'єктів переднього плану з відеопослідовності, отриманої зі статичної камери», оскільки фон, на якому виникають блискавки є статичним, а на передньому плані немає об'єктів окрім блискавок. У таких системах сцена мало змінюється від кадру до кадру в відеопотоці, тому називається «фоном». Всі нові об'єкти, що потрапляють в зону видимості камери, відрізняються від фону і називаються «переднім планом». Формально задача виділення об'єктів переднього плану в відеопотоці ставиться як визначення для кожного пікселя кожного кадру, чи належить він фону, або об'єкту на передньому плані [30]. Тобто для отримання зображення з виокремленими блискавки із вхідних кадрів необхідно з'ясувати лише, які пікселі є фоном та зафарбувати їх у певний колір; таким чином плями на зображені, які залишились без змін – шукані блискавки.

Найпростішим методом для вирішення даної задачі є метод, заснований на аналізі кожного пікселя зображення. Пікселі аналізуються незалежно один від одного. Зазвичай спочатку будується колірна модель фону, а під час обробки оцінюється, наскільки поточний колір пікселя їй відповідає. В якості моделі фону може використовуватися, наприклад, нормальний розподіл або комбінація зі статично [31] або динамічно [32] заданого числа нормальних розподілів. Однак, дані методи потребують наявність кадру, який не містить об'єктів переднього плану. Інші алгоритми засновані на аналізі зміни фонової і шумової складової сцени у послідовності кадрів [33]. Спостерігаючи за змінами, що мають місце на зображенні в часі, з'являється можливість оцінити як сам фон, так і перешкоду, що спотворює його. Таким чином, з'являється реальна можливість відокремити корисний сигнал,

який створюється динамічним об'єктом, від шуму і відносно стабільного фонового рівня. Тобто можна визначити, що алгоритм визначає піксель фоном, якщо його характеристики відповідають пікселю в аналогічній позиції на попередньому кадрі.

Хоча подібні методи чутливі до шуму і не враховують зв'язність розмітки (серед більшості пар сусідніх пікселів обидва або належать фону, або переднього плану одночасно) [30], вони зазвичай мають досить високу швидкістю роботи та є досить простими у своїй реалізації, а описані проблеми можна вирішити шляхом повторної обробки зображення, яка передбачає видалення шумів та добудови видалених частин об'єктів переднього плану.

В якості відео, що буде оброблятися за даною методикою виступає відео [34] з супутників NASA, на якому статична хмарність та форма блискавок близька до кола або еліпса (рис. 2.1).

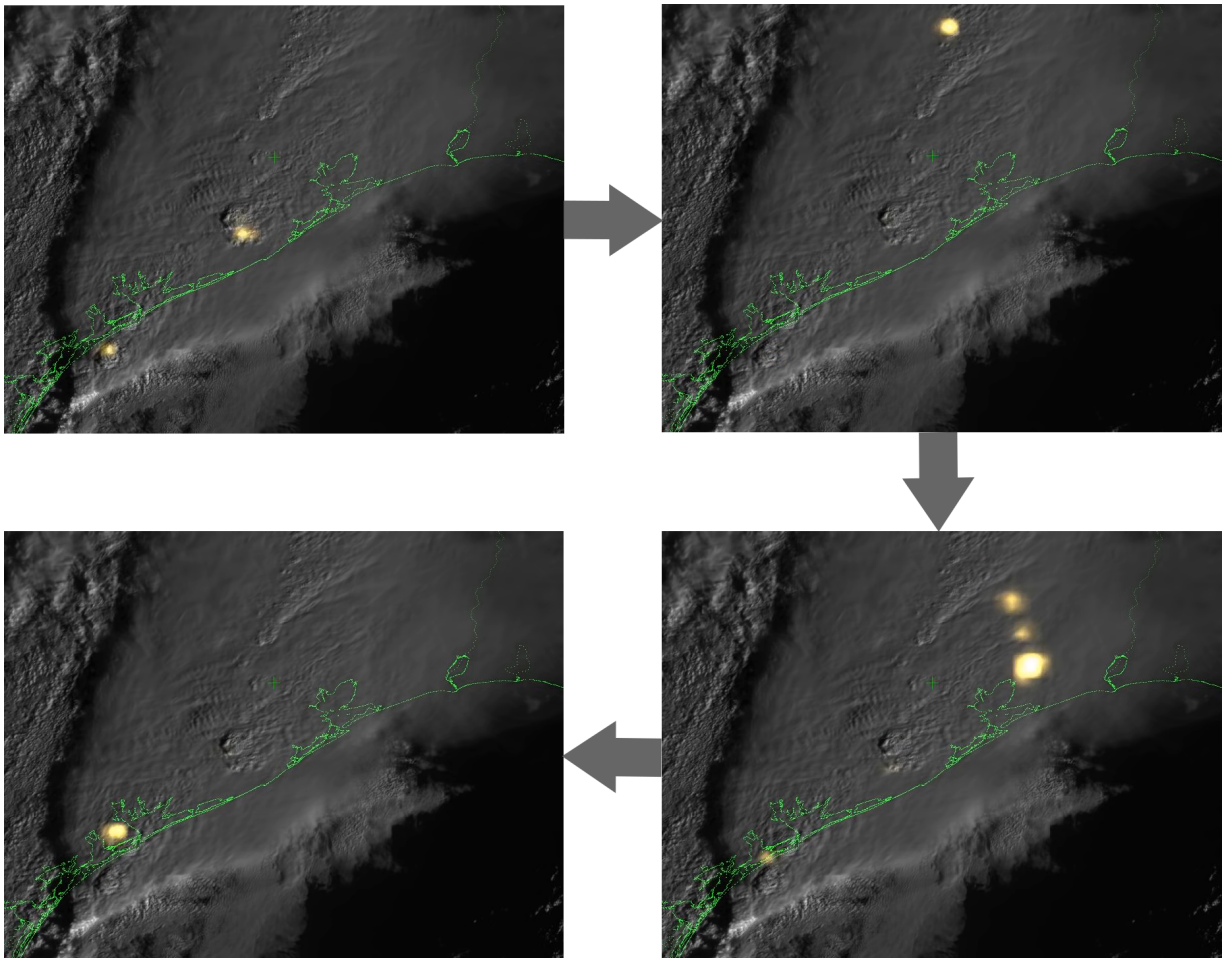

Рисунок 2.1 – Кадри відео, створеного супутниками NASA, що демонструють грозу з блискавками зі статичним фоном

2.1.2 Алгоритми покращення якості обробленого зображення

Алгоритми видалення шумів та добудови видалених частин об'єктів є простими у своїй реалізації, добре вивчені та описані, оскільки почались використовуватися дуже давно [35]. Наприклад, для видалення шумів достатньо застосувати наступну формулу для кожного пікселя при однократному скануванні зображення:

$$\alpha=\alpha(x_0\vee x_1\vee x_2\vee x_3\vee x_4\vee x_5\vee \bar{x}_6)\wedge(x_2\vee x_3\vee x_4\vee x_5\vee x_6\vee x_7\vee \bar{x}_0)\wedge \\ \wedge(x_4\vee x_5\vee x_6\vee x_7\vee x_0\vee x_1\vee \bar{x}_2)\wedge(x_6\vee x_7\vee x_0\vee x_1\vee x_2\vee x_3\vee \bar{x}_4), \quad (2.1)$$

де α – колір даного пікселя, а x_i має значення 1 (для пікселів об'єктів переднього плану) або 0 (для пікселів фону) та представляє собою один з 8-ми сусідніх пікселів даного при розташуванні, показаному на рис. 2.2.

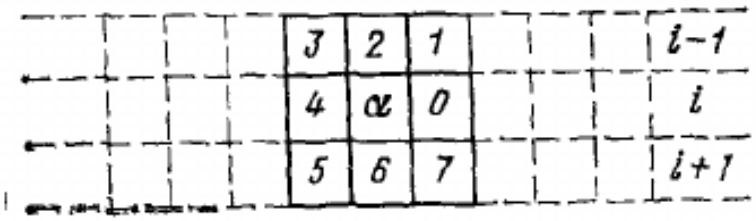

Рисунок 2.2 – Правила нумерації 8-ми сусідніх пікселів з даним

Визначається, що процедури видалення шумів можуть привести до втрати корисної інформації, тому процес заповнення пустот необхідно виконати перед ним. Як показали експерименти на ЕОМ, хороші результати показує наступний алгоритм: піксель вважається пікселем переднього плану, якщо більше половини при розгляді 8 сусідніх з ним елементів також належать пікселям переднього плану. Число циклів такої обробки залежить від ряду факторів та встановлюється емпірично.

2.1.3 Визначення кольорової різниці як інструмент виокремлення блискавок

Міжнародний комітет CIE [36] пропонує для визначення кольорової різниці декілька стандартів: CIE 1976 Lab* колірний простір та CIE 1994 LCh колірний простір, які визначають різницю у кольорі через змінні ΔE_{ab} та ΔE_{CH} відповідно, де літера «E» походить з німецького «Empfindung» – «Почуття».

За стандартом CIE 1976 Lab різниця обчислюється за формулою:

$$\Delta E_{ab} = \sqrt{(L_2 - L_1)^2 + (a_2 - a_1)^2 + (b_2 - b_1)^2}, \quad (2.2)$$

де використовуються координати 1-го кольору (L_1, a_1, b_1) та координати 2-го кольору (L_2, a_2, b_2).

За стандартом CIE 1994 L*C*h різниця обчислюється за формулою:

$$\Delta E_{CH} = \sqrt{\left(\frac{L_2 - L_1}{K_L}\right)^2 + \left(\frac{C_2 - C_1}{1 + K_1 C_1}\right)^2 + \left(\frac{H_2 - H_1}{1 + K_2 C_1}\right)^2}, \quad (2.3)$$

де використовуються координати 1-го кольору (L_1, C_1, H_1) та координати 2-го кольору (L_2, C_2, H_2) та ваговий коефіцієнт K, який визначається в залежності від сфери застосування. В даному випадку логічно використовувати коефіцієнти для промислової сфери: $K_L = 2; K_1 = 0{,}048; K_2 = 0{,}014$.

Основну проблему складає той факт, що більшість стандартних засобів для роботи із графічною інформацією використовують модель кольору RGB. В [37] для визначення різниці кольорів запропонована схема застосування моделей кольору як показано на рис. 2.3.

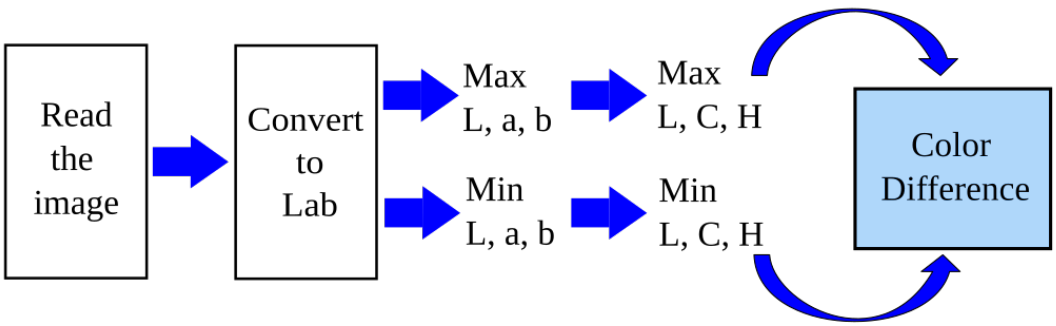

Рисунок 2.3 – Схема алгоритму для визначення різниці кольору

Тобто для обчислення різниці кольорів, спершу, необхідно конвертувати координати кольору у моделі RGB до координат Lab або LCH, в залежності від вибору формули (2.2) або (2.3) відповідно.

Для цього спершу необхідно конвертувати RGB значення у значення простору XYZ [38]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M \begin{bmatrix} r \\ g \\ b \end{bmatrix}, \tag{2.4}$$

де M – матриця перетворень, яка обчислюється в залежності від основних еталонних кольорів RGB, які визначають той чи інший RGB робочий простір, а $[r\,g\,b]'$ – вектор-стовпець, утворений шляхом лінеаризації початкових компонентів у моделі RGB, визначених в інтервалі $[0;1]$. При виборі в якості робочого простору стандарту sRGB матриця M має наступні коефіцієнти:

$$\begin{bmatrix} 0{,}4124 & 0{,}3576 & 0{,}18052 \\ 0{,}2126 & 0{,}7152 & 0{,}0722 \\ 0{,}0193 & 0{,}1192 & 0{,}9505 \end{bmatrix}, \tag{2.5}$$

а компонент v_i вектор-стовпця $[r,g,b]'$ при урахуванні того, що існує вектор-стовпець $V=[R\,G\,B]'$, де $V_i \in [0;1]$, обчислюється за формулою:

$$v_i = \begin{cases} \dfrac{V_i}{12{,}92} \cdot 100, & V_i \le 0.04045 \\ \left(\dfrac{V_i + 0.055}{1.055} \right)^{2{,}4} \cdot 100 \end{cases}. \tag{2.6}$$

Отримані значення компонент у моделі XYZ знаходяться в інтервалі [0; 100]. Для обчислення значень у моделі Lab використовується формула [39]:

$$\begin{aligned} L &= 116 f_y - 16 \\ a &= 500 (f_x - f_y), \\ b &= 200 (f_y - f_z) \end{aligned} \tag{2.7}$$

де f_x, f_y, f_z обчисляються за формулами:

$$f_x=\begin{cases}\sqrt[3]{x_r}, x_r>\epsilon \\ \dfrac{\kappa x_r+16}{116}\end{cases}, \tag{2.8}$$

$$f_y=\begin{cases}\sqrt[3]{y_r}, y_r>\epsilon \\ \dfrac{\kappa y_r+16}{116}\end{cases}, \tag{2.9}$$

$$f_z=\begin{cases}\sqrt[3]{z_r}, z_r>\epsilon \\ \dfrac{\kappa z_r+16}{116}\end{cases}. \tag{2.10}$$

де $x_r=\frac{X}{X_r}$, $y_r=\frac{Y}{Y_r}$, $z_r=\frac{Z}{Z_r}$ (X_r, Y_r, Z_r залежать від обраного балансу білого), а $\epsilon=0.008856; \kappa=903.3$, значення яких визначено дійсним CIE стандартом. Оскільки для конвертації RGB до XYZ простору був використаний стандарт sRGB, який використовує баланс білого D65, приймаємо $X_r=95{,}047; Y_r=100{,}0; Z_r=108{,}883$.

Значення координат кольору у моделі LCH, отримуються із розрахованих значень Lab [40]:

$$L_{LCH}=L_{Lab}$$

$$C=\sqrt{a^2+b^2}$$

$$H=\begin{cases}\arctan\left(\frac{b}{a}\right), \arctan\left(\frac{b}{a}\right)\geq 0, \\ \arctan\left(\frac{b}{a}\right)+360^\circ\end{cases}. \tag{2.11}$$

Однак, можна помітити, що описані вище формули потребують складних математичних обчислень, що призведе до суттєвого зниження часової ефективності роботи алгоритму з виділення блискавок, тому була запропонована проста формула обчислення різниці кольору на базі моделі RGB:

$$\Delta E_{RGB} = |R_2 - R_1| + |G_2 - G_1| + |B_2 - B_1|, \quad (2.12)$$

де використовуються координати 1-го кольору (R_1, G_1, B_1) та координати 2-го кольору (R_2, G_2, B_2).

2.1.4 Функція для виділення блискавок

Алгоритм вирішення цього завдання використовує попередній кадр при обробці поточного. Щоб виділити блискавки у відеокадрах, фільтрування зображень виконується наступним чином (кольорова модель – RGB):

$$y_{ij} = \begin{cases} x_{ij}, якщо\, d_{ij} > 0 \\ 0, інаше \end{cases}, \quad (2.13)$$

де x_{ij} – значення пікселя з координатами (i, j) на кадрі перед обробкою, y_{ij} – відповідне значення на сформованому відеозображенні, що містить лише контури блискавки, і

$$d_{ij} = tol - \Delta E, \quad (2.14)$$

де tol – допуск максимальної різниці, ΔE – визначає різницю кольорів пікселів з координатами в каналі відповідно до використаного методу (2.2), (2.3) або (2.12) на поточному та попередньому кадрі перед обробкою.

2.2 Методика виокремлення блискавок на кадрах з відео, що демонструють динамічну хмарність

Більшість відео, створених супутниками, що демонструють процес поширення грозового фронту, демонструють також і високу динамічну хмарність. Цей факт значно ускладнює виявлення блискавки. Для аналізу обрані відео [41][42] з супутників NASA, до яких під час зйомки були застосовані фільтри таким чином, що блискавка являє собою яскраву пляму з кольоровим контуром. Такі фільтри дозволяють з більшою точністю стежити за формою блискавки в горизонтальній площині. На рис. 2.4 представлені кадри з відео [41] через чотири кадри.

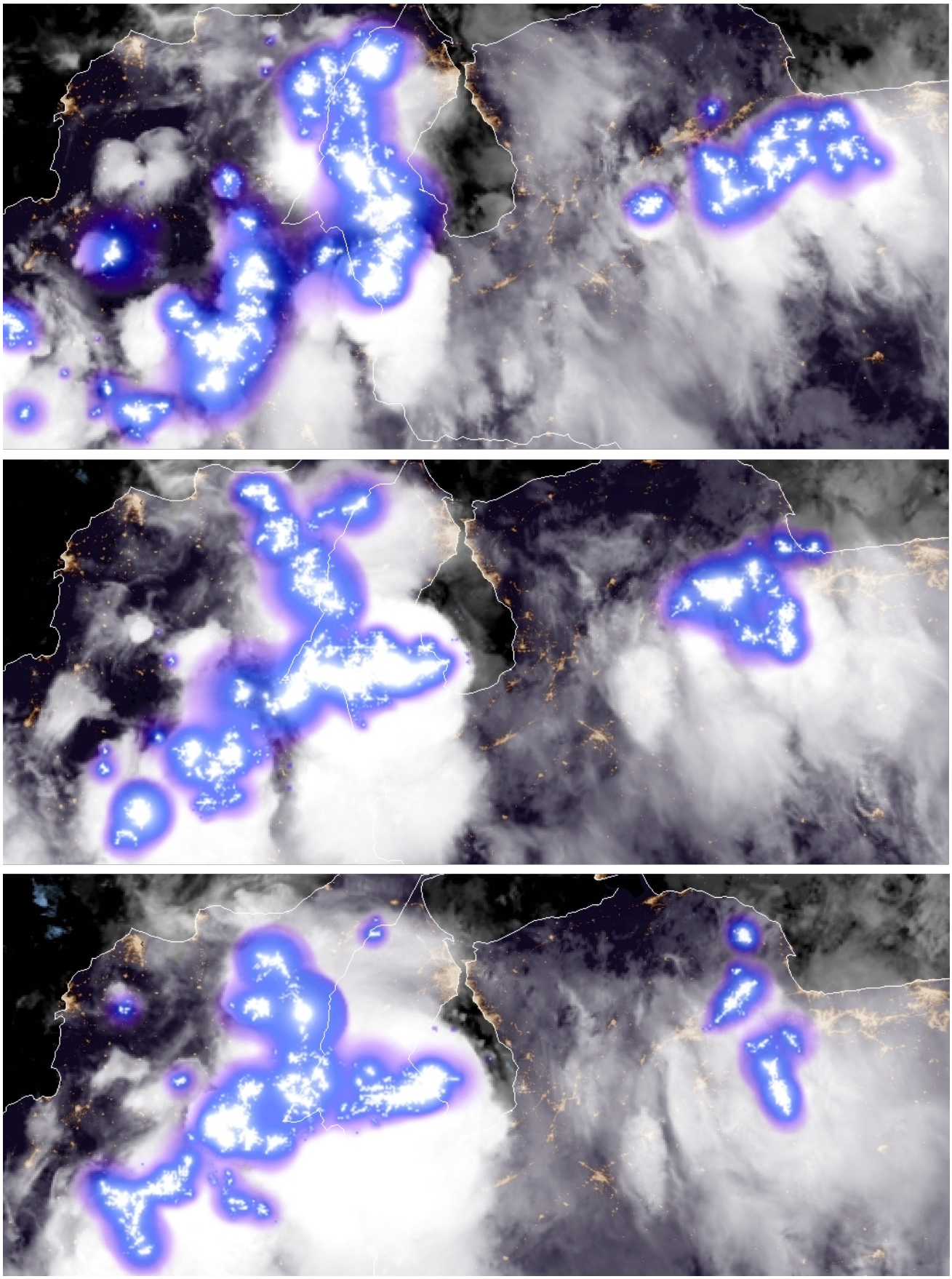

Рисунок 2.4 – Супутникові знімки грозового фронту з розрядами блискавок

Через наявність храм та інших рухомих об'єктів під час гроз розглядати блискавку лише як динамічне явище недостатньо, оскільки внаслідок такого алгоритму відокремлення деякі отримані плями є сторонніми об'єктами. З метою вирішення даної проблеми необхідно шукати інші методики виокремлення блискавок на кадрах з метеорологічних супутників.

2.2.1 Роздільна здатність моделей кольору

Кольорові характеристики об'єкта, який необхідно ідентифікувати, можуть бути розглянуті в задачі розпізнавання образів, при цьому колірна модель має значення. Колірна модель являє собою математичну модель опису представлення кольорів у вигляді кортежів чисел (зазвичай з трьох, рідше – чотирьох значень), які називають колірними компонентами або колірними координатами. Всі можливі значення кольорів, що задаються моделлю, визначають колірний простір [43]. Через відмінність колірних просторів різних моделей, а також врахування деякими умов перегляду, певні моделі можуть враховувати та показувати деякі особливості певних об'єктів, в той час як інші – ні. Це спонукає до пошуку оптимальних моделей для ідентифікації явищ та предметів, що досліджуються. Зараз активно проводяться дослідження пошуку моделі, яка найбільш підходить для вирішення певного класу задач в різних областях знань, наприклад: метод виділення проблемних ділянок шкіри базується на моделях RGB і XYZ [44], пошук особливостей ушкоджених регіонів у ретинопатії виконувався в HLS, LCH та Lab моделях [45], моделі Lab і HSV виявились найбільш інформативні при пошуку доріг на кадрах, отриманих при аерофотозйомці [46]. Нами були розглянуті найбільш перспективні для вирішення завдання: RGB, HSV, HSL, Lab і LCH.

Для аргументації вибору даних моделей необхідно розглянути їх властивості. Модель RGB була обрана тому, що є найбільш популярною і широко використовується у комп'ютерній графіці, а RGB (скорочено від англ. Red, Green, Blue – червоний, зелений, синій) – адитивна колірна модель, що описує спосіб синтезу кольору, за якою червоне, зелене та синє світло накладаються разом, змішуючись у різноманітні кольори. Широко застосовується в техніці, що відтворює

зображення за допомогою випромінення світла [43]. У даній моделі колір кодується градаціями складових каналів (Red, Green, Blue), тому за збільшення величини градації якогось каналу — зростає його інтенсивність під час синтезу. Кількість градацій кожного каналу залежить від розрядності бітового значення RGB. Зазвичай використовують 24-бітну модель, у котрій визначається по 8 біт на кожен канал, і тому кількість градацій дорівнює 256, що дозволяє закодувати 16 777 216 кольорів (рис. 2.5). Як видно на рисунку модель слабо виділяє яскравість, що характерно для блискавки, тому з великою вірогідністю дана модель не буде найкращою.

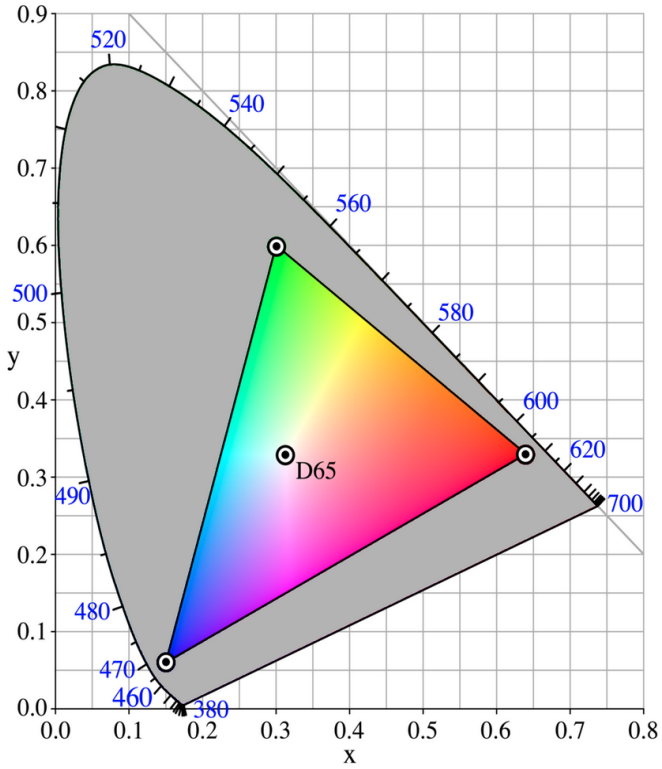

Рисунок 2.5 – Обмеження RGB по можливості передачі кольорів

HSV та HSL являють собою колірні моделі, засновані на трьох характеристиках кольору: колірному тоні (Hue), насиченості (Saturation) і значенні кольору (Value, в сенсі яскравість) та світністю (Lightness), тобто близькістю до білого кольору. Слід відмітити, що обрані моделі є різними. Вони були обрані оскільки кожна їх характеристика може виділити певну характеристику блискавки, яка являє собою білу, яскраву пляму з кольоровим контуром.

Найпростіший спосіб продемонструвати обидві моделі у тривимірному просторі – скористатися циліндричною системою координат (рис. 2.6). Тут координата H визначається полярним кутом, S – радіус-вектором, а V та L відповідно – Z-координатою. Тобто, відтінок змінюється при русі вздовж кола циліндра, насиченість – вздовж радіуса, а яскравість та світність відповідно – вздовж висоти. Таким чином значення H визначене від 0 до 360, а значення S, V, L – від 0 до 100.

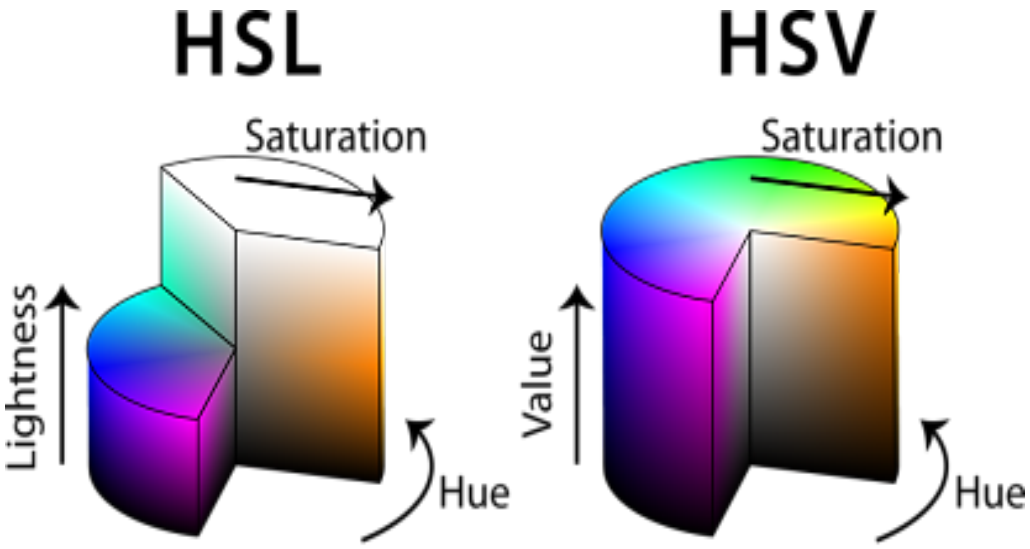

Рисунок 2.6 – Циліндричне представлення моделей HSL та HSV

Модель Lab (CIELab) [36] – це система задання кольорів, що використовує як параметри світлосилу (L), відношення зеленого до червоного (a) та відношення синього до жовтого (b). Ці три параметри утворюють тривимірний простір, точки якого відповідають певним кольорам. Колірна модель L*a*b розроблялась як апаратно-незалежна, тобто вона задає кольори без врахування особливостей відтворення кольорів. У колірному просторі Lab значення світлості відокремлено від значення хроматичної складової кольору (відтінок, насиченість) (рис. 2.7). Світлість задана координатою L (змінюється від 0 до 100, тобто від найтемнішого до найсвітлішого), хроматична складова – двома декартовими координатами a и b, які при представлені кольору у 24-бітній моделі в діапазоні від -127 до 128. Перша позначає положення кольору в діапазоні від зеленого до червоного, друга – від синього до жовтого.

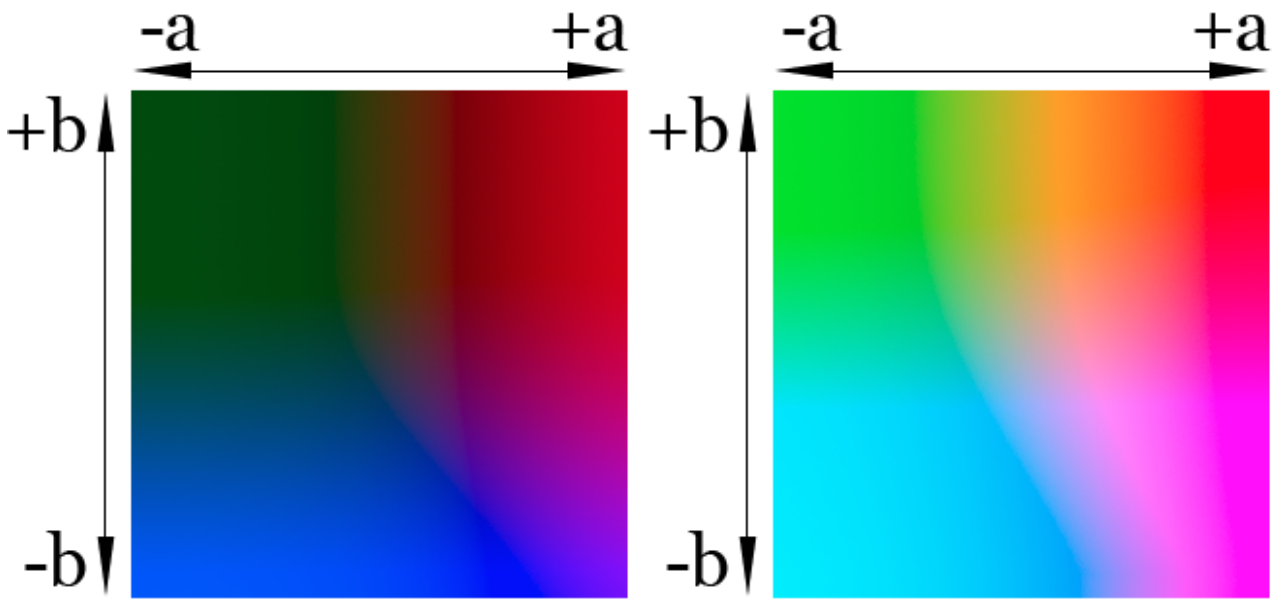

Рисунок 2.7 – Значення кольорів в залежності від L: 25% (ліворуч) та 75% (праворуч)

На відміну від кольорових просторів RGB, Lab однозначно визначає колір. Тому Lab широко використовується в програмному забезпеченні для обробки зображень як проміжного кольорового простору, через який проходить конвертування даних між іншими кольоровими просторами. При цьому особливі властивості Lab зробили редагування в цьому просторі потужним інструментом корекції кольору. Завдяки характеру визначення кольору в Lab з'являється можливість окремо впливати на яскравість, контраст зображення і на його колір. У багатьох випадках це дозволяє прискорити обробку зображень. Дана модель була обрана, оскільки показник L зможе потенційно відділити пікселі блискавки та пікселі фону за своїм значенням.

Під час пошуку особливостей блискавок було помічено, що всі блискавки мають низький показник C моделі LCH, до того ж L має той самий сенс, що й у моделі Lab. До того ж, показник H, так само як в моделях HSL та HSV, може потенційно виділити блискавки від фону. LCH [36] – це скорочення від Lightness, Chroma, Hue (світлосила, кольоровість, тон). Компоненти кольору в цій колірній моделі мають деякий зв'язок з компонентами колірної моделі HSL. Але між компонентами цих колірних моделей є серйозні відмінності.

Кути тони в LCH не цілком відповідають тонам HSL. У колірному просторі HSL насиченість представлена діапазоном значень від 0 до 100 та представляє собою результат простої трансформації RGB в полярні координати. Як зазначалось раніше, значення C, теоретично, є необмеженим, оскільки модель Lab, від компонентів якої розраховується даний показник може мати різні результати конвертації в залежності від обраного балансу білого. CIE LCH застосовуються у багатьох сферах: як програмний код для генерації зразків кольорів у інструментах статистики, як самостійні інструменти для проектування та тестування зразків або як бібліотеки, що дозволяють іншим програмам використовувати кольоровий простір.

2.2.2 Визначення колірної моделі для ідентифікації блискавок

Задля визначення найефективнішої моделі кольору для виокремлення блискавок серія експериментів була проведена за наступним сценарієм: відібрані зображення з джерел [41][42]. За допомогою графічного редактору з них вручну виокремлені блискавки (рис. 2.8)

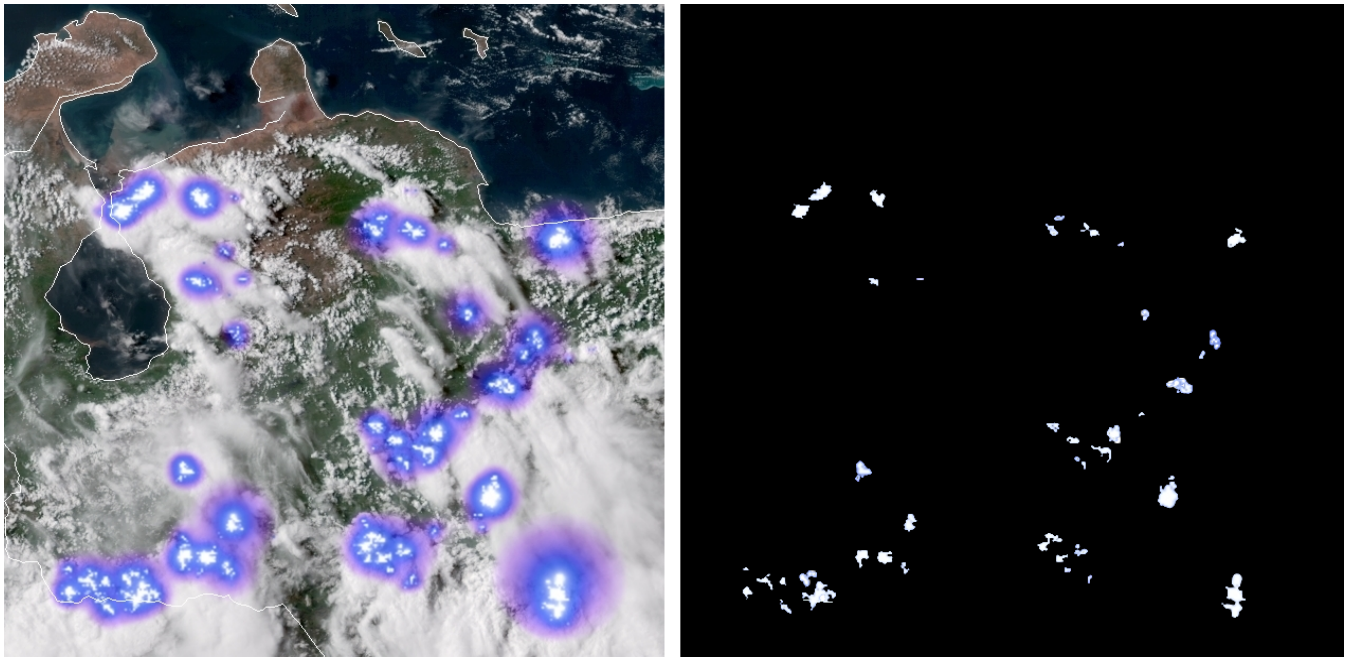

Рисунок 2.8 – Приклад даних для аналізу кольору блискавок

Для дослідження кольорового складу зображень була розроблена програма (рис. 2.9), яка в якості вхідних даних приймає два зображення: оригінальне та зображення з вирізаними блискавками на прозорому або чорному фоні. Програма у

кожному каналі кольорової моделі підраховує кількість пікселів певної величини. Результатом обробки є графіки, які демонструють ці дані для другого та першого зображення різними кольорами (для початкового кадру – зеленим, а для кадру з виокремленими блискавками – синім).

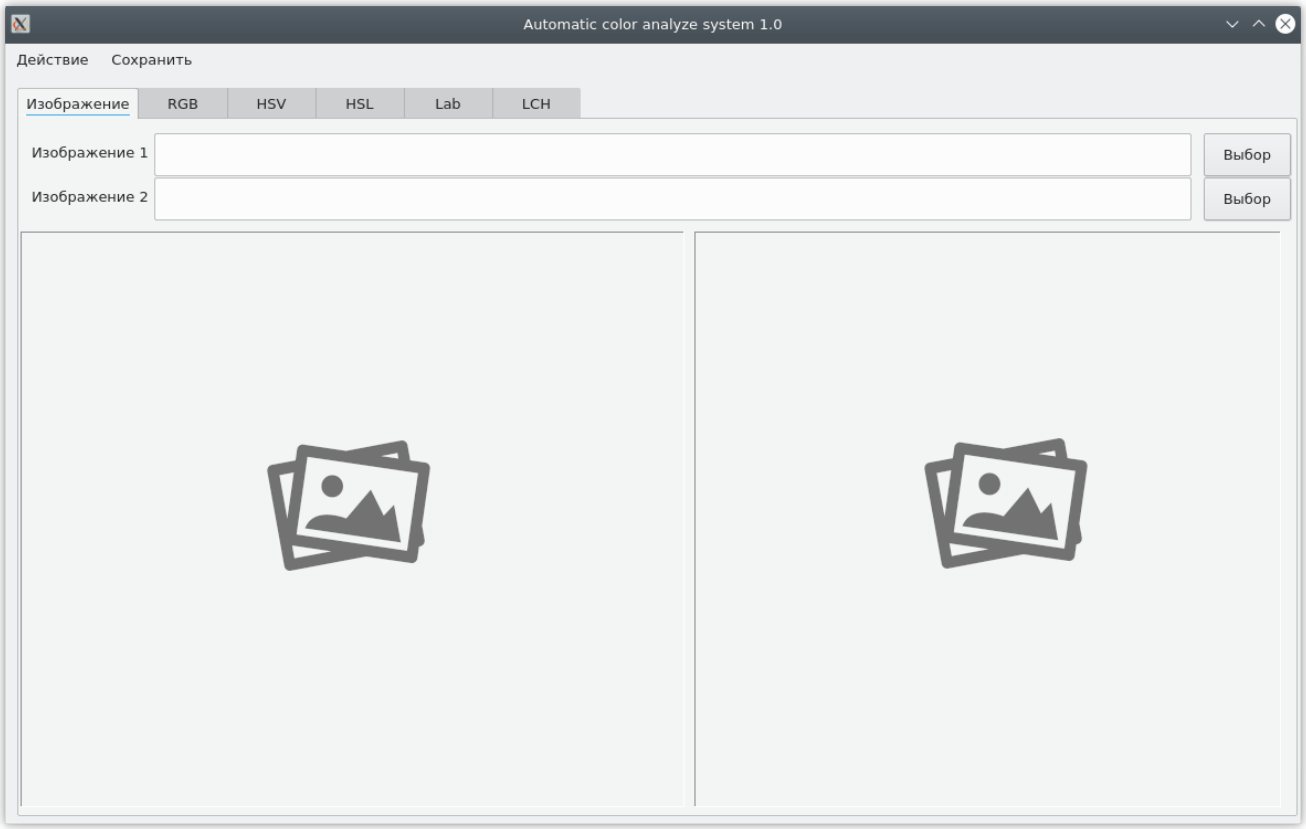

Рисунок 2.9 – Головне вікно програми для дослідження кольорового складу

Необхідно визначити такий канал, на якому для ненульових значень зображення з виокремленими блискавками, різниця була б найменшою. Однак, аналіз кольору блискавок не дав позитивного результату навіть після різних варіантів попередньої обробки: видалення каналу, виділення контурів методом різниці Гаусса, їх різноманітне комбінування. Всі обрані колірні моделі у всіх показниках демонстрували значну різницю між кількістю пікселів, що належить до одного значення показника (рис. 2.10). Даний розподіл є найкращим з усіх отриманих. Як видно, більша частина синього діапазону знаходиться на зеленому фоні, що означає наявність досить великої кількості пікселів фону з такою ж характеристикою, як і у пікселів блискавки.

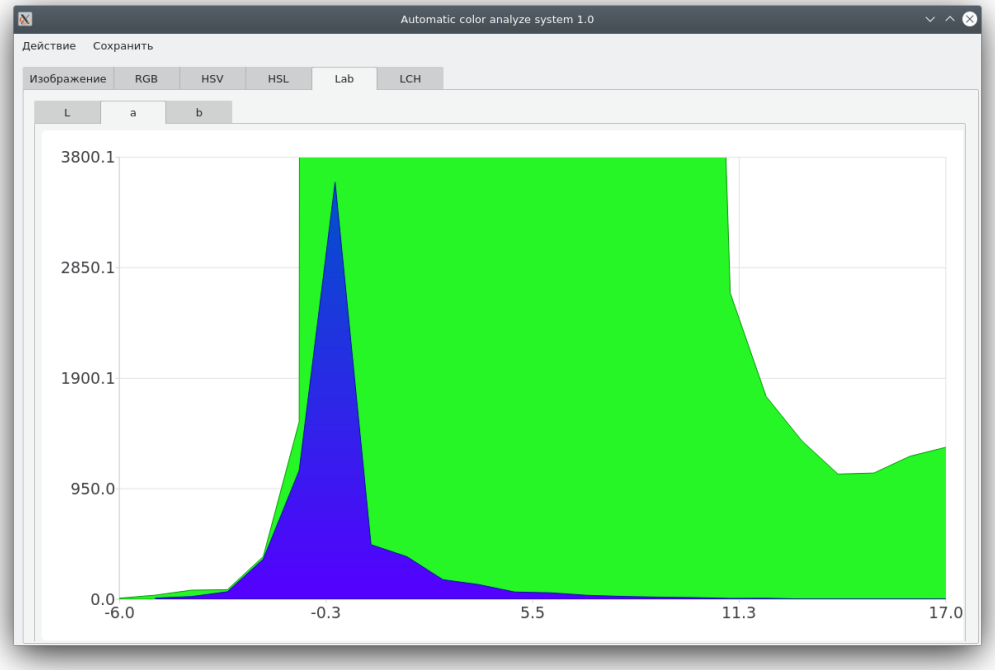

Рисунок 2.10 – Розподіл значень за a-координатою для моделі Lab для кадру

Однак, виявлено, що головною особливістю таких форматів є наявність чіткого ореолу навколо блискавок. Тому замість аналізу самих блискавок був проаналізований їх контур (рис. 2.11).

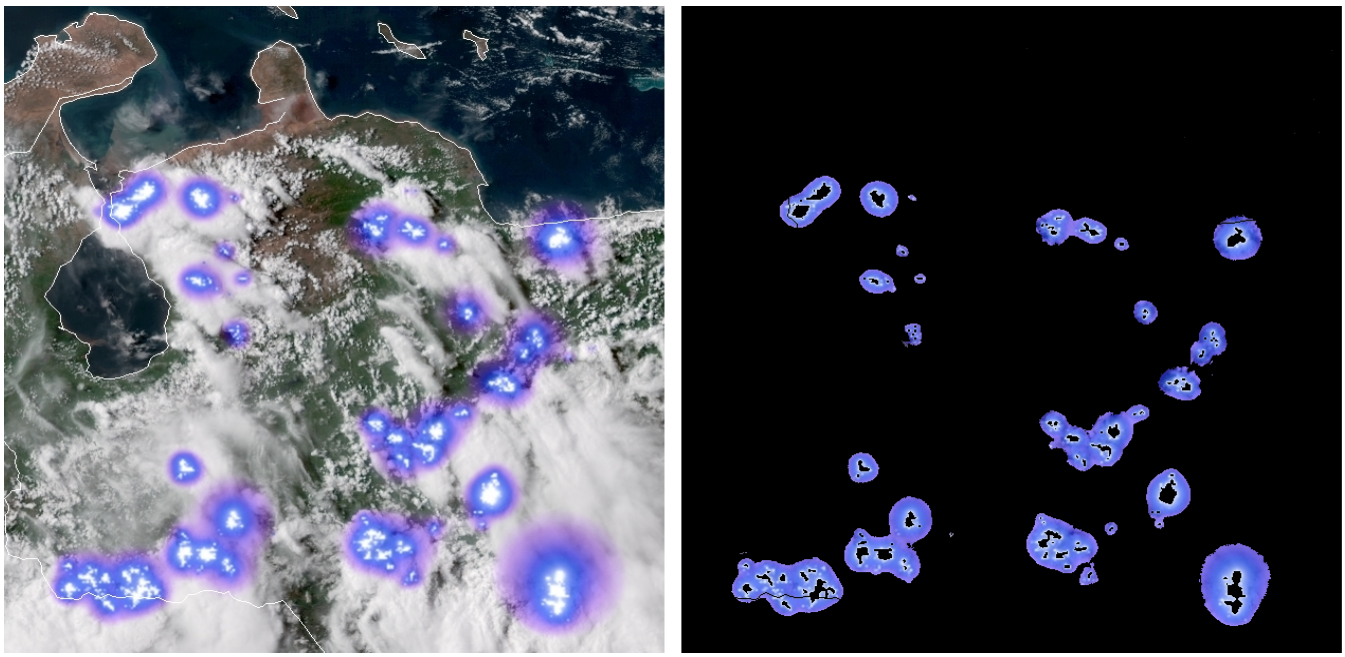

Рисунок 2.11 – Приклад даних для аналізу кольору контуру блискавок

Результати дослідження показали, що найкраще для ідентифікації контуру розглядати його в моделях Lab і LCH, оскільки низькі значення показника b присутні лише на контурах, а високі – лише в незначній кількості за його межами, а показник H займає досить вузьку частину шкали. Однак, дані показники ще не придатні для виокремлення блискавок, розподіли за пікселями все ще не досягли припустимого рівня. Для цього необхідно налаштувати більш точне вирізання контуру, оскільки для попереднього аналізу контур був досить грубо вирізаний вручну. Більш точне виділення контуру досягається шляхом сегментації того, що існує наразі за шкалою показника H: інтервал поділений на три частини частини (низькі, середні та високі значення), оскільки відтінок ореолу збільшується при віддалені від центру самої блискавки. З контуру, що був вирізаний вручну, виокремлені пікселі, що відповідають кожному виділеному інтервалу (рис. 2.12).

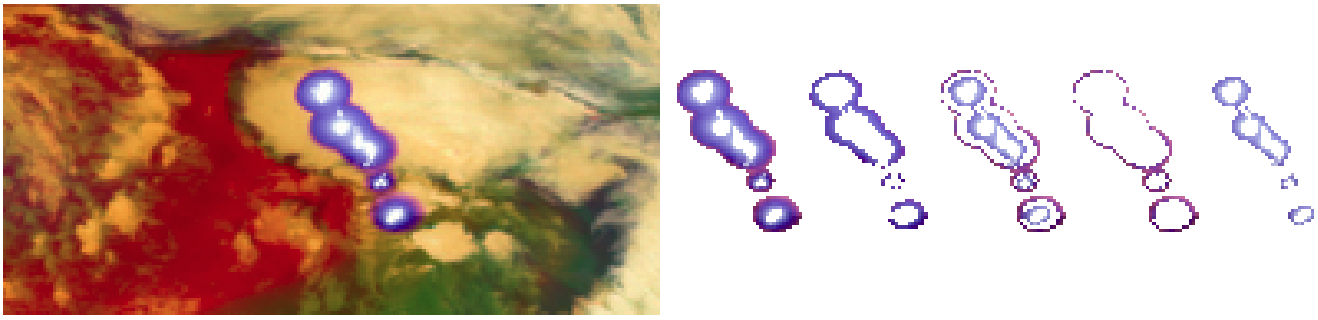

Рисунок 2.12 – Початкове зображення та виокремлені контури за різними показниками (зліва направо): вирізані вручну, середня частина H-діапазону, залишок після виокремлення, висока частина H-діапазону та низька

Аналіз кожного контуру окремо, а також варіанти їх різноманітної комбінації, показали, що найбільш ефективний розподіл кольорів у моделях Lab та LCH дає комбінація низької та середньої частини діапазону. Таким чином для ідентифікації блискавки придатна лише частина ореолу, розподіл якою а b-координатою в результаті її аналізу виглядає так, як показано на рис. 2.13, що є цілком придатним для подальшої роботи.

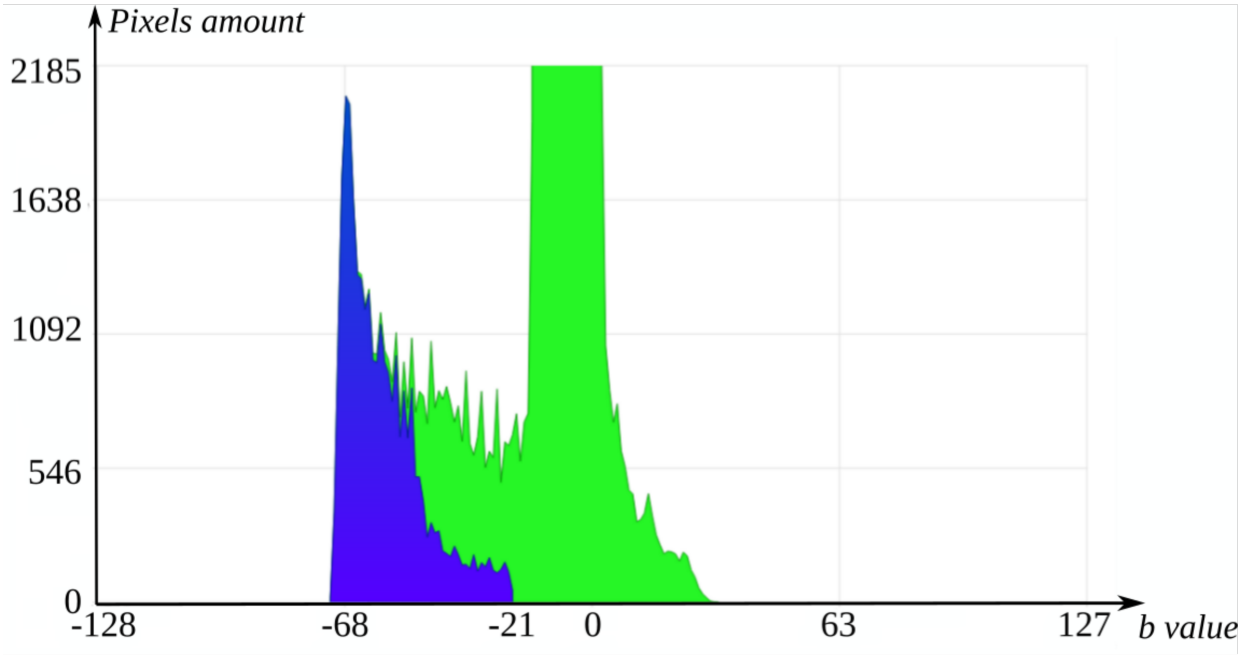

Рисунок 2.13 – Піктограма частоти кольору в каналі b моделі Lab

Для фільтрації зображень з метою виокремлення контурів блискавки в подальшому будуть використатися обидві моделі кольору (Lab та LCH), оскільки багато досліджень показують, що точність розпізнавання об'єкта зростає при комбінуванні двох [47], трьох [48] і навіть семи [49] моделей.

2.2.3 Визначення належності пікселя до контуру зображень

Після того, як визначенні показники моделей за якими можна ідентифікувати блискавку, необхідно розробити формули за якими можна було в віднести піксель до блискавки чи до фону. З джерел [41][42] обрано 5 кадрів, для яких результат розподілу показників кольору виявився приблизно однаковим та є найбільш позитивним в моделях Lab та LCH оскільки низькі значення показника b присутні лише на контурах, а високі – лише в незначній кількості за його межами, а показник H займає досить вузьку частину шкали. Схема застосування цих кольорових моделей показана на рис. , що є адаптацією схеми, показаної на рис. 2.3.

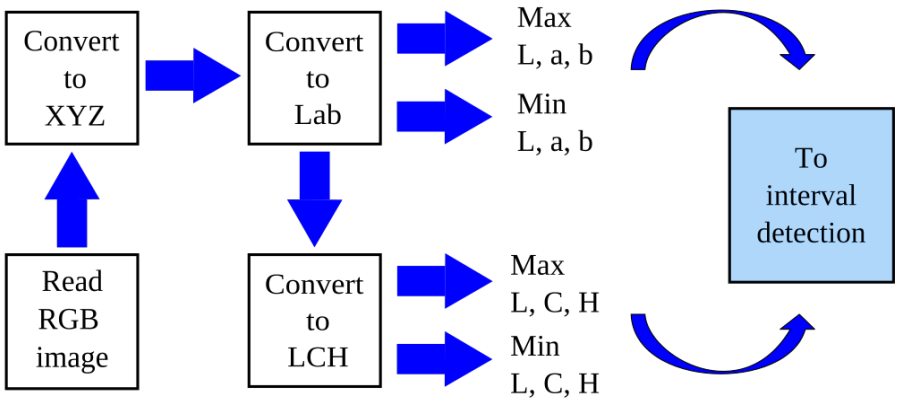

Рисунок 2.14 – Діаграма обробки для визначення кольорового інтервалу

Діапазони значень в цих моделях наведені в табл. 2.1. Показник L має однакове значення в обох моделях.

Таблиця 2.1 – Діапазони кольору ореолів блискавок в моделях Lab і LCH

	L	a	b	C	H
Кадр 1 [41]	(24, 85)	(1, 42)	(-68, -21)	(19, 79)	(274, 302)
Кадр 2 [41]	(22, 89)	(-2, 40)	(-69, -17)	(14, 80)	(270, 299)
Кадр 3 [41]	(24, 90)	(0, 41)	(-67, -14)	(18, 78)	(272, 301)
Кадр 4 [42]	(21, 92)	(0, 43)	(-61, -13)	(10, 79)	(270, 302)
Кадр 5 [42]	(25, 87)	(2, 39)	(-63, -15)	(8, 82)	(273, 301)
Середнє	(23, 89)	(0, 41)	(-66, -16)	(14, 80)	(272, 301)
Результат	**(22, 91)**	**(-3, 44)**	**(-70, -12)**	**(11, 81)**	**(268, 303)**
% шкалою	**69**	**18**	**23**	**70**	**10**

Значення рядка «середнє» обчислюються таким чином:

$$c_{lk}=\frac{\sum_{i=1}^{n} c_{ilk}}{n};c_{uk}=\frac{\sum_{i=1}^{n} c_{iuk}}{n}, \quad (2.15)$$

де c_{ilk} та c_{iuk} – це нижня та верхня межа показника кожного i-го кадру у k-му каналі, а n – загальна кількість кадрів у експерименті, де $n=5$. Використовуються наступні значення k:

– $k=1$ для L -координати моделей Lab та LCH;
– $k=2$ для a -координати моделі Lab;
– $k=3$ для b -координати моделі Lab;
– $k=4$ для C -координати моделі LCH;
– $k=5$ для H -координати моделі LCH.

Значення рядка «результат» обчислюються за формулою:

$$l_k = c_{lk} + 0{,}005 * R_k * \min_i (c_{ilk} - c_{lk}),$$
$$u_k = c_{uk} + 0{,}005 * R_k * \max_i (c_{iuk} - c_{uk}), \quad (2.16)$$

де 0,005 – коефіцієнт розширення, отриманий експериментально, R_k – число можливих значень у кольоровому k-каналі:

– $R_1=100$ (для L -координати моделей Lab та LCH);
– $R_2=256$ (для a -координати моделі Lab);
– $R_3=256$ (для b -координати моделі Lab);
– $R_4=100$ (для C -координати моделі LCH);
– $R_5=360$ (для H -координати моделі LCH).

Відсоток за шкалою розраховується за формулою:

$$percent = \frac{u_k - l_k}{R_k} * 100\,\% \quad (2.17)$$

Фільтри були розроблені для виявлення ореолів блискавки. Ці функції визначають d_{ij} у (2.13).

Функція лінійного фільтра:

$$d_{ij} = \sum_{k=1}^{K} f(x_{ijk}, lk, uk) - K + 1,$$
$$f(x_{ijk}, l_k, u_k) = \begin{cases} 1, якщо\ l_k < x_{ijk} < u_k, \\ 0, otherwise \end{cases} \quad (2.18)$$

де $K=5$ та визначає кількість каналів, що розраховуються, x_{ij} – значення пікселя з використанням координат:

– $k=1$ для L -координати моделей Lab та LCH;
– $k=2$ для a -координати моделі Lab;
– $k=3$ для b -координати моделі Lab;
– $k=4$ для C -координати моделі LCH;
$k=5$ для H -координати моделі LCH.

Функція квадратичного фільтра:

$$d_{ij}=1-\sum_{k=1}^{K}\frac{\left(x_{ijk}-\frac{(l_k+u_k)}{2}\right)^2}{\frac{(u_k-l_k)^2}{4}} \tag{2.19}$$

2.2.4 Визначення належності пікселя до блискавки

Використовуючи кадри з відфільтрованим контуром, необхідно виділити власне блискавки. Для цього розроблений наступний алгоритм: виконується аналіз відфільтрованого кадру, починаючи з лівого верхнього кута зліва направо, зверху вниз. Піксель кольору фону позначається як спалах блискавки, якщо:

$$z_{ij}=\begin{cases} x_{ij}\ if \sum_{k=j-1}^{k=j+1}\sum_{n=i-1,n\neq i\wedge k\neq j}^{n=i+1}\delta(y_{nk})\geq 3 \\ x_{ij}\ if \sum_{k=j-1}^{k=j+1}\sum_{n=i-1,n\neq i\wedge k\neq j}^{n=i+1}(\delta(y_{nk})+\delta(z_{nk}))\geq 4 \\ 0,\ інакше \end{cases}, \tag{2.20}$$

де x_{ij} – значення пікселя з координатами (i,j) вихідного кадру, y_{ij} – значення відповідного пікселя на відфільтрованого кадрі з виділеними ореолами блискавок, z_{ij} – значення пікселя на кадрі з виділеними спалахами блискавок, і

$$\delta(a)=\begin{cases} 1,\ якщо\ a\neq 0 \\ 0,\ інакше \end{cases}. \tag{2.21}$$

Такий підхід дозволяє працювати також з незамкненими контурами, що характерно для невеликих блискавок, втрата яких для джерела [42] критична. Однак, в результаті такого алгоритму з'являються частинки зовнішнього контуру. Для вирішення проблеми властивості кольору аналізували для блискавки, як це було зроблено для контуру з використанням тих самих п’яти зображень. Отримані діапазони значень та отриманий інтервал представлені в табл. 2.2. Всі рядки обчислюється так само, як і для табл. 2.1. У табл. 2.2. немає стовпця H-значення, оскільки його значення не складають одного інтервалу і є розрізненими.

Таблиця 2.2 – Діапазони кольору блискавок в моделях Lab і LCH

	L	a	b	C
Кадр 1 [41]	(52, 100)	(-5, 20)	(-48, 1)	(0, 51)
Кадр 2 [41]	(42, 100)	(-5, 19)	(-50, 1)	(0, 54)
Кадр 3 [41]	(56, 100)	(-3, 12)	(-43, 2)	(0, 46)
Кадр 4 [42]	(48, 99)	(-1, 10)	(-27, 3)	(0, 28)
Кадр 5 [42]	(41, 100)	(-1, 17)	(-40, 2)	(0, 43)
Середнє	(48, 100)	(-3, 16)	(-42, 2)	(0, 44)
Результат	**(45, 100)**	**(-6, 21)**	**(-52, 3)**	**(0, 49)**

Отже, для правильного виявлення блискавки (без шумів) (2.13) можна використовувати з фільтрами (2.18) або (2.19), використовуючи $K=4$ (2.18) і (2.19), так як зараз використовуються чотири кольорових канали замість п’яти. Нові коефіцієнти для фільтрів перераховуються розробленим програмним забезпеченням автоматично після зміни кількості каналів, що впливають на результат.

Однак відеозображення мають лінії, що показують межі регіонів, які складаються з тих самих кольорів, що й для блискавки. Для вирішення цієї проблеми був використаний підхід, розроблений для виявлення блискавки зі статичним фоном, оскільки ці лінії не рухаються від кадру до кадру. Таким чином поєднується кілька підходів з метою виявлення блискавки.

2.3 Інструменти конструктивно-продукційного моделювання

2.3.1 Узагальнений конструктор

В основі конструктивно-продукційного моделювання лежить концепція узагальненої конструктивно-продукційної структури [50][51][52], або узагальненого конструктора (УК):

$$C=\langle \mathrm{M},\Sigma,\Lambda\rangle, \tag{2.22}$$

де M – неоднорідний поповнюваний носій, Σ – сигнатура відносин і відповідних операцій, таких як конкатенація, підстановка і висновок, над атрибутами, Λ – набір тверджень інформаційного забезпечення конструктора (ІЗК), який включає в себе: онтологію, мету, правила, обмеження, строки початку і завершення конструювання.

Особливості конструктивно-продукційного моделювання наступні [50][51] [52]: можливість задати атрибути елементам і операціям, носій, який можна поповнювати, модель виконавця у вигляді основних алгоритмів, та зв'язок операцій з алгоритмами їх реалізації. Основні положення онтологічного забезпечення конструктивно-продукційного моделювання розроблені в [53].

Онтологія узагальненого конструктора в його неформальному поданні приведена в [50][51]. Сигнатура Σ включає в себе набори операцій: Ξ – зв'язування, Θ – підстановку і виведення, Φ – операції над атрибутами. Сигнатура також містить відносини підстановки «$\rightarrow$».

Операції зв’язування елементів конструктора об’єднують окремі елементи у конструкції або їх частини (проміжні форми).

Під формою ${}_{w_t}l$ з набором атрибутів w_l розуміємо:

– ${}_{w_t}l={}_{w_0}\otimes\left({}_{w_1}m_1,{}_{w_2}m_2,\ldots,{}_{w_k}m_k\right)$ для $\forall {}_{w_i}m_i\in\mathrm{M}$;

– ${}_{w_t}l={}_{w_0}m_j$, якщо $l={}_{w_0}\otimes\left(\varepsilon,\ldots\varepsilon,{}_{w_j}m_j,\varepsilon,\ldots\varepsilon\right)$;

– ${}_{w_t}l={}_{w_0}\otimes\left({}_{w_1}l_1,{}_{w_2}l_2,\ldots,{}_{w_k}l_k\right)$,

де ${}_{w_1}l_1,{}_{w_2}l_2,\ldots,{}_{w_k}l_k$ – форми, ${}_{w_0}\otimes$ – будь-яка операція зв'язування Ξ над атрибутом w_0, ε – порожній елемент.

Відносини підстановки – двомісні відносини з атрибутами – ${}_{w_i}l_i\rightarrow{}_{w_j}l_j$.

Нехай $S=\langle {}_{w_1}l_1 \rightarrow {}_{w_2}l_2, {}_{w_2}l_2 \rightarrow {}_{w_3}l_3, {}_{w_m}l_m \rightarrow {}_{w_{m+1}}l_{m+1}\rangle$ – послідовність відношень підстановки або $S=\varepsilon$, і $g=\langle \otimes_1(w_{1,1}, w_{2,1}, w_{k_{1,1}}), \otimes_2(w_{1,2}, w_{2,2}, w_{k_{2,2}}), \ldots, \otimes_3(w_{1,3}, w_{2,3}, w_{k_{3,3}})\rangle$ – послідовність операцій над атрибутами. Правило підстановки – $\psi:\langle s, g\rangle$. Тут $\otimes$ – це будь-яка операція над атрибутами $\otimes \in \Phi$.

Множину правил підстановки будемо позначати $\Psi=\{\psi_i:\langle s_i, g_i\rangle\}$.

Припустимо, що зазначена форма ${}_{w_t}l=\otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_h}l_h, \ldots, {}_{w_k}l_k)$ і відношення підстановки ${}_{w_h}l_h \rightarrow {}_{w_q}l_q$ є такими, що ${}_{w_h}l_h \prec {}_{w_t}l$ (відношення $\prec$ – містить), то результатом ${}_{w_t^*}l^*$ тринарної операції підстановки $\Rightarrow({}_{w_h}l_h, {}_{w_q}l_q, {}_{w_t}l)$ буде форма ${}_{w_t^*}l^*=\otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_q}l_q, \ldots, {}_{w_k}l_k)$, де $\Rightarrow \in \Theta$.

Двомісна операція часткового виведення ${}_{w_t^*}l^*=|\Rightarrow_{v_p}(\Psi, {}_{w_t}l)$ ($|\Rightarrow \in \Theta$) полягає у:

– виборі одного із доступних правил підстановки $\psi_r:\langle s_r, g_r\rangle$ із відношеннями заміщення s_r;

– виконання операцій підстановки на його базі;

– виконання операцій над атрибутами g_r.

Операція повного виведення ($\|\Rightarrow \in \Theta$) полягає у поетапному перетворенні форм, починаючи з початкової форми і закінчуючи конструкцією, що задовольняє умові завершення виведення, що передбачає циклічне виконання операцій часткового виведення. Це бінарна операція ${}_{w_t^*}l^*=|\Rightarrow_{v_p}(\Psi, {}_{w_t}l)$, де ${}_{w_t}l \in \mathrm{M}$.

Отримані конструкції операцій повного виведення належать до $\Omega(C)$.

З метою формування конструкцій проводиться ряд уточнюючих перетворень:

– спеціалізація визначає предметну область: семантичну природу носія, кінцевий набір операцій та їх семантику, атрибути операцій, порядок їх виконання та обмеження правил підстановки;

– інтерпретація зв'язує сигнатуру операцій з алгоритмами їх виконання, тим самим пов'язуючи інформаційну модель засобів формування конструкцій та модель виконавця, що породжують конструктивну систему;

– конкретизація конструктора розширює аксіоматику множиною правил підстановки;

– реалізація, суть якої полягає у формуванні набору конструкцій з використанням елементів носія.

Спеціалізацію узагальненого конструктора на основі конструктивно-продукційного підходу та L-систем [54] можна розглядати як:

$$C=\langle \mathrm{M},\Sigma,\Lambda\rangle_S \rightarrow C_L=\langle \mathrm{M}_L,\Sigma_L,\Lambda_L\rangle, \quad (2.23)$$

де M_L включає символьні термінали, а також проміжні форми та багатосимвольні конструкції, Σ_L включає одну операцію, тобто об'єднання символів та рядків символів (як правило, знак операції між операндами опускається), Λ_L – інформаційне забезпечення, що включає основи конструктивно-продукційного моделювання та особливості L-систем $\Lambda_L=\Lambda\cup\Lambda_1$.

Онтологія ІЗК Λ_1 включає зазначені вище позначення та їх семантику, поняття «символ», «конкатенація», «рядок символів» та інші відомі концепції багатосимвольної обробки, а також положення, наведені нижче.

Уточнюється операції часткового виведення $|\Rightarrow_{v_p}(\Psi,{}_{w_t}l)$: виконуються всі дозволені операції підстановки Ψ застосовних до терміналів форми ${}_{w_t}l$, переглядаючи зліва направо, крім випадків рекурсії.

Початкові умови подаються у вигляді рядка символів (аксіоми). Набір нетерміналів порожній.

Конструктивна система дозволяє генерувати певний набір конструкцій (можливо, один) або виконувати перевірку віднесення зазначеної конструкції до вищезазначеного набору.

У деяких випадках необхідно формувати два або більше різних наборів конструкцій аналогічним чином, де набори конструкцій, що утворюються, різні, а процеси їх утворення мають незначну варіативність.

У ряді випадків виникає необхідність схожим чином формувати дві або більше відмінних один від одної множини конструкцій. Іншими словами, множини формованих конструкцій різні, а процеси їх формування мають незначну варіативність.

У таких випадках доцільно застосовувати параметричні конструктори. Назвемо сімейством конструкторів множину конструкторів, що відрізняються обмеженою кількістю положень інформаційного забезпечення. При визначенні сімейства в круглих дужках задаються параметри конструкторів (перераховуються варіативні в рамках сімейства елементи ІЗК).

2.3.2 Параметричні багатосимвольні конструктори

Конкретизуємо C_L до рівня сімейства параметричних багатосимвольних конструкторів:

$$C_L=\langle \mathrm{M}_L,\Sigma_L,\Lambda_L\rangle_K, C_{MS}(B,P,n)=\langle \mathrm{M}_{MS},\Sigma_{MS},\Lambda_{MS}\rangle, \tag{2.24}$$

де B – початковий ланцюг символів (аксіома), P – множина правил постановки, n – кількість операцій часткового виведення, $M_{MS}\supset\{+,-,[,]\}$, $\Sigma_{MS}=\Xi_{MS}=\{\circ\}$, $\circ$ – операція конкатенації, $\Lambda_{MS}=\Lambda_L\cup\Lambda_2$. Онтологічна складова Λ_2 включає наведені вище позначення та їх семантику, а також наступні положення:

– мета конструювання – формування багатосимвольного ланцюжка фрактальної структури;

– правила підстановки задаються параметром P;

– обмеження – операції над атрибутами відсутні, правила підстановки містять єдине відношення підстановки;

– початкові умови – аксіома задається параметром B;

– умова завершення – виконання операцій часткового виведення.

В результаті інтерпретації формуємо конструктивну систему як сукупність двох моделей: конструктора і внутрішнього виконавця (остання у вигляді конструктора алгоритмів, які здатний виконати виконавець)

$$\begin{aligned}&\langle C_{MS}(B,P,n)=\langle \mathrm{M}_{MS},\Sigma_{MS},\Lambda_{MS}\rangle, C_A=\langle \mathrm{M}_A,\Sigma_A,\Lambda_A\rangle\rangle_I,\\ &C_{A,MS}(B,P,n)=\langle \mathrm{M}_{A,MS},\Sigma_{A,MS},\Lambda_{A,MS}\rangle,\end{aligned} \tag{2.25}$$

де C_A – модель виконавця у вигляді конструктора, який здатен виконувати базові та сконструйовані операції, M_A – набір базових $\{A_1^0|_{A_i,A_i}^{A_i,A_j}, A_2^0|_{Z_1,Z_2,A_i}^{A_i}, A_3^0|_{l_i,l_j}^{l_i,\circ l_j}\}\subset M_A$ та

сконструйованих $\{A_4|^{l_j}_{l_h,l_q,l_i}, A_5|^{l_j}_{l_i,\Psi}, A_6|^{l_j}_{l_i,\Psi}\}\subset\Omega(C_A)$ алгоритмів; $\Sigma_A=\{;:\}$ включає сигнатуру операцій послідовного та умовного виконання алгоритмів; ІЗК Λ_A приведено в [51]; $M_{A,MS}=\langle M_{MS}, M_A\rangle$, $\Sigma_{A,MS}=\langle\Sigma_{MS},\Sigma_A\rangle$, $\Lambda_{A,MS}=\Lambda_{MS}\cup\Lambda_A\cup\Lambda_3$.

Алгоритми M_A реалізують операції:

– виконання операції композиції $A_1^0|^{A_i,A_j}_{A_i,A_j}$ ($A|^Y_X$ – алгоритм над даними з вхідного набору X з результатними значеннями з набору Y, A^0 – генерація алгоритму), $A_i, A_j\in\Omega(C_{A,MS}), A_i\cdot A_j$ послідовне виконання алгоритму A_j після алгоритму A_i;

– умовне виконання $A_2^0|^{A_i}_{Z_1,Z_2,A_i}$, що полягає у виконанні A_i алгоритму за умови $Z_1\supseteq Z_2$;

– конкатенації ланцюгів символів $A_3^0|^{l_i,\circ l_j}_{l_i,l_j}$, $l_i, l_j\in M_{MS}$;

– виконання операції підстановки $A_4|^{l_j}_{l_h,l_q,l_i}$, $l_i, l_j, l_h, l_q\in M_{MS}$, l_i, l_j – поточна форма, в якій операція підстановки виконується до і після її виконання, l_h, l_q – ланцюжки в лівій та правій частині відношення підстановки;

– виконання операції часткового і повного виведення $A_5|^{l_j}_{l_i,\Psi}, A_6|^{l_j}_{l_i,\Psi}$, $\Psi\subset\Lambda_{MS}$ – множина правил підстановки.

ІЗК Λ_3 включає приведені вище визначення, позначення та їх семантику:

$$\Lambda_3=\Big\{(A_1^0|^{A_i\cdot A_j}_{A_i,A_j}\leftarrow\cdot),(A_2^0|^{:A_i}_{Z_1,Z_2,A_i}\leftarrow:),(A_3^0|^{l_i,\circ l_j}_{l_i,l_j}\leftarrow\circ),(A_4|^{l_j}_{l_h,l_q,l_i}\leftarrow\Rightarrow),\ (A_5|^{l_j}_{l_i,\Psi}\leftarrow|\Rightarrow),(A_6|^{l_j}_{l_i,\Psi}\leftarrow||\Rightarrow)\Big\}. \quad (2.26)$$

2.3.3 Сімейство параметричних конструкторів-перетворювачів

Сімейство параметричних конструкторів-перетворювачів з конструкції у вигляді ланцюжка символів в конструкцію у вигляді зображення:

$$C_L=\langle \mathrm{M}_L,\Sigma_L,\Lambda_L\rangle_{R\vdash} C_{GF}(\Omega_i(C_{MS}),\alpha,ATTR)=\langle \mathrm{M}_{GF},\Sigma_{GF},\Lambda_{GF}\rangle, \quad (2.27)$$

де $\Omega_i(C_{MS})$ – ланцюжки символів, отримані в результаті реалізації конструктора C_{MS}; α – кут, $ATTR$ – атрибути не-терміналів, такі як довжина та ширина лінії,

M_{GF} включає безліч терміналів T (множина усіх можливих кривих на площині та символів $\{+,-,[,]\}$), не-терміналів $N=\{A\}$, правил підстановки, $\Sigma_{MS}=\Xi_{MS}\cup\Phi_{GF}$, $\Xi=\{\circ f\}$, $\Phi_{GF}=\{*,+,-,\wedge,\&\}$, $\Lambda_{GF}=\Lambda_L\cup\Lambda_4$.

Позначимо відрізок ламаної ν з атрибутами $i\leftarrow\nu$ – порядковий номер при формуванні ламаної, $X\leftarrow\nu$ – координати початку, $l\leftarrow\nu$ – довжина, $h\leftarrow\nu$ – ширина, $\beta\leftarrow\nu$ – кут нахилу.

Введемо операції над атрибутами:

– додавання та віднімання відповідно $+(c,a,b)$, $-(c,a,b)$ з операндами a,b і результатом c;

– обчислення кінця X_{i+1},Y_{i+1} поточного відрізка (і початку наступного) $*(l,\beta,X_i,Y_i,X_{i+1},Y_{i+1})$ з початковими координатами X_i,Y_i, довжиною l і кутом нахилу β;

– збереження поточних координат X_i,Y_i і куту нахилу β у стек $\wedge(X_i,Y_i,\beta)$;

– присвоєння поточним координатам X_i,Y_i і куту нахилу β значень, вилучених зі стеку $\&(X_i,Y_i,\beta)$.

ІЗК Λ_4 включає наведені вище визначення, позначення і їх семантику, а також наступні положення:

– онтологія доповнюється відомими поняттями «площина», «відрізок», «дійсне число», «координати», «кут», та іншими, що дозволяють оперувати як з лінійними геометричними фігурами, так і з дійсними числами;

– мета конструювання – формування моделі блискавки типу хмара-земля на площині;

– правила підстановки:

$$\{\langle\langle A\rightarrow\nu A\rangle,\langle *(l,\beta,X_i,Y_i,X_{i+1},Y_{i+1})\rangle\rangle,$$

$$\langle\langle A\rightarrow +A\rangle,\langle +(\beta,\beta,\alpha)\rangle\rangle,\langle\langle A\rightarrow -A\rangle,\langle -(\beta,\beta,\alpha)\rangle\rangle,$$

$$\langle\langle A\rightarrow \wedge A\rangle,\langle \wedge(X_i,Y_i,\beta)\rangle\rangle,\langle\langle A\rightarrow \& A\rangle,\langle \&(X_i,Y_i,\beta)\rangle\rangle,$$

$$\langle A\rightarrow\varepsilon,\langle\varepsilon\rangle\rangle\};$$

– обмеження – правило $\langle A \to \varepsilon, \langle \varepsilon \rangle \rangle$ виконується, якщо не можна застосовувати всі інші;

– початкові умови – ланцюжок $\Omega_i(C_{MS})$; початкова точка X_0, Y_0; початковий кут $\beta = 270^\circ$, початковий номер точки $i = 0$;

– умова завершення – виконання правила $\langle A \to \varepsilon, \langle \varepsilon \rangle \rangle$.

Визначимо конструктивну систему, інтерпретувавши C_{GF}:

$$\begin{array}{l} \langle C_{GF}(\Omega_i(C_{MS}), \alpha, ATTR) = \langle \mathrm{M}_{GF}, \Sigma_{GF}, \Lambda_{GF} \rangle, C_B = \langle \mathrm{M}_B, \Sigma_B, \Lambda_B \rangle \rangle_{l^{\cdot}} \\ C_{B,GF}(\Omega_i(C_{MS}), \alpha, ATTR) = \langle \mathrm{M}_{B,GF}, \Sigma_{B,GF}, \Lambda_{B,GF} \rangle, \end{array} \quad (2.28)$$

де C_B – конструктор, який розширює можливості C_A наявністю сформованих алгоритмів $\{(A_7|_{a,b}^{c} \hookleftarrow +), (A_8|_{a,b}^{c} \hookleftarrow -)\} \subset \Omega(C_B)$, $M_B \supset M_A$, $M_{B,GF} = \langle M_{GF}, M_B \rangle$, $\Sigma_B = \Sigma_A, \Sigma_{B,GF} = \langle \Sigma_{GF}, \Sigma_B \rangle$, $\Lambda_B = \Lambda_A, \Lambda_{B,GF} = \Lambda_{GF} \cup \Lambda_B \cup \Lambda_5$.

ІЗК Λ_5 включає наведені вище позначення і їх семантику, а також визначення атрибутики операцій (алгоритмів, що їх реалізують):

$$\begin{array}{l} \Lambda_5 = \{(A_7|_{a,b}^{c} \hookleftarrow +), (A_8|_{a,b}^{c} \hookleftarrow -), (A_9|_{l,\beta,X_i,Y_i}^{X_{i+1},Y_{i+1}} \hookleftarrow *), \\ (A_{10}|_{X_i,Y_i,\beta}^{S} \hookleftarrow \wedge), (A_{11}|_{S}^{X_i,Y_i,\beta} \hookleftarrow \&)\}. \end{array} \quad (2.29)$$

Надалі виконується конкретизація шляхом задання значень параметрів в конструктивній системі і відповідна реалізація.

Висновки до розділу 2

Методологія виокремлення блискавок з відео, створених погодними супутниками була запропонована. Розроблена як методологія для виокремлення блискавок з відео, що демонструють статичний грозовий фронт, яка базується на порівнянні кольору пікселів двох сусідніх кадрів, так і методологія для виокремлення блискавок з відео, що мають рухомий грозовий фронт при сильно-динамічній хмарності, основу якої складає підхід оцінки кольору пікселя у колірних моделях Lab та LCH.

Були розроблені конструктори для моделювання блискавок типу-хмара земля, які для отримання результату необхідно конкретизувати.

3 ПРОЕКТУВАННЯ І РОЗРОБКА ІНСТРУМЕНТАРІЮ КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

3.1 Формалізація задачі з точки користувача, який виокремлює блискавки та користувача, що моделює блискавки типу хмара-земля

Формалізація задачі на рівні зовнішнього проектування представлена у вигляді діаграми прецедентів (Use-case diagram). На ній показано сукупність акторів і прецедентів та взаємозв’язки між ними (рис. 3.1). Актор – це особа, яка взаємодіє з системою. В даному випадку існують два актори – це користувач, який виокремлює блискавки та користувач, що моделює блискавки типу хмара-земля.

Користувач, який виокремлює блискавки виконує такі дії над програмою:

– завантаження кадрів з відеофайлів, створених метеосупутниками;
– виокремлення блискавок з вхідних зображень, результатом якого є нове зображення з блискавками на однотонному фоні; ця дія включає у себе: видалення каналу кольору, видалення, власне, фону, заповнення пустот та видалення шумів; розширюється дана дія створенням нового зображення, яке містить виокремлені блискавки на однотонному фоні;
– розрахунок параметрів блискавок з серії кадрів з виокремленими блискавками таких як: координати виникнення відносно верхнього лівого кута вхідного кадру та розмір у пікселях; дія розширюється можливістю зберегти дані у файл.

Користувач, який моделює блискавки типу хмара-земля виконує:

– конструювання моделі поширення блискавки типу хмара-земля у атмосфері, що включає у себе: збереження правил продукції у файл, завантаження правил продукції з файлу та, власне, створення правил продукції; дія розширюється можливістю подальшої візуалізації блискавки згідно з створеними правилами.

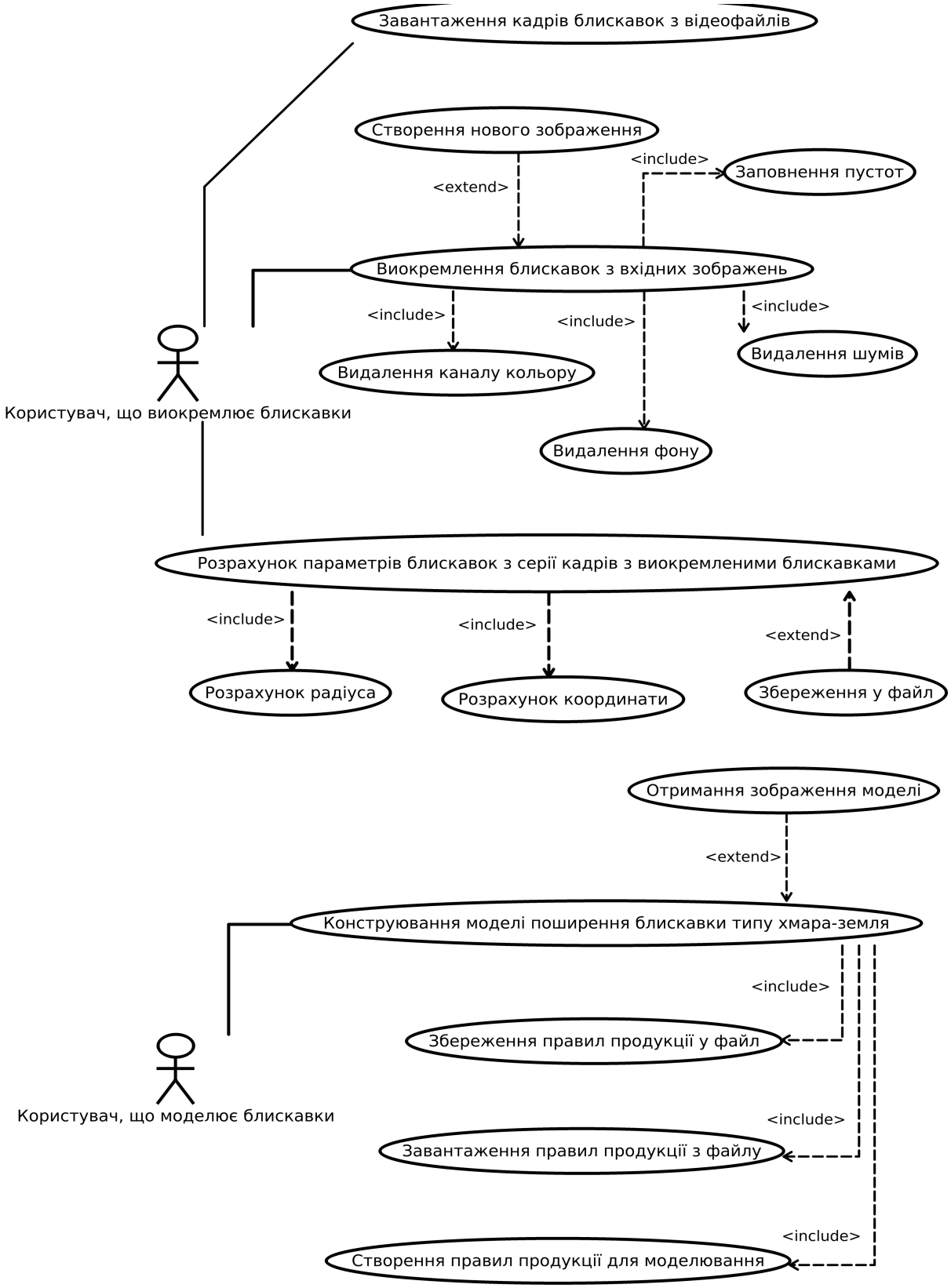

Рисунок 3.1 – Діаграма прецедентів системи конструктивно-продукційного моделювання молнієвої активності

3.2 Базова архітектура автоматизованої системи конструктивно-продукційного моделювання молнієвої активності

В процесі проектування було виконано декомпозицію модулів системи на 6 пакети відповідно до призначення класів:

– MainWindow – головний модуль;
– Dialog – модуль діалогів;
– Utils – загальновживані методи;
– Actions – контролер операцій над зображеннями;
– Spots – модуль для реалізації різних типів апроксимації блискавок;
– Productions – контролер операцій над конструкціями;

На рис. 3.2 представлена схема взаємодії основних модулів програми.

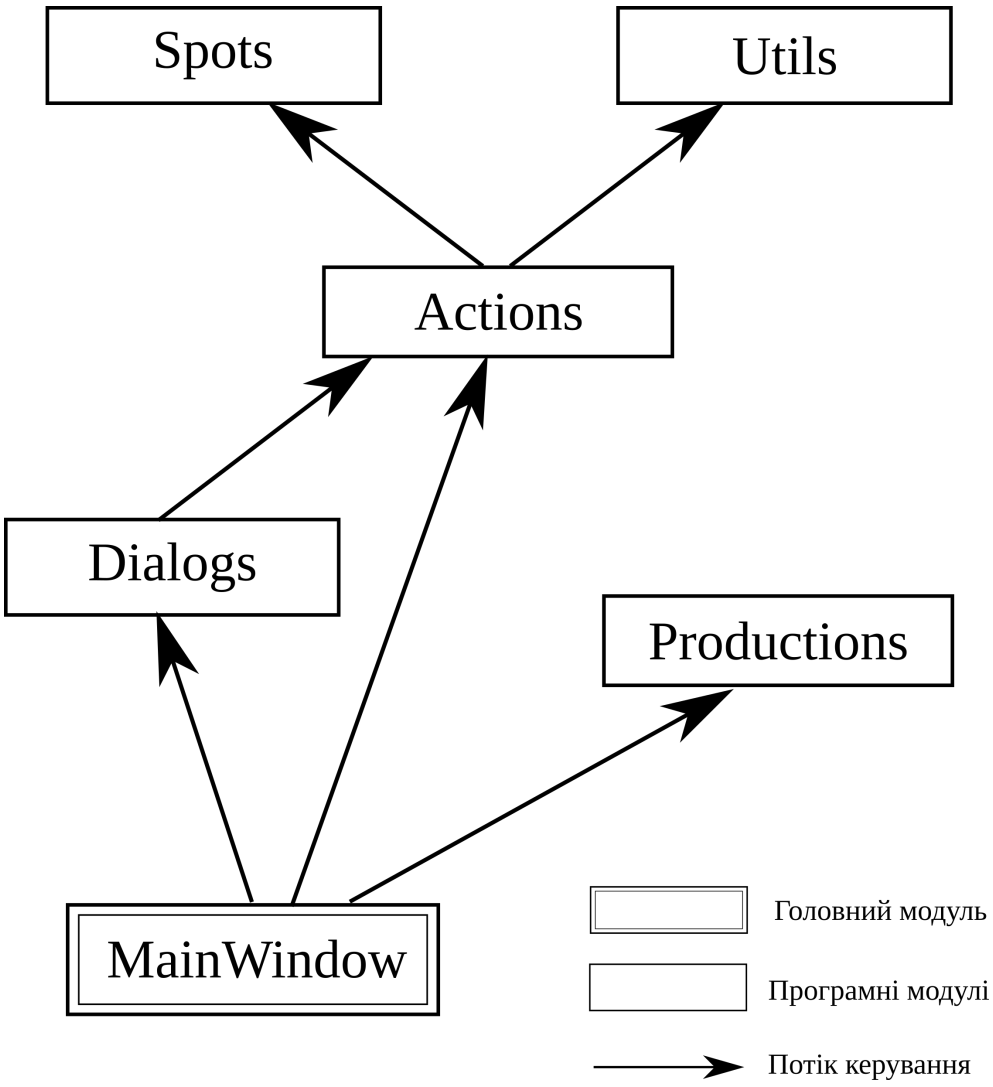

Рисунок 3.2 – Схема взаємодії основних модулів програми

Модуль MainWindow є головним модулем програми і організує взаємодію всіх інших модулів між собою. Відповідає за отримання даних від користувача (Dialogs), передачу їх до логіки програми (модуль Actions), а також демонстрацію даних користувачеві.

Модуль Dialog включає в себе різноманітні діалогові вікна для взаємодії з користувачем (прийом даних та оповіщення). Він використовує модуль Actions, оскільки отримує від користувача необхідні параметри для їх створення та, власне, створює їх.

Модуль Utils представляє собою модуль, який містить класи, що реалізують загальновживані складні логічні задачі. Дозволяє уникнути дублювання коду.

Модуль Actions представляє собою основну логіку програми. Даний модуль реалізує можливість застосовувати різноманітні методи обробки кадрів з відео, створених метеосупутниками, для пошуку потрібних даних на них а для відтворення спрощеного представлення блискавок грозового фронту. Використовує операції модуля Utils та Spots.

Spots необхідний для вибору типу спрощеного представлення блискавок. Модуль виокремлений для більш простого масштабування програми (додавання нової апроксимованої форми блискавки у грозовому фронті).

Модуль Poductions містить класи, які дозволяють будувати правила продукції блискавок типу хмара-земля і їх подальше відтворення.

3.3 Внутрішнє проектування системи виокремлення блискавок і конструктивно-продукційного моделювання

3.3.1 Вибір мови програмування з урахуванням критеріїв задачі виокремлення блискавок

Наразі вибір мови програмування починається з вибору критеріїв, які повинна задовольняти обрана мова. Найбільш складною задачею є розробка частини програми з виокремлення блискавок, остільки потребує багато розрахунків та організації обробки великої кількості даних.

За функціональністю розрізняють мови програмування на мови опису алгоритмів та мови опису специфікацій. Мови опису алгоритмів дозволяють створювати програми, які являють собою систему формальних приписів, спрямованих на вирішення конкретних завдань, які виконує ЕОМ. Мови опису специфікацій являють собою протилежну методологію розробки, при якій програма являє собою набір приписів, що ставлять комп'ютеру певне завдання в загальному вигляді, без написання формалізованого алгоритму. Хоча останні розроблялись, в першу чергу, для вирішення задач штучного інтелекту, їх суттєвими недоліками є нелінійна структура програм та відносно невисока ефективність реалізації. Такі особливості не підходять для реалізації поставленої задачі, тому у подальшому розглядаються лише мови опису алгоритмів.

За системою типів мови програмування поділяються на типізовані і нетипізовані. Нетипізовані мови дозволяють виконувати будь-яку можливу операцію над будь-якими даними. Це зазвичай будь-які мови асемблера, які працюють безпосередньо з двійковим поданням даних в пам'яті. Типізовані мови визначають які операції можна виконувати над певним типом даних, наприклад, операція віднімання не може бути застосована лише до числових типів, а для рядків символів – є невизначено. Хоча і нетипізовані мови дозволяють писати на гранично низькому рівні, виконувати будь-які операції над будь-якими типами даних, а одержуваний код зазвичай більш ефективний, що є головним критерієм вирішуваної задачі, такий підхід має багато недоліків. По-перше, ускладнюється процес написання програми, оскільки при такому підході виникають труднощі при роботі з комплексними типами даних (списки, рядки або структури). По-друге, відсутність перевірок призведе до помилок, які при налагоджені буде важко розпізнати. По-третє, низький рівень абстракції означає, що робота з будь-яким складним типом даних нічим не відрізняється від роботи з числами, що звичайно буде створювати багато труднощів. Таким чином, використання нетипізованої мови, призведе до збільшення часу написання та налагодженя програми, що є критичним при написані складних великих програм.

Типізовані мови за часом перевірки поділяються на статично та динамічно типізовані. При статичній типізації, типи всіх виразів точно визначені до виконання програми, і зазвичай перевіряються при компіляції. Мови зі статичною типізацією, в свою чергу можуть бути явно типізований (manifestly typed) або типовиводящі (type-inferred). При динамічній типізації перевірку типів виконується на етапі виконання. Такі мови не вимагають визначення типів виразі та пов'язані зі значенням при виконанні, а не з текстовим виразом. Крім того, одна змінна може зберігати різні типи даних при роботі програми. Однак, помилки типів не можуть бути автоматично виявлені, поки фрагмент коди не буде виконаний. Це ускладнює налагодження і підриває ідею типобезпечності в цілому. До того ж, час виконання програм при цьому зростає, оскільки вимагає визначення типів при виконанні програми.

За суворістю типізації мови діляться на сильно і слабо типізовані. Слабо типізовані мови неявно конвертують один тип в інший, наприклад, рядки в числа і навпаки. Це може бути зручно в деяких випадках, однак багато програмних помилок може бути пропущено. Ускладнюється налагодження. Сильно типізовані мови не дозволяють неявну конвертацію, і вимагають явної, що збільшую безпечність програм та дозволяє вияаити помилки ще на етапі синтаксичного аналізу.

Розрізняють компільовані та інтерпретовані мови. Компільовані мови передбачають побудову об'єктного модуля, що являє собою форму, що придатна для виконання на ЕОМ. Компіляція може проводитися безпосередньо в машинний код, або в будь-яке проміжне представлення (байт-код), яке потім інтерпретується віртуальною машиною. Інтерпретовані мови виконуються безпосередньо, без етапу компіляції.

Отже, був сформований наступний перелік критеріїв:

– швидкість виконання програми;
– простота розробки програми;
– простота пошуку помилок і налагодження;
– наявність інструментів для створення інтерфейсу користувача.

Для розробки проводився вибір серед наступних мов програмування: C++, C#, Python та Java.

Порівнюючи вищезазначені мови, можна виділити наступні переваги C++ серед інших, з урахування висунутих критеріїв:

– висока швидкість виконання програм, завдяки компіляції та відсутності збірників сміття;
– статична явна типізація;
– підтримка лямбда-виразів;
– можливість розробляти програми з сильною типізацією;
– проста логічна структура;
– підтримка парадигми ООП;
– наявність бібліотек для роботи з математичними виразами.

3.3.2 Технологічна платформа для розробки системи

При реалізації системи використовувалася наступна технологічна платформа [55].

Qt Framework – це повна система розробки програмного забезпечення. Qt містить повний набір класів бібліотеки C++ і завантажений API для спрощення розробки додатків. Qt дозволяє створювати читабельний, легко обслуговуваний і багаторазовий код з високою продуктивністю. До того ж, створюваний код є крос-платформовим. Відмінна особливість Qt від інших бібліотек – використання Meta Object Compiller (MOC) – системи попередньої обробки початкового коду (загалом, Qt – це бібліотека не для чистого C++, а для його особливого діалекту, з якого й «перекладає» MOC для подальшої компіляції будь-яким стандартним C++ компілятором). MOC дозволяє в багато разів збільшити потужність бібліотек, вводячи такі поняття, як слоти та сигнали.

Qt Designer – інструмент для створення графічного інтерфейсу користувача в інтерактивному режимі, що значно спрощує написання графічних програм. Ідеологія створення форм у Qt базується на використанні менеджерів розташування, котрі

надають «гумовий» дизайн, при якому розмір і розташування елементів форм визначаються автоматично, що значно прискорює розробку графічного інтерфейсу.

Qt Creator IDE – інтегроване середовище розробки, призначене для створення крос-платформових додатків з використанням бібліотеки Qt. Qt Creator може використовувати GCC або Microsoft VC++ як компілятор і GDB як засіб для налагодження програм. Містить вбудований редактор форм (*Qt Designer*) і довідкову систему (*Qt Assistant*), дозволяє розширення плагінами (наприклад, плагіну для тривимірного моделювання, який може знадобитись при моделюванні блискавки типу хмара-земля). Для створення проектів використовується qmake, середовище забезпечує зручне підсвічування синтаксису з можливістю створення своїх стилів.

3.3.3 Ієрархія та взаємодія класів системи

При проектуванні системи класів перше, що було розроблено – модуль Actions. Цей модуль у своїй архітектурі є досить простим, однак він виконує основні функції роботи програми. Його основу складає абстрактний базовий клас IAction, від якого наслідуються класи-дії. Клас IAction має три чистих віртуальні методи:

– process() – виконати обробку над об'єктом як того передбачає операція;
– getName() – отримати назву операції, яка непредставлена даним класом

та три віртуальних методи:

– reset() – повернути значення параметрів до значення параметрів за замовченням;
– prepare() – виконати підготовку певних параметрів, до старту циклу обробки;
– clear() – для очищення ресурсів, відкритих методом prepare().

Кількість класів обумовлена операціями, які необхідно виконувати для виокремлення блискавок та їх подальшої апроксимації:

– DeleteColorChennel – клас, що реалізує видалення колірного каналу;
– DeleteBackground – видалення фону, на якому розташовані спалахи блискавок грозового фронту;
– DeleteNoises – видалення шумів після видалення фону;
– FillEmpties – заповнення пустот, що утворились після видалення фону;

– FindSpots – пошук плям блискавок, їх апроксимація до більш простої форми та подальший розрахунок їх параметрів.
– CreateApproximateLightning – створення блискавок заданої апроксимованої форми, за отриманими раніше параметрами з подальшим розміщенням на зображеннях та їх збереженням.

Всі описані вище класи реалізують методи process() та getName(), а також перевизначають метод reset(), який викликає й метод reset() базового класу. Методи prepare() та clear() перевизначають лише класи FindSpots та CreateApproximateLightning. Дані класи використовують їх відкриття та закриття потоків роботи з файлом. Дані методи у базовому класі мають порожнє тіло.

При роботі над реалізацією методу process() класу DeleteBackground знадобилися наступні класи, які поміщені до модуля Utils:

– DecisiveFunction – клас, що реалізує визначення належності пікселя до заданого класу образів за встановленими параметрами кольору у моделях Lab або LCH та заданими правилами.
– ColorConvertor – клас, який реалізує конвертацію кольорів RGB до моделей Lab та LCH.

Оскільки виконання операцій з метою виокремлення блискавок – це набір операцій, які необхідно застосувати до вхідних зображень, то необхідно створити клас-контейнер, який містив би операції у необхідній послідовності. Клас Actions:

– add() – додати операцію у кінець;
– remove() – видалити операцію;
– moveUp() – перемістити вверх за списком;
– moveDown() – спустити вниз за списком;
– getSize() – отримати інформацію про кількість операцій у списку;
– get() – отримати доступ до заданої операції.

Діаграма класів модулю Actions та Spot представлена на рис. 3.3. Дані модулі представлені окремо, оскільки містять багато класів та зв'язків. Для того, щоб не перевантажувати діаграму класів програми представмо їх окремо.

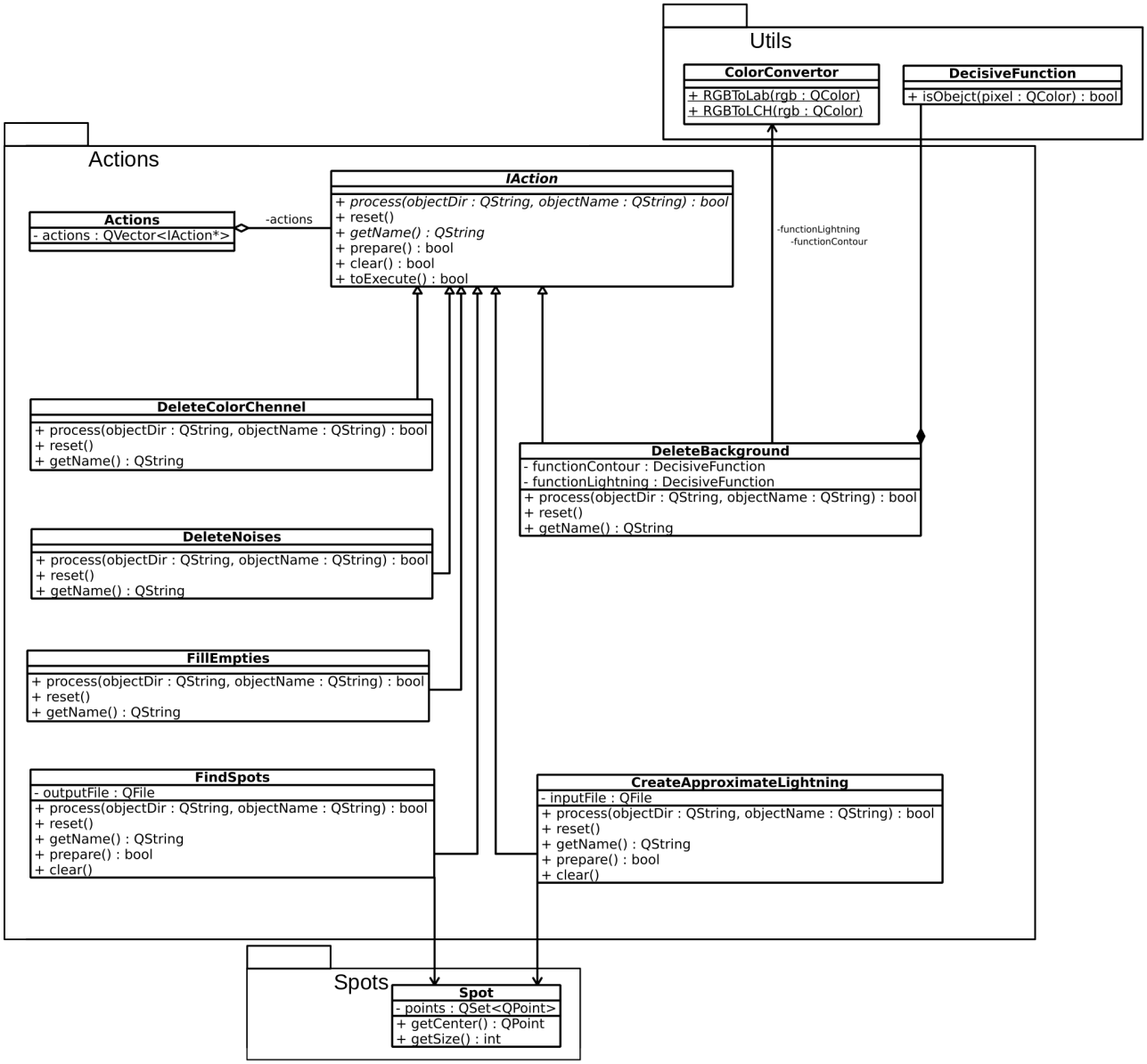

Рисунок 3.3 – Діаграма класів модулів Actions та Spot та використовувані ними класи Utils

Оскільки в ході роботи над програмою з'ясувалося, що для досягнення мети – тобто виокремлення блискавок – необхідно виконувати дії у приблизно однаковому порядку. Для спрощення реалізації створення набору методів був створений статичний метод CreateForPreProc(), який створює контейнер з наступними операціями у описаному порядку:

– DeleteColorChennel;

– DeleteBackground;

– DeleteNoises;

– FillEmpties;
– DeleteNoises;
– FindSpots.

Оскільки іноді необхідно виконати лише деякі методи для кожного елементу передбачене поле, яке визначає чи необхідно виконувати дану операцію. Для зняття або встановлення такого флагу відведенні відповідні методи set() та get().

Для приведення дій у виконання використовується клас ActionRunner. Він має публічний слот run(), який запускає цикли обробки над контейнером Actions та сигнали для сповіщення викликаючого класу:

– finished() – для сповіщення про виконання всіх дій з контейнеру;
– actionNameChanged() – перехід від однієї дії до іншої;
– processedQuantityChanged() – інформування щодо кількості зображень, які ще залишилось обробити.

Для отримання імен зображень використовується структура Source, яка містить наступні члени:

– from – номер зображення з якого необхідно почати обробку;
– to – номер зображення яким необхідно завершити обробку;
– nameGenerator – клас, який буде використаний для генерації імен файлів.

Розглянемо механізм генерації імен файлів. Його основу складає абстрактний базовий клас INameGenerator. Клас має два методи:

– getPath() – для отримання шляху до зображення;
– getName() – для отримання, власне, імені файлу за його номером.

Клас має дві реалізації FixedNameGenrator, який генерує імена у форматі, представленому на рис. 3.4.

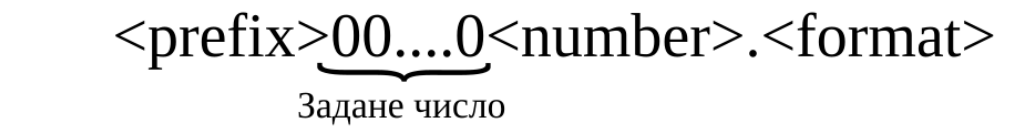

Рисунок 3.4 – Принцип побудови імені файлу класом FixedNameGenerator

SimpleNameGenerator будує імена у форматі <prefix><number>.<format>.

Розглянемо класи модуля Productions. Його складають такі класи:

– ProdRules – для зберігання та оперування правилами продукцій;
– DrawableModel – для відтворення згенерованої послідовності графічним способом,

а також структура NTermanalDescriptor, що описує графічні властивості не-терміналу створеної граматики та має наступні поля:

– lExpected – математичне очікування довжини лінії графічного представлення не-терміналу;
– lDispertion – дисперсія довжини лінії графічного представлення не-терміналу
– width – ширина лінії графічного представлення не-терміналу.

Для розуміння призначення класу ProdRules розглянемо його методи та опишемо їх призначення:

– addRule() – додає нове правило, яке містить ліву та праву частину окремо, а також визначає для не-терміналу у лівій частині правила NTermanalDescriptor структуру;
– updateRightPart () – обновити значення правої частини для не-терміналу;
– updateDescriptor() – обновити значення структури NTermanalDescriptor для не-терміналу;
– setAxiom() – визначити даний не-термінал аксіомою;
– save() – зберегти розроблені правила продукції у файл;
– load() – завантажити правила продукції з файлу;
– buildSequence() – побудувати послідовність згідно з правил, починаючи із визначеної аксіоми, за заданою кількістю переписувань.

DrawableModel має основний метод draw(), який будує за послідовністю, побудованої методом buildSequence() та властивостями не-терміналів модель блискавки типу хмара-земля.

Модуль Dialogs складається з класів SourceDialog та DeleteBackgroundDialog.

На основі описаної вище архітектури програми була побудована діаграма класів, спрощене представлення якої знаходиться на рис. 3.5.

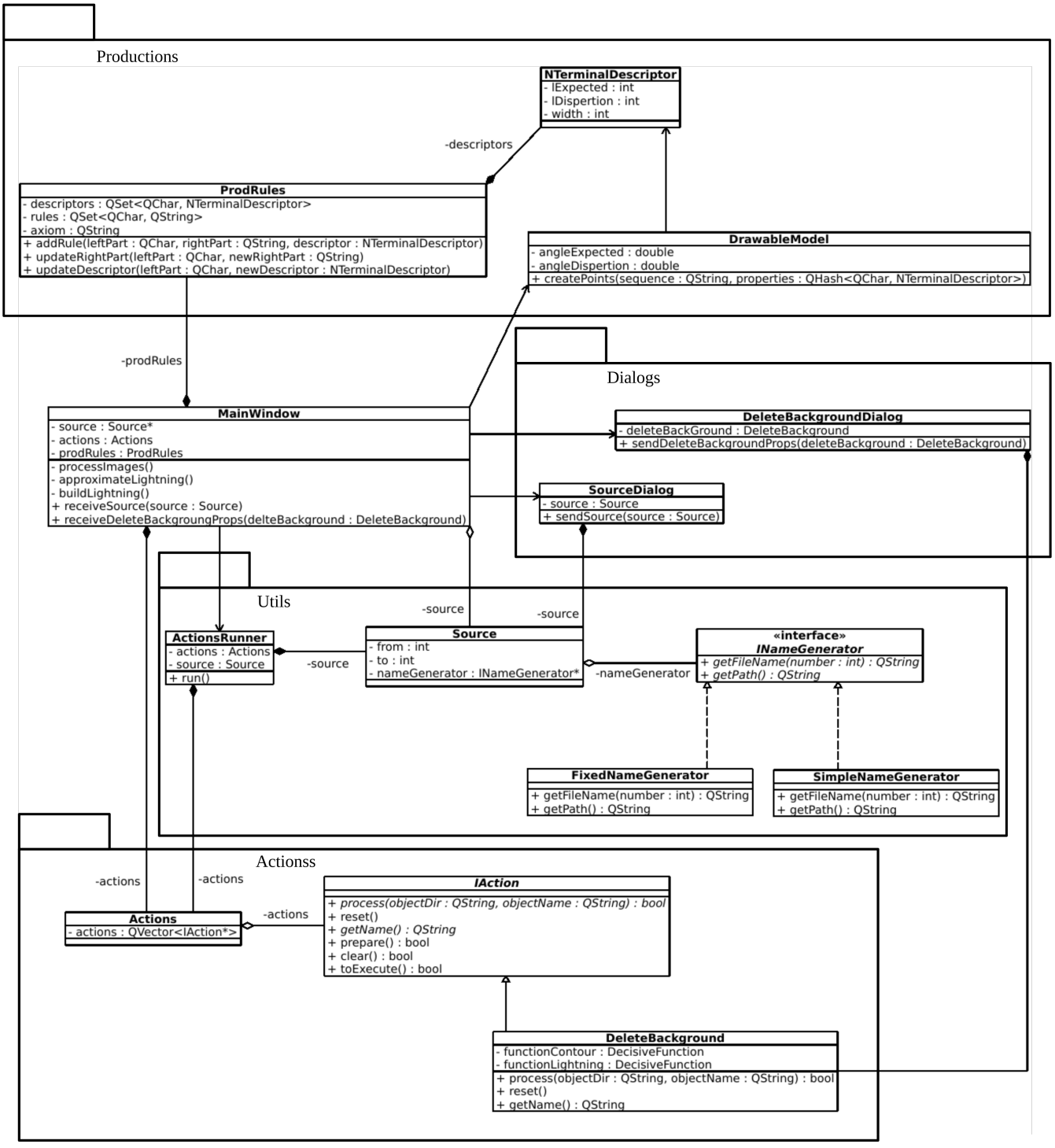

Рисунок 3.5 – Спрощена діаграма класів програми

3.3.4 Використані принципи проектування

Як зазначалось раніше, при розробці даного програмного продукту була використана парадигма об'єктно-орієнтованого програмування. Його принципи дозволили побувати гнучку та легку у розробці систему.

Найбільш вдалим прикладом принципу поліморфізму є метод process() класу IAction. Це дозволяє класу ActionsRunner виконувати необхідну операцію щодо виокремлення блискавок, викликаючи даний метод необхідної реалізації IAction. Необхідна реалізація визначається зовні, тому робота ActionsRunner не залежить від виконуваної операції. Таким чином, для додавання нової операції необхідно буде лише, власне, додати нову дію-спадкоємця IAction, реалізувати клас та визначити яким чином користувач буде заповнювати його поля та в який момент часу буде запускатися обробка (тобто розширити функціонал класів інтерфейсу користувача).

Принцип інкапсуляції був використаний з метою знизити зв'язування класів між собою. Наприклад, клас Actions має приватне поле, що являє собою вектор дій, зміна якого зовнішніми класами досягається через відведені публічні методи. Це дозволить за необхідністю змінити тип контейнера, не змінюючи при цьому код файлів, що взаємодіють з даним класом.

Використання принципу наслідування є очевидним (рис. 3.5) і не потребує особливого загострення на собі.

Окрім базових принципів ООП для досягнення легко масштабованої програма, структура якої є чіткою та зрозумілою були використанні SOLID-принципи. Очевидно, що кожен клас має лише одну задачу, а тому програма реалізує принци єдиної відповідальності (SRP).

Подальше масштабування програмами можливе лише за умови дотримання принципу відкритості-закритості (OCP), що означає, що функціонал класу можна розширювати, але ні в якому разі не змінювати, оскільки це може привезти до необхідності змінювати реалізації класів, що використовують даний і, як наслідок, до помилок, що буде складно виявити та усунути.

До того ж об’єкти в програмі можуть бути замінені їх нащадками без зміни коду програми (LSP – принцип підставновки Лісков); клієнти потрапляють в залежність від методів, якими вони не користуються (ISP – принцип відділення інтерфейсу), а модулі високого рівня не залежать від модулів низького (DIP – принцип інверсії залежностей).

3.3.5 Використані шаблони проектування

Шаблон *фабричний метод*, або віртуальний конструктор використовується для породження нащадків класу IAction. Даний метод приймає код дії та створює необхідну в залежності від значення вхідного параметру, що дозволяє використовувати в коді програми не специфічні класи, а маніпулювати абстрактними об'єктами на більш високому рівні. Реалізує принцип відкритості-закритості та спрощує додавання нових операцій у програму.

Шаблон Ітератор використовується для застосування всіх визначених дій до вхідного набору зображень.

Патерн Шаблонний метод пропонує розбити алгоритм на послідовність кроків, описати ці кроки в окремих методах і викликати їх в одному шаблонному методі один за одним. Це дозволить підкласам перевизначити деякі кроки алгоритму, залишаючи без змін його структуру та інші кроки, які для цього підкласу не є важливими [56].

Використана технологічна платформа дозволила дещо спростити процес реалізації складних операцій.

Механізм збереження налаштувань користувача реалізує стандартний клас *QSettings*. Даний інструментарій призначено для роботи, що потребує значної кількості експериментів. Користувачеві буде зручно змінювати лише деякі налаштування від сеансу до сеансу. Це зекономить його час та спростить використання програми. Інформація щодо налаштувань зберігається в системному реєстрі в Windows і в файлах списків властивостей в macOS і iOS. У системах Unix використовують текстові файли INI. Таким чином Qsetting – це абстракція навколо цих технологій, що дозволяє зберігати і відновлювати параметри. API QSettings заснований на QVariant, що дозволяє зберігати більшість типів.

Клас *QThread* надає незалежний від платформи спосіб управління потоками. Об'єкт QThread управляє одним потоком управління в програмі. Усі основні операції виконуються у паралельному потоці. Для цього реалізований клас ActionRunner.

3.4 Розробка інтерфейсу користувача

3.4.1 Проектування інтерфейсу користувача

При проектуванні програмного засобу «Автоматизована система конструктивно-продукційного моделювання молнієвої активності» враховувались наступні принципи побудови [57]:

- інтерфейс має бути інтуїтивно зрозумілим для користувача;
- термінологія, що використовується у програмному продукті має відповідати загальновживаним нормам;
- умовні позначки мають використовуватись у загальновживаному сенсі;
- порядок розміщення елементів повинен буди логічним та послідовним;
- поля введення інформації повинні мати стислі підписи, що пояснюють значення даного поля;
- кількість елементів на формі обмежена, за потрібності необхідно створювати нові форми та діалогові вікна;
- не допускається використовувати складні та незрозумілі конструкції.

До того ж варто передбачити систему обробки помилок. При діях користувача, коли програма не може коректно приступити до виконання операції або сталася помилка при роботі, необхідно виводити попередження або повідомлення про помилку. Текст повідомлення повинен бути стислим та містити інформацію у зрозумілому для користувача вигляді. Необхідно інформувати користувача про те, якими шляхами він може вирішити дану проблему.

При виконанні довготривалих операцій, таких як, процедура виокремлення блискавок, необхідно виводити повідомлення про те, що програма працює у штатному режимі (наприклад, яка дія зараз виконується) та скільки часу залишилось до завершення роботи на спеціальній шкалі.

Важливий принцип відновлюваності системи, оскільки користувачі завжди допускають помилки. При правильному проектуванні інтерфейсу значно знижується кількість помилок користувача (наприклад, використанні меню дозволяє уникнути помилок, які виникають при введенні команд з клавіатури). У інтерфейсах повинні

бути засоби, які дозволяють уникнути помилок через неуважність та втому користувача, а також коректно відновити стан програми після їх виникнення.

Не допускається виконання деструктивних дій без їх підтвердження, тобто, якщо користувач вибрав потенційно деструктивну операцію, то необхідно надіслати запит з метою отримання підтвердження такого наміру.

Крім того, оскільки характеристики моніторів користувачів можуть відрізнятися, в інтерфейсі повинні бути засоби, які допомогли б користувачам настроїти інтерфейс для зручного користування. Це можуть бути засоби, що дозволяють відображати збільшений текст та можливість вільно змінювати розмір робочого вікна програми.

На етапі проектування до остаточного затвердження інтерфейсу користувача зручно оперувати ескізами екранів. При розробці ескізів багато уваги приділялося внутрішньо-програмній узгодженості інтерфейсу. Назви кнопок, розташування ключових елементів форм і загальний стиль по можливості уніфікувалися або робилися схожими.

Спроектований інтерфейс складається з головного вікна (рис. 3.6).

Вікно складається з наступних елементів:

– панелі для демонстрації зображення;
– стрічки стану виконання операції;
– панелі вкладок з наступними вкладками: підготовка, обробка та модель;
– кнопки виконання задачі з параметрами на активній вкладці.

В залежності від виконуваної задачі, необхідно змінити вкладку на панелі.

На кожній панелі, в залежності від поставленої задачі, розміщуються параметри які необхідно ввести до системи задля виконання задачі, так для задачі «Підготовка» необхідна наявність таких елементів:

– елементів вибору вхідних зображень;
– елементів вибору вихідного шляху;
– списку операцій з можливістю налаштування їх параметрів, включення або виключення із результатного набору.

Для задачі «Обробка» необхідна наявність таких елементів:

– елементів вибору вхідного шляху до файлу;
– елементів вибору вихідного шляху для зображень;
– параметри нанесення блискавок на зображення.

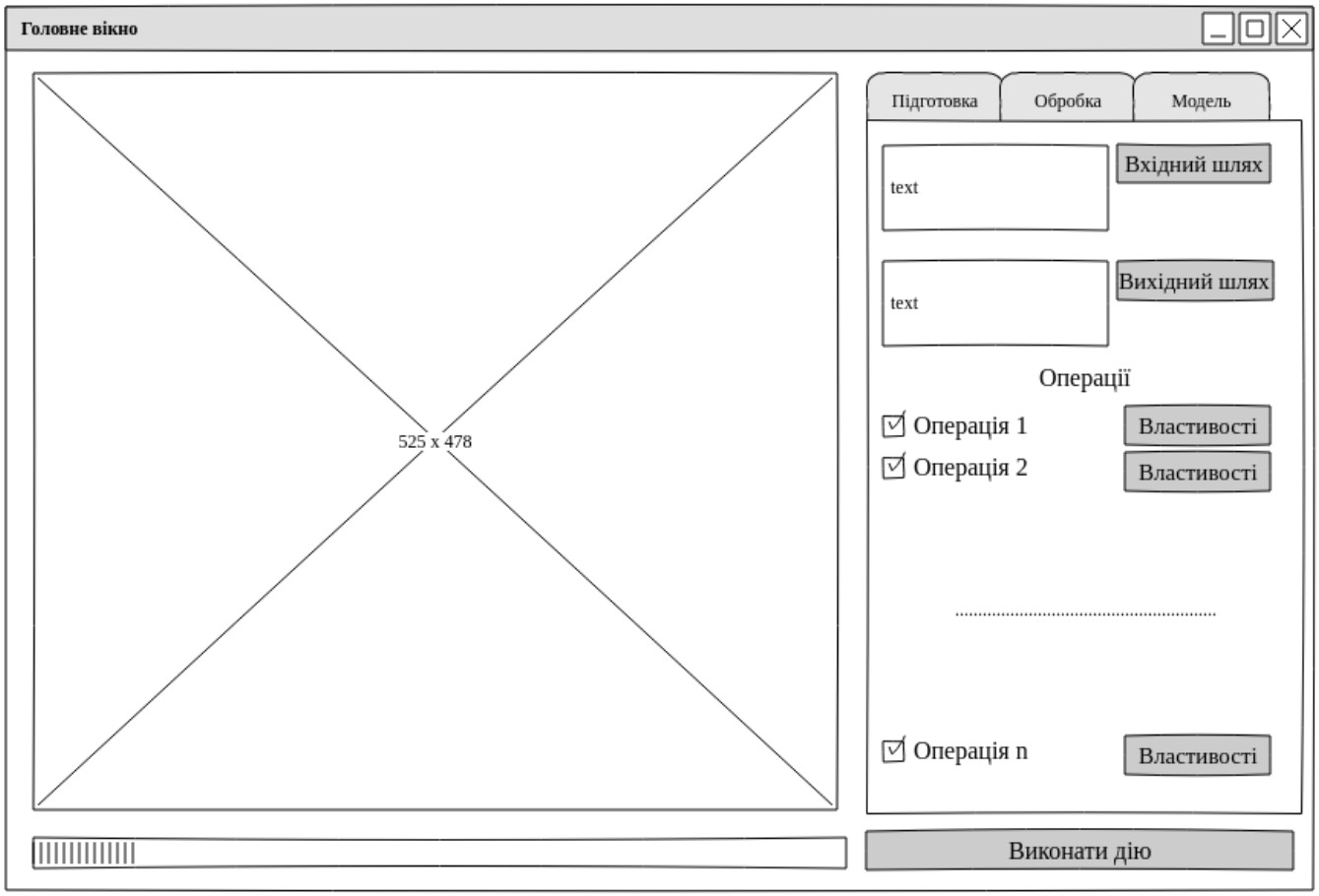

Рисунок 3.6 – Ескіз головного вікна програми

Додатково передбачаються діалоги вибору властивостей та параметрів. Шлях до файлу – це стандартне діалогове вікно вибору файлу. Вибір кольору – це системний діалог визначення кольору.

Для визначення вхідного шляху для зображення використовується діалог, ескіз якого представлено на рис. 3.7.

Для вибору каналу кольору, який необхідно видалити використовується наступний діалог (рис. 3.8).

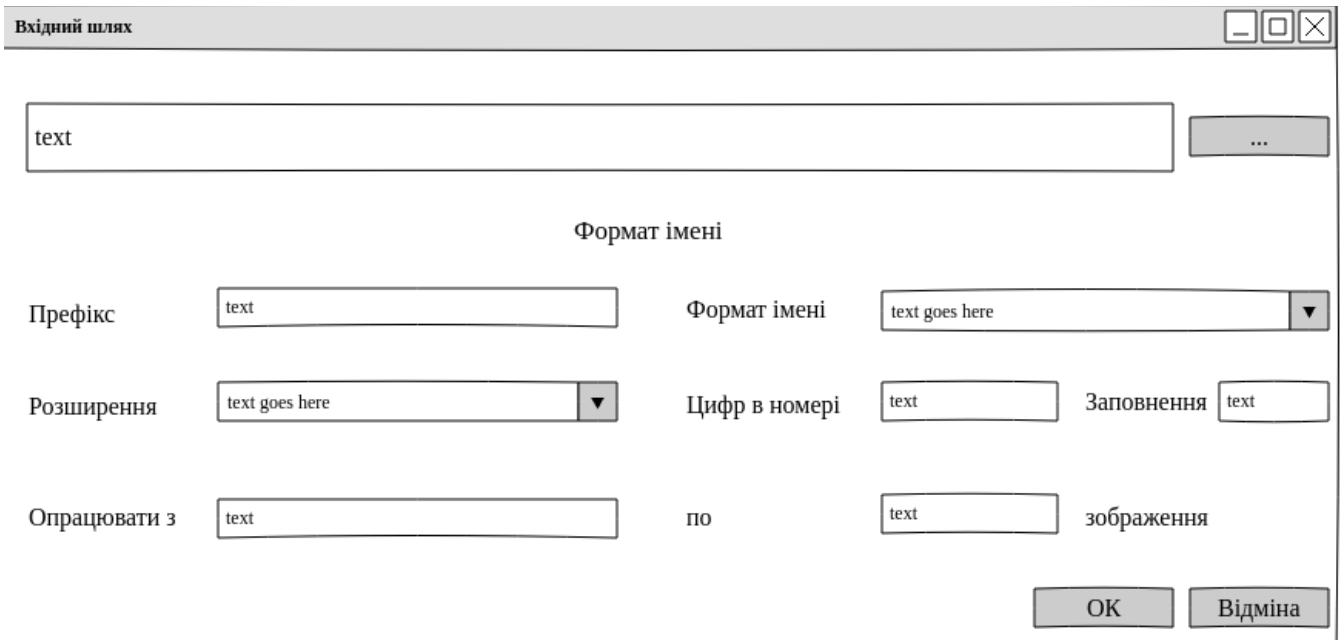

Рисунок 3.7 – Ескіз вікна вибору шляху до зображень

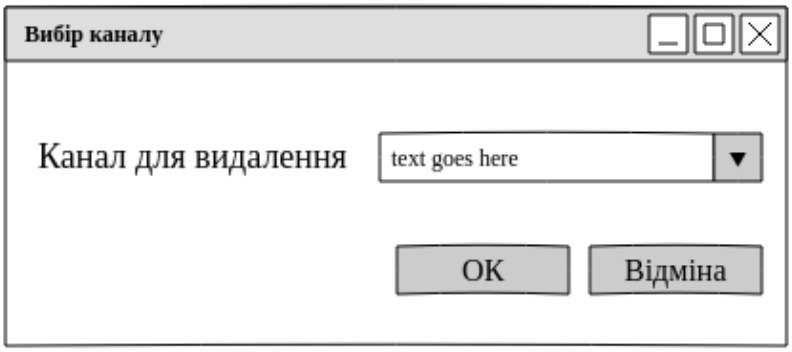

Рисунок 3.8 – Ескіз вікна вибору каналу для видалення

Для заповнення параметрів видалення кольору передбачається використовувати вікно (рис. 3.9), на якому розміщені елементи для вибору кольору фону та етапи обробки зображення з вибором їх параметрів.

Задля зручного налаштування параметрів кольору передбачене вікно, де розміщена панель з двома вкладками, де на одній розміщені параметри моделі Lab, а на другій – LCH (рис. 3.10). Є можливість налаштувати інтервал для кожної координати окремо, до того ж значення L-координати дублюються.

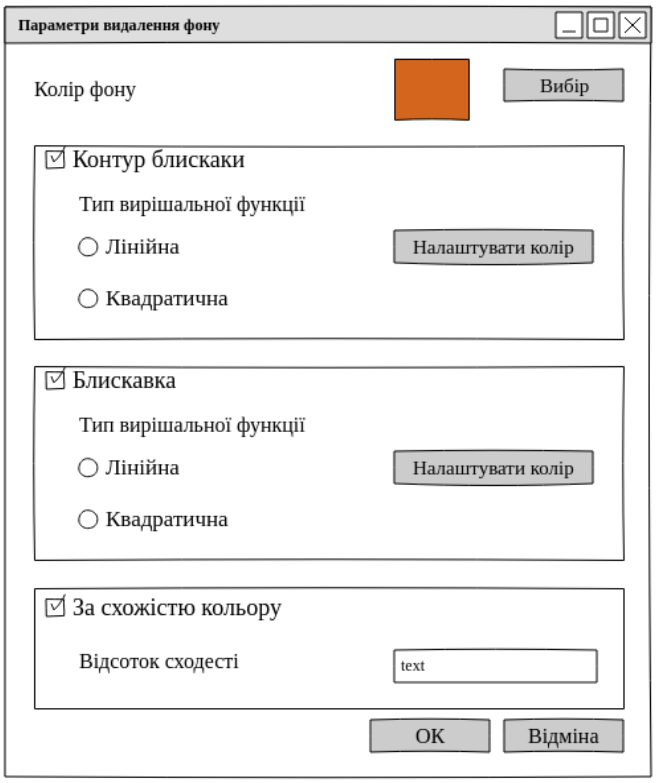

Рисунок 3.9 – Ескіз діалогу вибору параметрів видалення фону

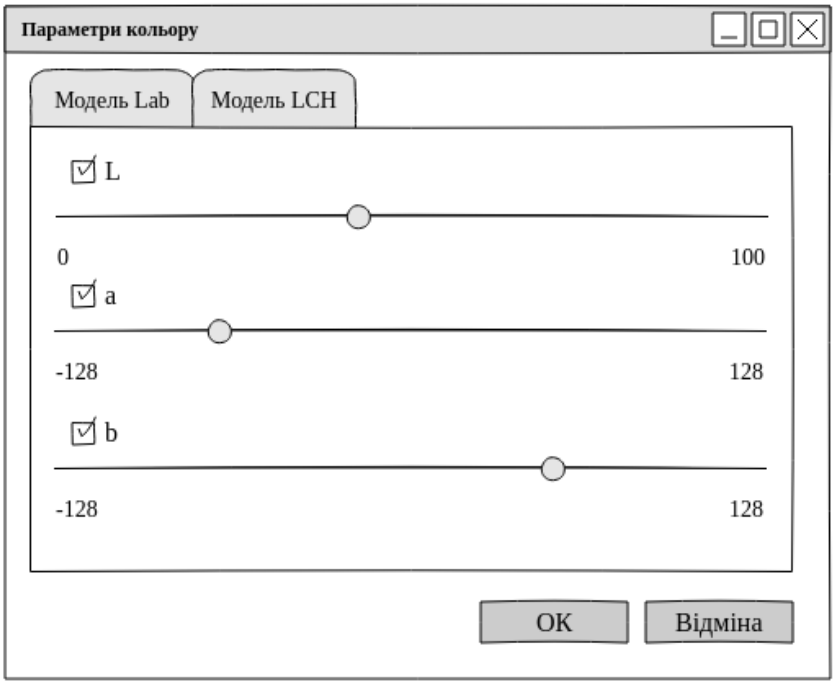

Рисунок 3.10 – Ескіз діалогу вибору колірних інтервалів

3.4.2 Реалізація інтерфейсу користувача

Після етапу проектування був створений інтерфейс користувача, який відповідає вимогам, що представлені та описані раніше. Завдяки перевагам обраної платформи розробки елементи вікон автоматично приймають вигляд відповідний до стилю налаштувань операційної системи користувача.

Ще однією особливістю розробки, було використання елементу *Grid Layout* з подальшою прив'язкою його до головного вікна, що дозволяє легко змінювати його розмір та адаптувати до розміру екрану та до потреб користувача.

При реалізації інтерфейсу та його подальшому використанні були виявлені деякі незручності та неточності при проектуванні інтерфейсу. Ці недоліки буди усунені, таким чином інтерфейс користувача був протестований та дороблений відповідно до вимоги користувачів. Наприклад, вибір кольору фону був перенесений на головне вікно програми (рис. 3.11), яке показується користувачеві після запуску розробленого додатку. Вкладка модель показано так, як показано

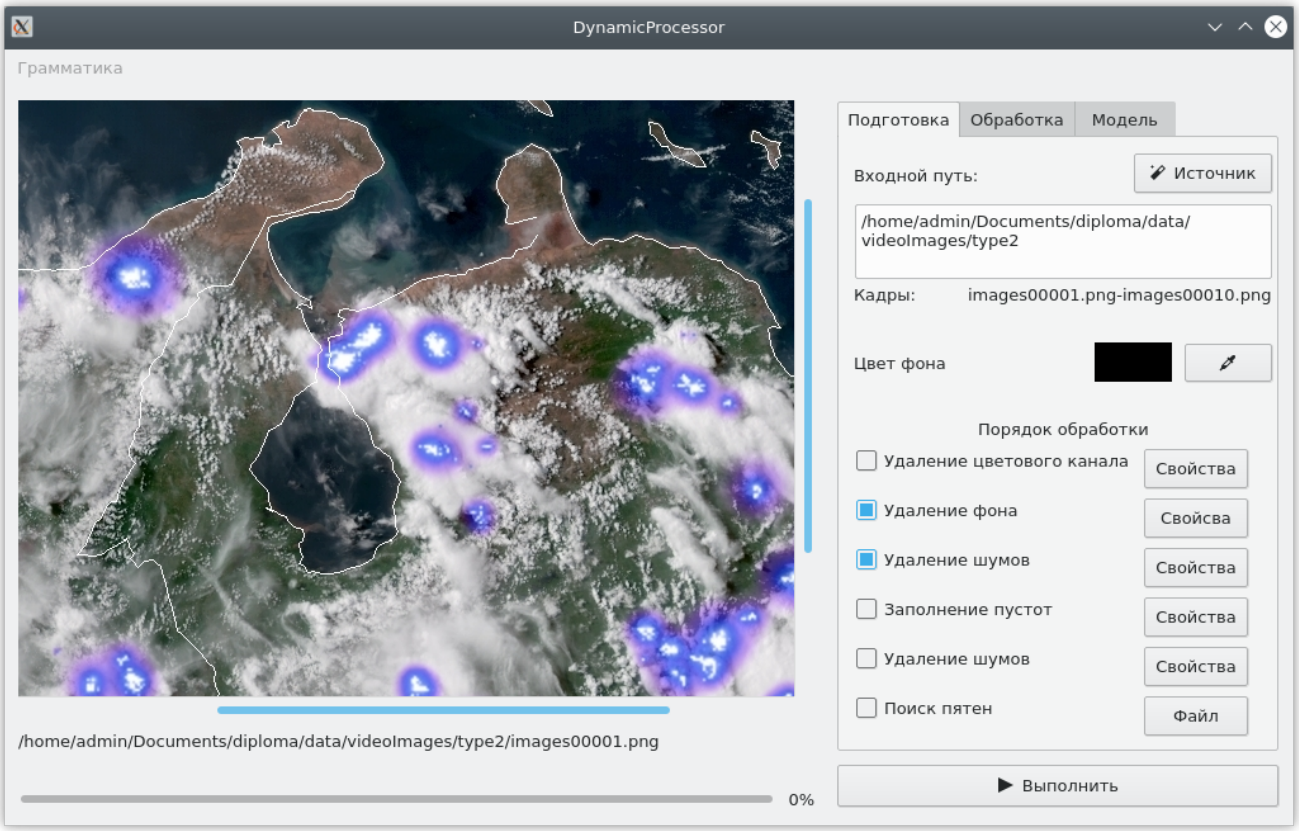

Рисунок 3.11 – Головне вікно після вибору параметрів джерела

На рис. 3.12 представлено інтерфейс програми при переході на вкладку модель та візуалізацій правил продукції.

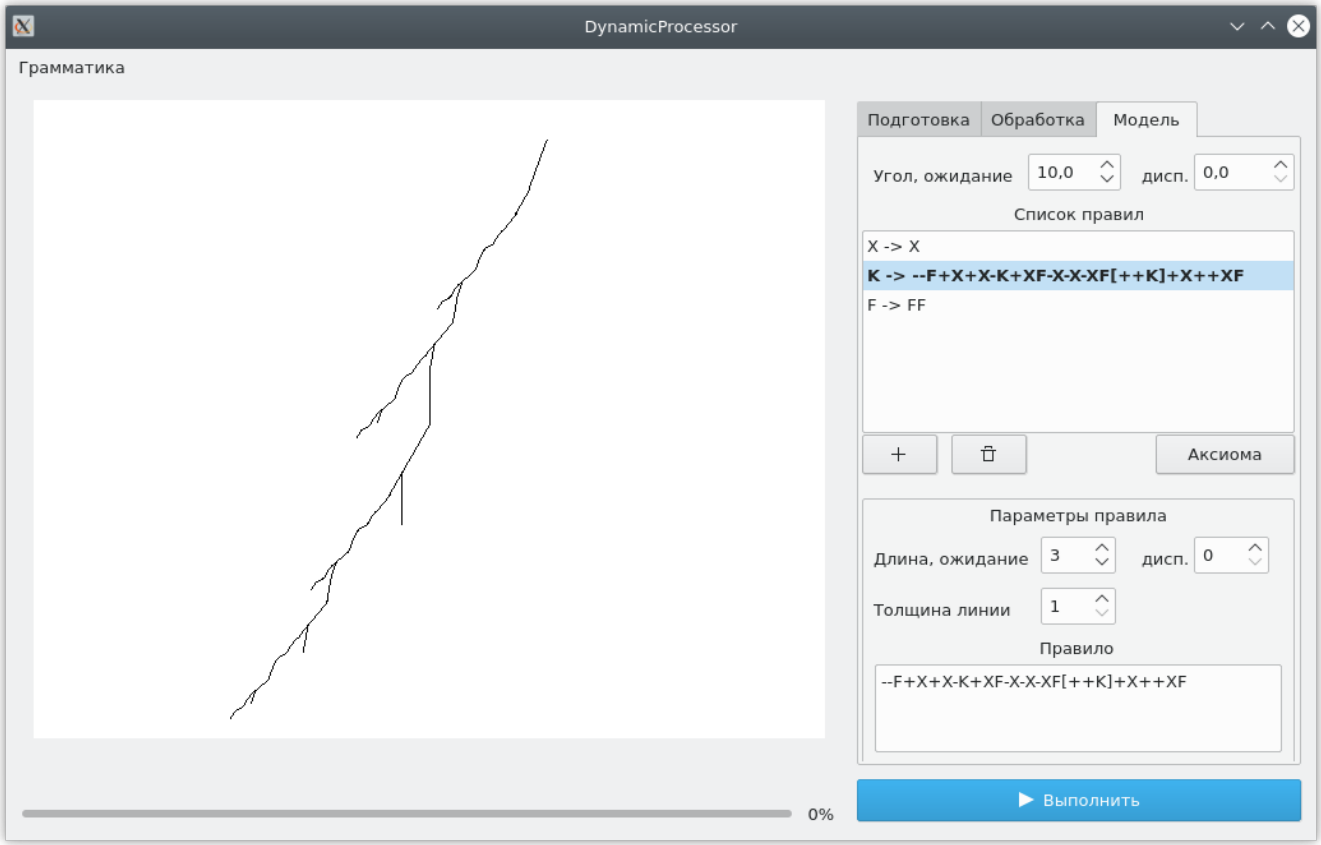

Рисунок 3.12 – Графічне представлення правил продукції

3.5 Тестування та налагодження програми

Оскільки програма являє собою складну систему, то якості обраного методу тестування був обраний метод еквівалентних розбиттів («чорний ящик»). Процес тестування по методу еквівалентного розбиття включає в себе ідентифікацію набору даних як умов введення, які дають однаковий результат при виконанні програми і класифікують їх як набір еквівалентних даних (оскільки вони змушують програму поводитися однаково і генерувати той же результат) і відбувається розбиття їх на групи з іншого еквівалентного набору даних.

3.5.1 Перевірка правильності виокремлення блискавок

Дане тестування направлено на перевірку того, що алгоритм програми щодо виокремлення блискавок працює очікуваним чином, перевірка адекватності даного

алгоритму не перевіряється оскільки це залежить від вхідних параметрів, для налаштування яких і, власне, створювалась програма.

Для перевірки того, що програма може виокремлювати об'єкти за поточним та попереднім кадром створимо 3 пари зображень у відтінках сірого, що розбиті на такі класи: карди, що демонструють процес розвитку блискавки з яскравим центром та неоднорідним забарвленням фону (рис. 3.13, 1а,б), кадри повністю однакові (рис. 3.13, 2а,б), кадри складаються з пікселів, де кожна складова кожного відрізняється на іншої на 25% (рис. 3.13, 3а,б).

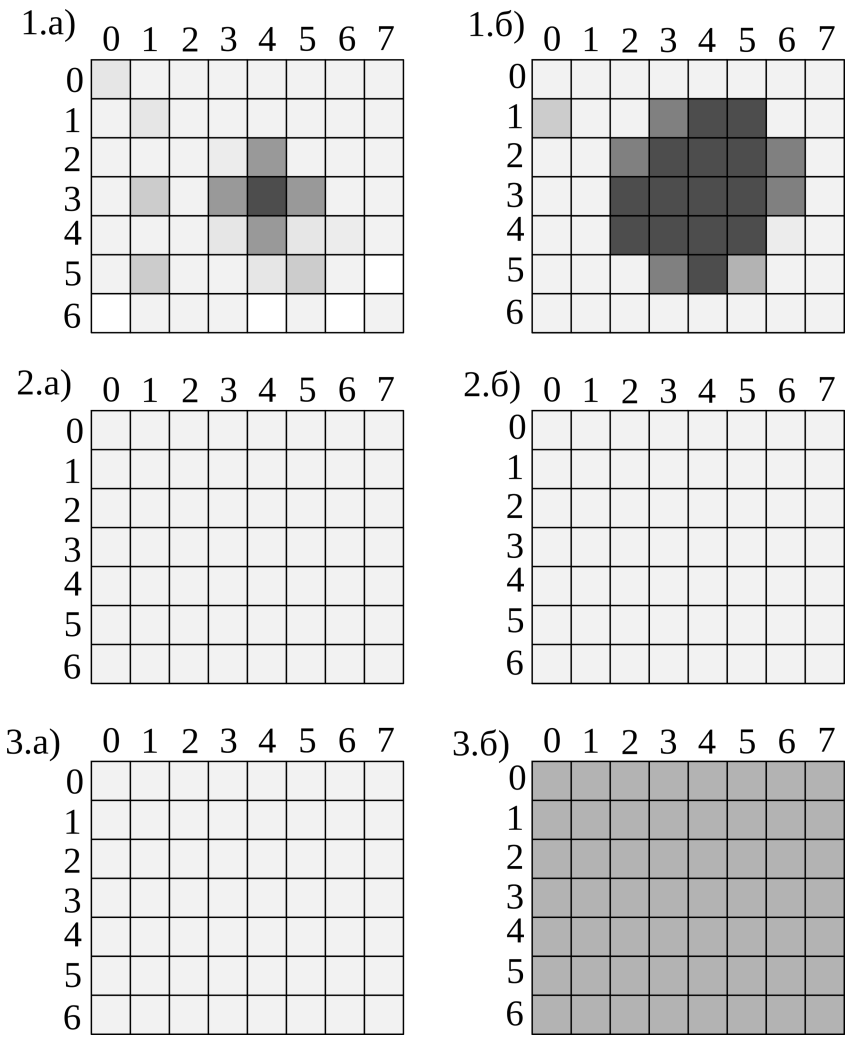

Рисунок 3.13 – Вхідні дані для тестування правильності виокремлення блискавок (статика)

Створимо зображення з очікуваним результатом за рівнями подібності: 0 (рис. 3.14, а), 20 (рис. 3.14, б) та 80 (рис. 3.14, в).

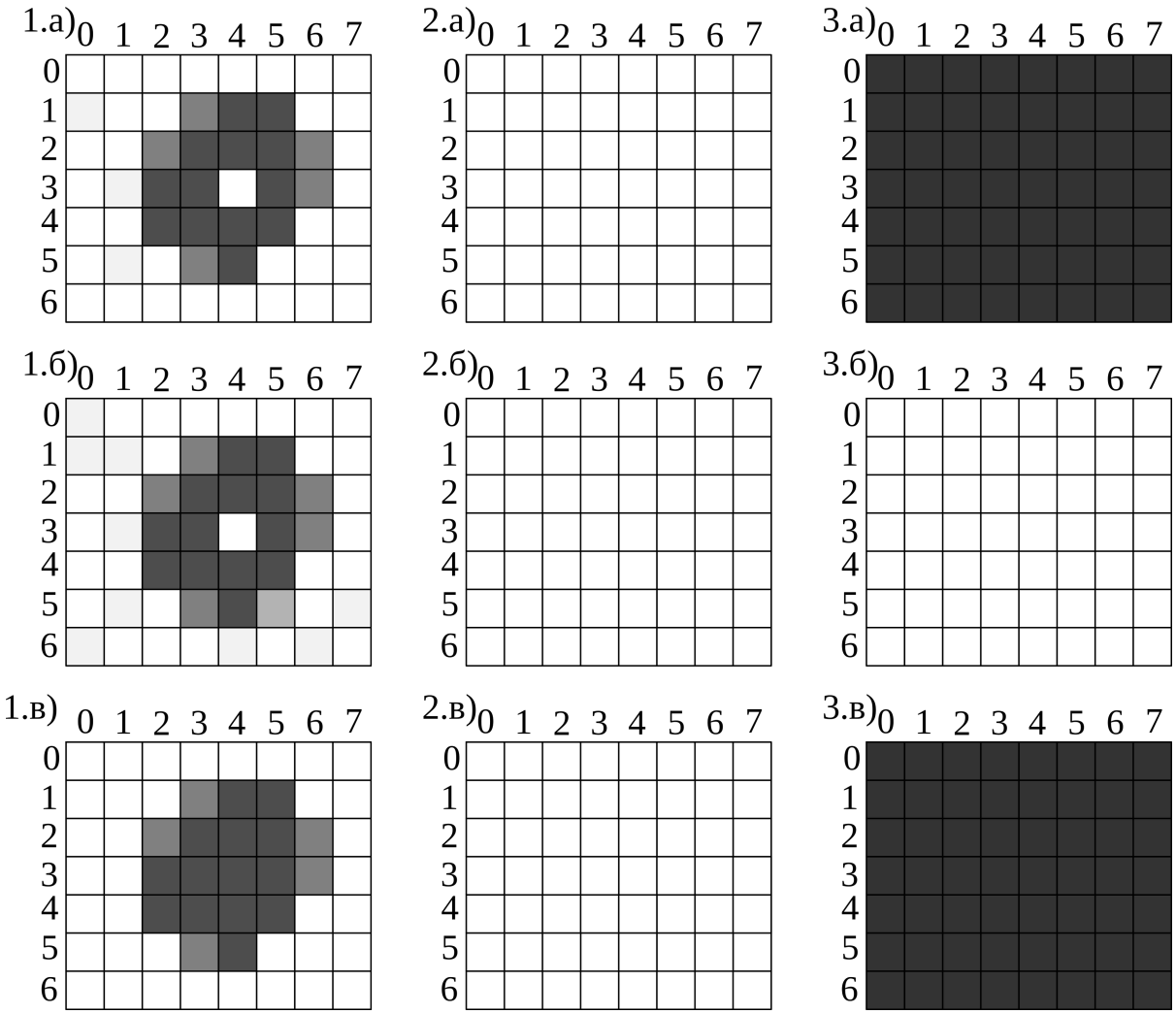

Рисунок 3.14 – Очікувані результати для наборі тестування виокремлення блискавок статика за різними пороговими рівнями

Програма видала результати, що при візуальному порівнянні співпадають з очікуваними.

3.5.2 Перевірка правильності роботи програми при видалені шумів

Була підготовлена серія із 3 тестових кадрів розміру 7*6 пікселів для простоти тестування, яка складалась з вхідного зображення та очікуваного результату. Розбиття на класи відбувалось за наступним принципом: зображення близьке до реальних умов, що в свою чергу містить кутові пікселі (рис. 3.15, 1.а), зображення без шумів (з плямою) (рис. 3.15, 2.а) та порожнє зображення (рис. 3.15, 3.а). Перевірка результатів відбувалась на основі візуального порівняння з вхідним.

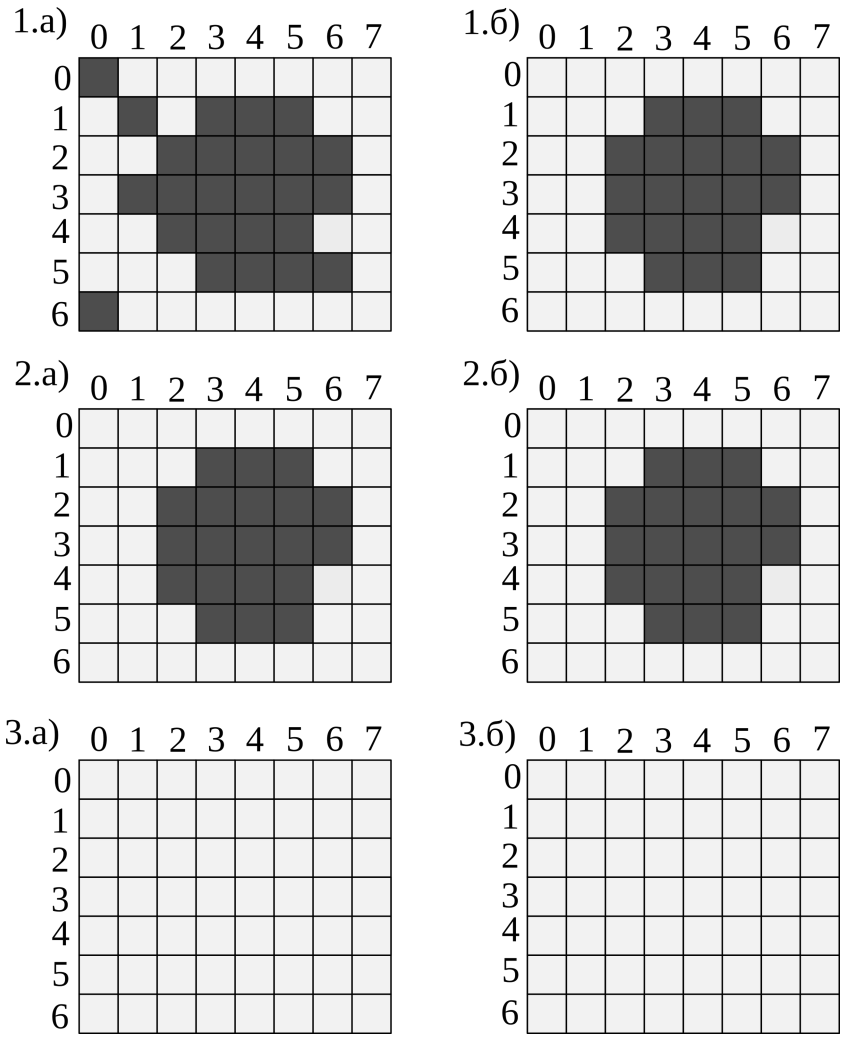

Рисунок 3.15 – Данні для тестування видалення шумів (а – вхід, б – вихід, що очікується)

3.5.3 Перевірка правильності розрахунку координат центру блискавки та її радіусу

Для вирішення даної задачі буди підготовлені серія з 4 кадрів за наступним принципом: набір пікселів у вигляді кола (рис. 3.16, а), набір пікселів у вигляді витягнутої стрічки (рис. 3.16, б), порожній кадр (рис. 3.16, в) та кадр повністю заповнений кольором (рис. 3.16, г).

Координати центру плями та її радіус розраховані за описаними раніше формулами вручну. В результаті очікуються текстовий файл з даними, що представлені у таблиці 3.1.

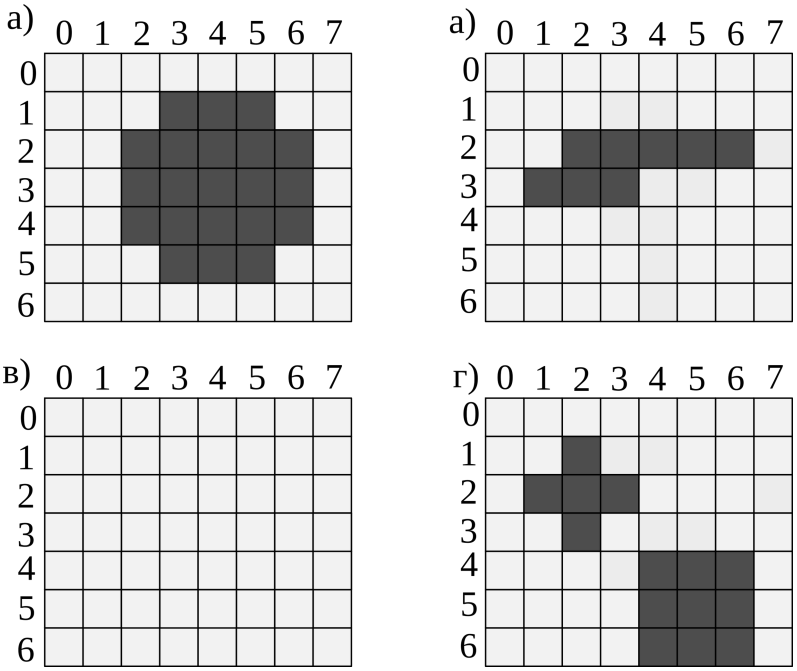

Рисунок 3.16 – Вхідні дані для перевірки правильності розрахунку координат центру блискавки та її радіусу

Таблиця 3.1 – Розраховані дані для тестування правильності пошуку плям

Рядок файлу	Номер кадру	x центру	y центру	Радіус
1	1	4	3	2
2	2	3	2	1
3	4	2	2	1
4	4	5	5	1

3.5.4 Перевірка правильності побудови моделі за правилами продукції

Через складність відтворення моделі за правилами продукції скористаємось готовими моделями [58] в якості вхідних даних, а перевірку виконаємо візуально: чи подібний результат отриманий за правилами до того, що наведений у джерелі.

У [58] представлені приклади рослиноподібних структур, утворених правилами L-системи з наступними реалізаціями:

– $\langle C_{MS}(X,\{X\to F[+X]F[-X]+X;\ F\to FF\},7),C_A\rangle_{R^1}\ \Omega_1(C_{MS})$,

$\langle C_{GF}(\Omega_1,20,l\leftarrow F=3,h\leftarrow F=1,l\leftarrow X=3,h\leftarrow X=1),C_B\rangle_{R^1}\ \Omega_2(C_{GF})$ (очікувана реалізація представлена на рис. 3.17, а, результат програми – рис. 3.18, а);

– $\langle C_{MS}(X,\{X\to F[+X][-X]FX;\ F\to FF\},7),C_A\rangle_R \Omega_3(C_{MS})$,

$\langle C_{GF}(\Omega_3,25.7,l\leftarrow F=3,h\leftarrow F=1,l\leftarrow X=3,h\leftarrow X=1),C_B\rangle_R \Omega_4(C_{GF})$

(очікувана реалізація – на рис. 3.17, б, результат програми – рис. 3.18, б);

– $\langle C_{MS}(X,\{X\to F-[[X]+X]+F[+FX]-X;\ F\to FF\},5),C_A\rangle_R \Omega_5(C_{MS})$,

$\langle C_{GF}(\Omega_3,22.5,l\leftarrow F=7,h\leftarrow F=1,l\leftarrow X=7,h\leftarrow X=1),C_B\rangle_R \Omega_4(C_{GF})$

(очікувана реалізація – на рис. 3.17, в, результат програми – рис. 3.18, в).

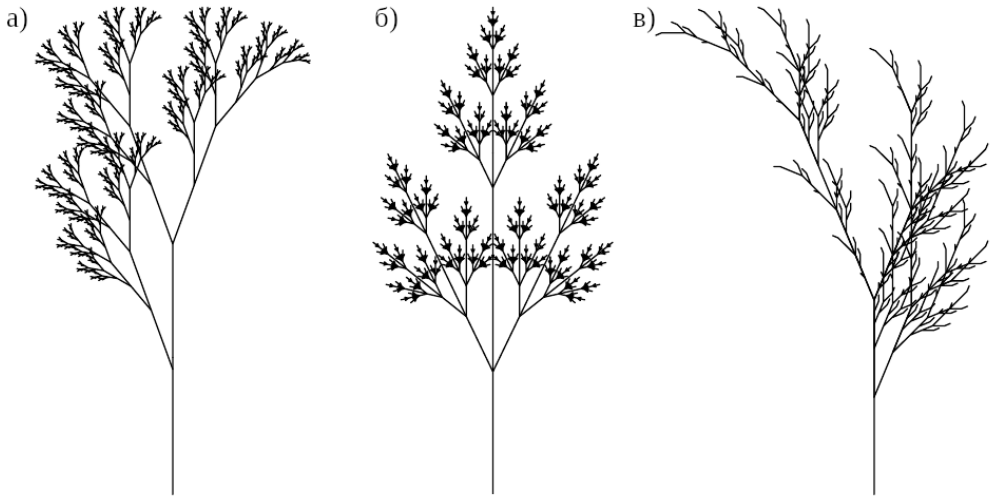

Рисунок 3.17 – Очікуваний результат моделювання для заданих правил конструкції

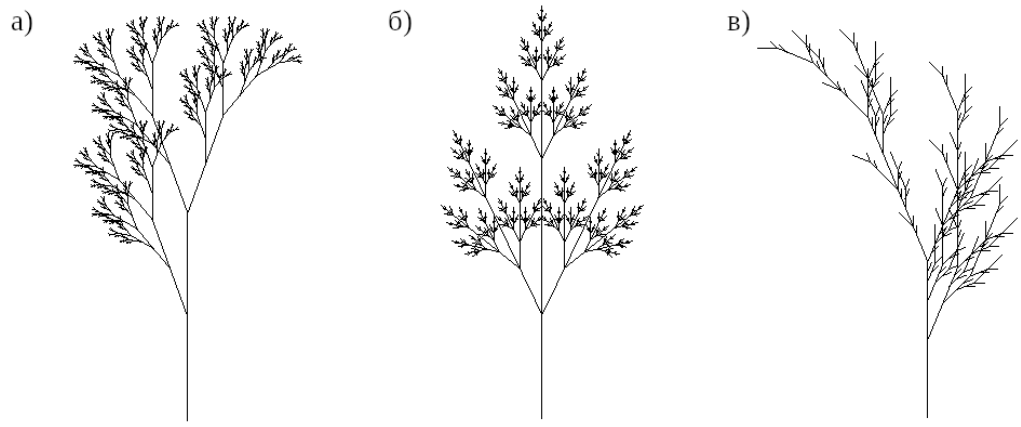

Рисунок 3.18 – Результат роботи програми моделювання для заданих правил конструкції

В результаті візуального порівняння очікуваного результату з отриманим був зроблений висновок, що побудована система коректно відтворює сконструйовані правила продукції.

Висновки до розділу 3

В межах дипломної роботи було виконане проектування та розробка програмного забезпечення для конструктивно-продукційного моделювання молнієвої активності. Дане програмне забезпечення логічно розділене на дві частини: виокремлення блискавок та моделювання блискавок типу хмара-земля. Розроблена архітектура системи відповідає принципам об’єктно-орієнтованого проектування та передбачає можливість подальшого розвитку шляхом додавання алгоритмів для виокремлення блискавок з нових типів відео, що демонструють грози та шляхом моделювання блискавки у тривимірному просторі.

Інтерфейс користувача спроектований таким чином, щоб робота з програмою була максимально простою.

Також було виконано тестування програми та встановлено, що всі функції програми працюють коректно.

Розроблений програмний продукт відповідає усім сучасним вимогам до прикладного програмного забезпечення та готовий для впровадження та використання.

4 ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНИХ АЛГОРИТМІВ ВИОКРЕМЛЕННЯ БЛИСКАВОК ТА ПІДХОДУ ЇХ МОДЕЛЮВАННЯ

4.1 Дослідження ефективності виокремлення блискавок у грозовому фронті

Складність дослідження ефективності виокремлення блискавок програмою у виникає у зв'язку з тим, що немає єдиного правильного варіанту виокремлення блискавок оскільки межа блискавок є нечіткою. Тому результати необхідно перевірити на адекватність – чи співпадає результат з очікуваним.

Нехай на вихідних кадрах існують помилки типу I та II. Помилки типу I виникають, коли програма на місці, де очікується блискавка (або контур), відтворила піксель кольору фону. Помилки типу II являють собою піксель відмінний від кольору фону, на місці, де не очікується блискавка (або контур). Слід зазначити, що помилки обох типів не є критичними в рамках 10% відстані від краю блискавки (або контуру).

Для перевірки ефективності виокремлення блискавок оберемо кадри з відео, що демонструють розповсюдження грозового фронту та виокремимо блискавки вручну. Для виявлення того, чи є результат адекватним використаємо програмний засіб, який приймає на вхід 2 кадри: підготовлений вручну та підготовлений програмою в результаті отримаємо нове зображення, що містить перефарбований кадр підготовлений програмою наступним чином. Пікселі позначені зеленим кольором, якщо вони представляють блискавку як на створеному вручну, так і на відфільтрованому зображенні. Сірі пікселі відображають такий ж випадок для фону зображення. Пікселі, що є блискавки на створеному вручну кадрі, а фоном у відфільтрованому позначено червоним кольором, а протилежний випадок – синім.

4.1.1 Виокремлення блискавок на кадрах зі статичним фоном

Використовуємо кадри [34]. Перше, чого необхідно досягти – якнайкращого результату видалення фону безпосередньо. Для цього використовуємо вирішальну функцію (2.13), (2.14). З'ясуємо, як найефективніше визначати різницю кольорів ΔE, використовуючи методи розрахунку (2.2), (2.3), (2.12) та визначимо параметр tol емпірично. З'ясувалось, що (2.2), (2.3), (2.12) дають позитивні результати при

визначені коректного параметру tol. Однак, визначимо, що (2.2), (2.3) потребують складних довготривалих математичних обчислень, а (2.12) – є простою у реалізації. Найкращі результати досягаються при порівнянні лише червоного та зеленого каналу, тому, спершу, можна видалити синій канал при значенні $tol = 0{,}13$.

Після видалення фону маємо помилки типу I та типу II. Помилки типу I представлені отворами всередині блискавки, які демонстрували яскравим і сильним спалах, що робило кольори пікселів у центральній частині майже однаковими через кілька послідовних кадрів. Проблема вирішується використанням алгоритму заповнення пустот для кожного кадру. Помилки типу II, на відміну від типу I, можуть розглядатися як шуми. Тому необхідно застосувати алгоритм видалення шумів для кожного кадру.

В результаті було встановлено, що найкращий результат для видалення блискавок зі статичним фоном досягається:

1. Видалення синього каналу у просторі RGB;

2. Видалення фону (2.13), (2.14), (2.12) з $tol = 0{,}13$;

3. Застосування алгоритму усунення шумів по 5 разів для кожного з кадрів, що, крім видалення шуму, згладжує форму блискавки;

4. Один раз для кожного кадру виконується алгоритм відновлення видалених частин блискавки;

5. Повторюється видалення шумів (по одному проходу для кожного кадру) для того, щоб згладити краї блискавки після виконання попереднього кроку.

Зображення після видалення фону (рельєфу та хмарності) з використанням вищезазначеного підходу показано у другому рядку, рис. 4.1 для відповідних кадрів вище. Як бачимо, помилок немає, а контури блискавки досить плавні. Після чого усі спалахи були апроксимовані до звичайної форми та відтворені (третій ряд, рис. 4.1).

Отримані результати дозволили перевірити адекватність побудови моделі грозового фронту без його переміщення [1].

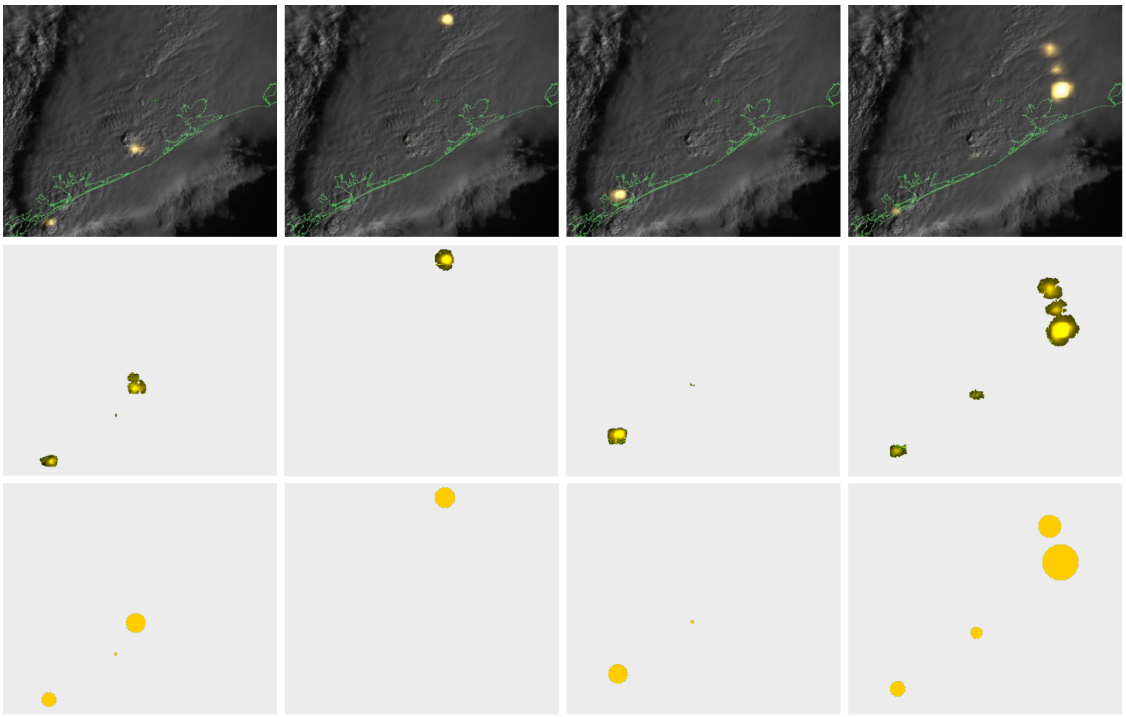

Рисунок 4.1 – Кадри відео, створеного супутниками NASA, грозового фронту зі статичним фоном

4.1.2 Виокремлення блискавок на кадрах з відео, що демонструють динамічну хмарність

Значення, наведені в таблиці 2.1 були підставлені у функції фільтрації, а всі відеозображення [41][42] відфільтровані за допомогою кожного з них. Було сформовано два комплекти зображень, що містять контури блискавок: відфільтрованих за допомогою (2.13), (2.18) та (2.13), (2.19). Для визначення найбільш ефективного фільтру, кадри з обох комплектів були порівняні з виокремленими вручну як було описано вище. На рис. 5 показані результати вилучення контуру для середнього кадру на рис. 2.4 за допомогою лінійного фільтра (2.13), (2.18) на рис. 4.2 а) і квадратичного (2.13), (2.19) на рис. 4.2 б). Як ми бачимо, квадратична функція залишає більше місця в середині контуру, ніж лінійна, що добре для невеликих тьмяних блискавок, оскільки в разі відсутності порожнини в контур блискавка буде втрачена. Також лінійний фільтр залишає фон, який не можна віднести до контуру, що призведе до появи помилок II типу на кадрах з виокремленими блискавками.

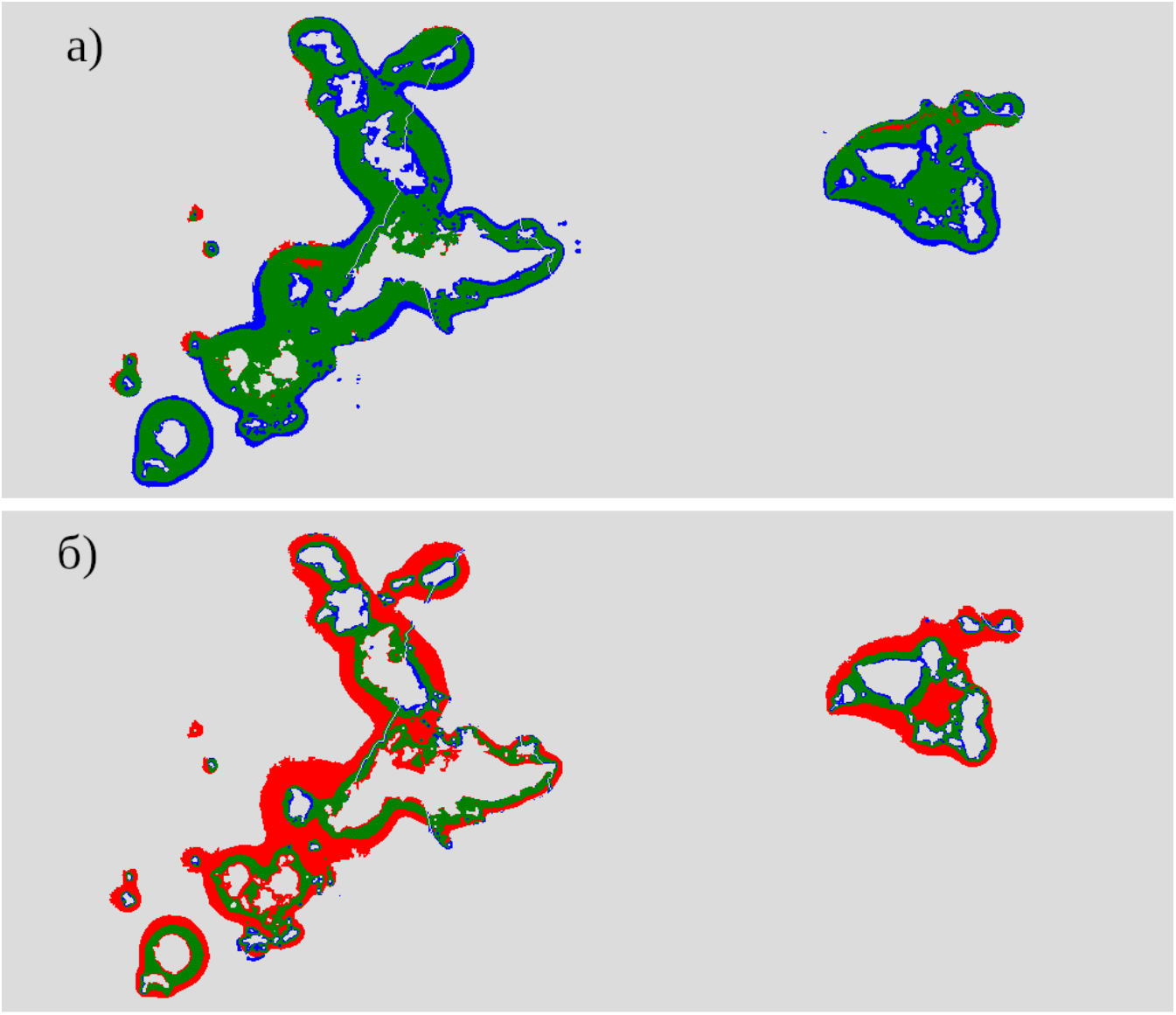

Рисунок 4.2 – Результат виокремлення ореолів блискавок

Зображення з отриманими ореолами були оброблені з метою пошуку блискавки з використанням 2.20, а після – ще раз з використанням квадратичного (2.13), (2.18) та лінійного (2.13), (2.19) фільтру окремо для кожного з комплектів зі значеннями інтервалів, розрахованими у таблиці 2.2.

Серія експериментів показує, що обидва фільтри (2.18) або (2.19) з параметрами, представленими в таблиці 2.2, однаково добре працюють для виявлення блискавки, тому рекомендовано для вирішення даної задачі використовувати фільтр (2.18), оскільки він простіший завдяки своїй лінійності. Кадри порівняння на рис. 4.3, а) та 4.3, б) показують результати виокремлення

блискавки для середнього кадру на рис. 2.4 після виявлення контурів за допомогою лінійних та квадратичних фільтрів відповідно.

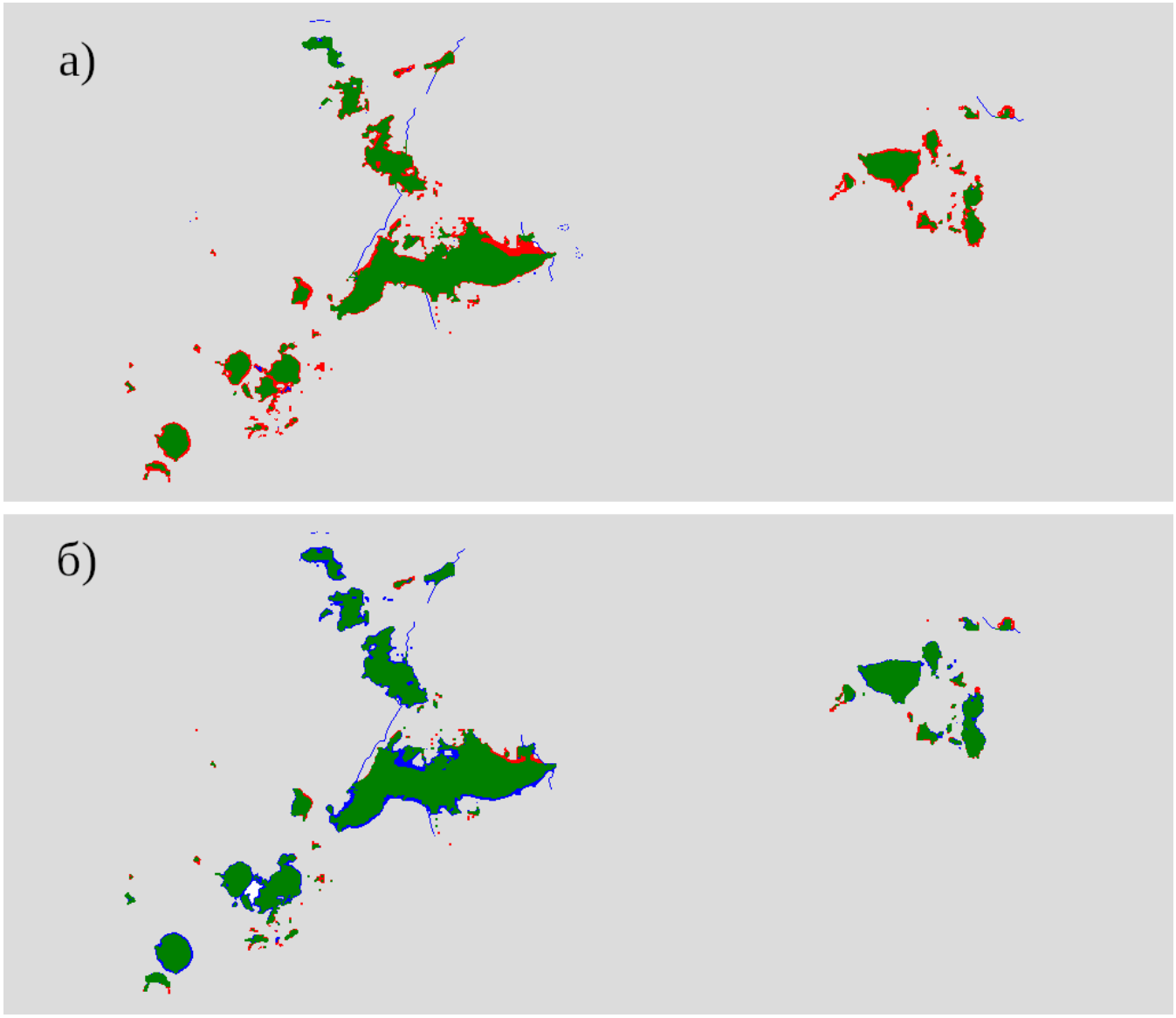

Рисунок 4.3 – Результат виокремлення блискавок з виокремлених контурів лінійним (а) і квадратичним (б) фільтром

Як видно, отримані кадри містять лінії – границі регіонів, що є небажаним. Для вирішення цієї проблеми був використаний розроблений підхід для виявлення блискавки зі статичним фоном, оскільки ці лінії не рухаються від кадру до кадру. Таким чином поєднується кілька підходів з метою виявлення блискавки. Таким чином для виокремлення блискавок з кадрів зі статичним фоном алгоритм складається з наступних кроків:

1. Формування ореолів блискавки з використанням фільтру (2.13), (2.19) зі значеннями інтервалів, що наведені в табл. 2.1.

2. Вибір пікселів підозрілих для блискавок за допомогою (2.20) та перевірка належності пікселя блискавки з використанням фільтру (2.13), (2.18) зі значеннями, розрахованими у табл. 2.2.

3. Перевіряється, чи не належить піксель лінії кордону регіону за допомогою (2.13). (2.14) та $tol = 13$.

Зображення після видалення фону (рельєфу та хмарності) з використанням вищезазначеного підходу показано у другому рядку, рис. 4.1 для відповідних кадрів вище. Як бачимо, помилок немає, а контури блискавки досить плавні. Після чого усі спалахи були апроксимовані до звичайної форми та відтворені (третій ряд, рис. 4.1).

Отримані результати дозволили перевірити адекватність побудови моделі грозового фронту з його переміщенням у просторі з часом [2].

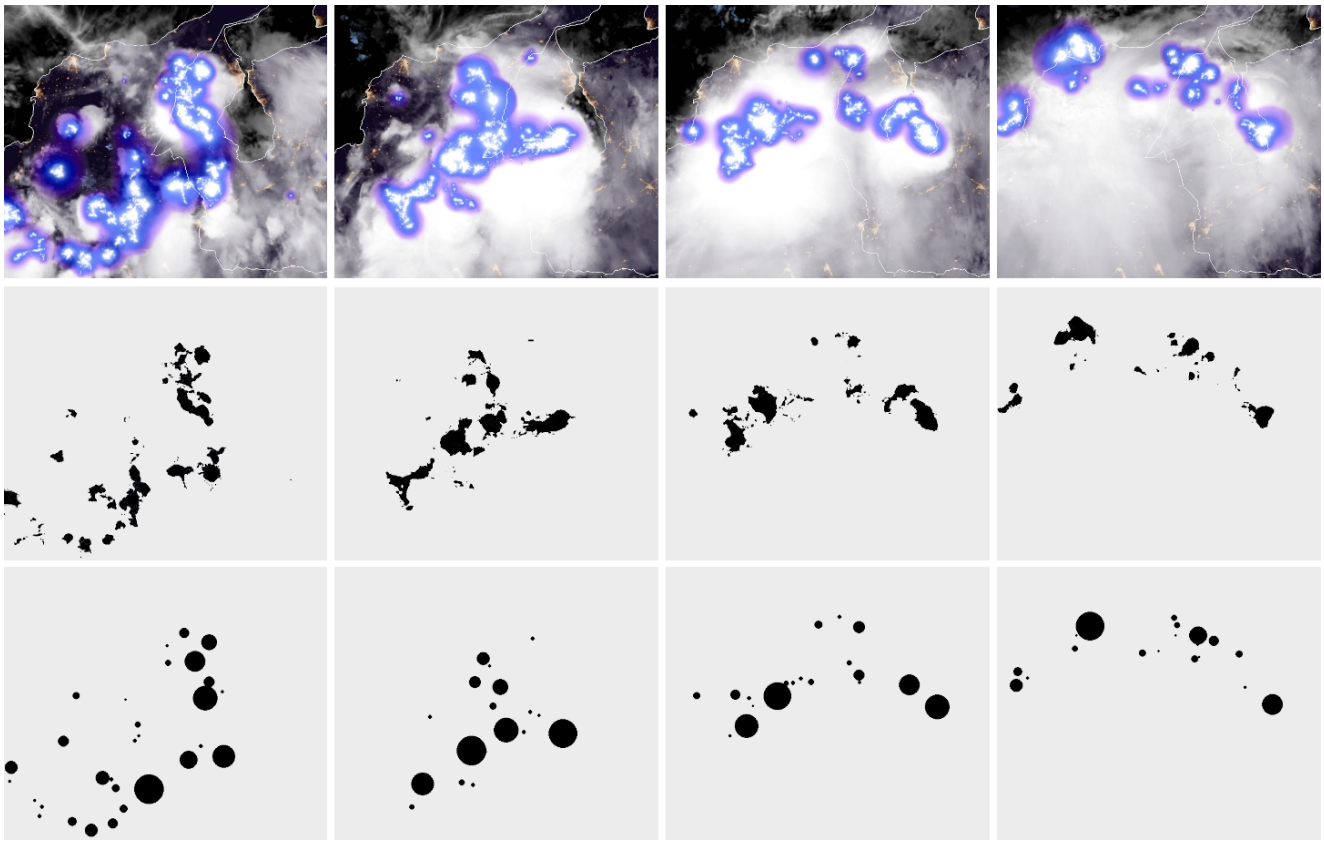

Рисунок 4.4 – Кадри відео, створеного супутниками NASA, грозового фронту з динамічною хмарністю

4.2 Дослідження ефективності використання конструкторів при моделюванні блискавок типу хмара-земля

Дослідження ефективності використання конструкторів при моделюванні блискавок типу хмара земля полягає у визначенні можливості побудови моделі блискавок, що схожа на реальну. Таким чином, спершу, необхідно зібрати матеріал для порівняння. Оскільки грози супроводжуються відсутністю сонячного світла, то фотографії робляться зі значними показниками витримки, що призводить до того, що на фото зафіксована не одна блискавки. Таким чином, матеріал для порівняння – це кадри, що вирізані з відео, які демонструють грози.

Експерименти проводяться наступним чином: обирається кадр, що демонструє поширення блискавки, після чого підбираються правила конструкції, щоб отримати модель блискавки найбільше схожою на її прототип. Порівняння виконуються візуально.

Оберемо прототип, який представлено рис. 4.5, б), та виконаємо конкретизацію та реалізацію багатосимвольних конструкторів з подальшою його графічною реалізацією.

Визначимо значення P_1 для побудови першого варіанту блискавки:

$$P_1 = \{ X \to --F+++++L---FF++F-T+T++F-T+T--FF--T$$
$$F \to T-T-T+T+T$$
$$L \to L--L++L$$
$$T \to TT \}.$$

В результаті реалізації $\langle C_{MS}(X, P, 7), C_A \rangle_{R} \Omega_1(C_{MS})$ отримаємо багатосимвольну конструкцію $\Omega_1(C_{MS})$, яка є одним із параметрів для конкретизації і реалізації графічного конструктора. Іншим його параметром є кут $\alpha = 18^\circ$. На рис. 4.5, а) представлена блискавка, що є реалізацією конструктора $\langle C_{GF}(\Omega_1, 18.0, l \hookleftarrow L = 7, h \hookleftarrow L = 2, l \hookleftarrow T = 2, h \hookleftarrow T = 2), C_B \rangle_{R} \Omega_2(C_{GF})$.

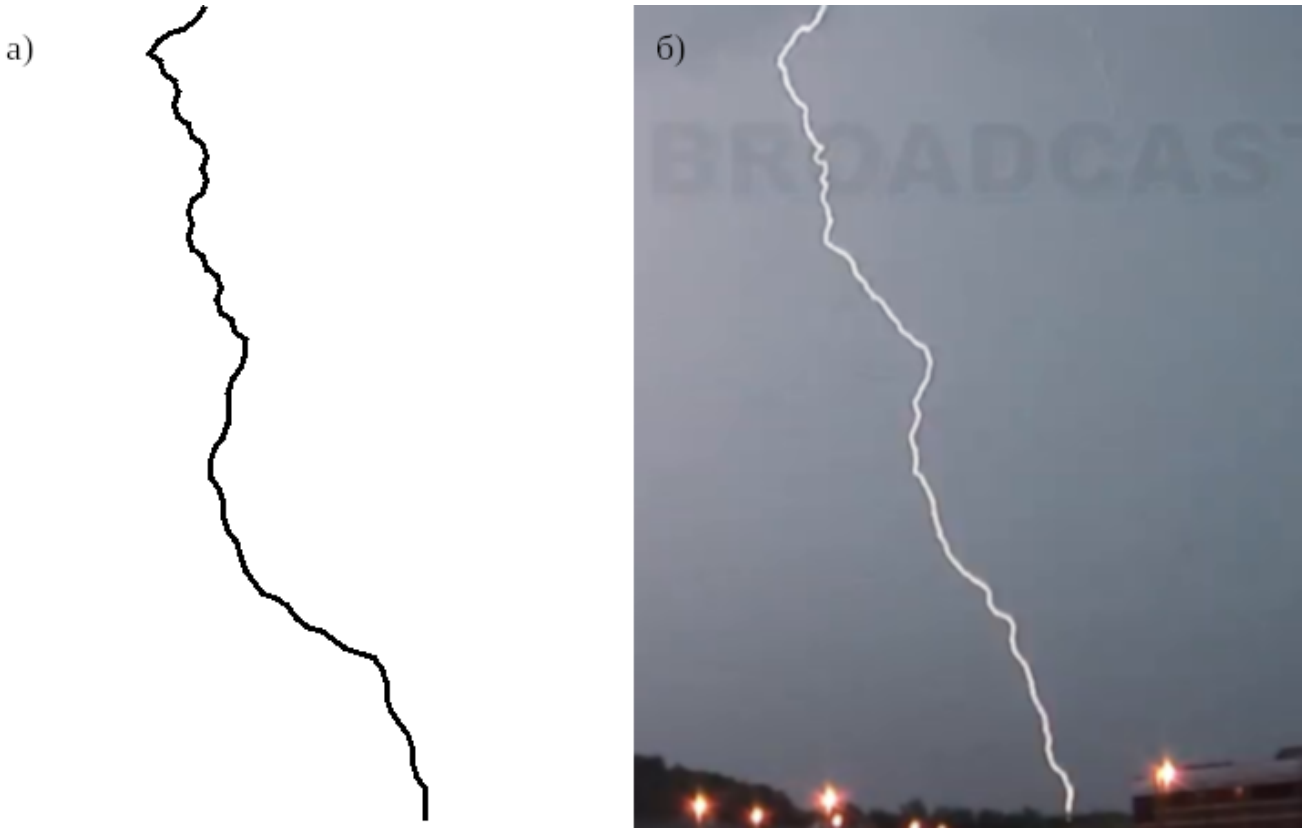

Рисунок 4.5 – Реалізація конструктора Ω_2 – а) та його прототип у природі – б)

Визначимо значення P_2 для побудови другого варіанту блискавки, який представлений на рис. 4.6, б).

$$P_2 = \{ K \to F[+++X] + FF[-B] ++ B$$
$$B \to -X - [+XB][+++XB] ++ X \} \cup P_1.$$

В результаті реалізації $\langle C_{MS}(X, P_2, 5), C_A \rangle_{R^1} \Omega_3(C_{MS})$ отримаємо багатосимвольну конструкцію $\Omega_3(C_{MS})$, яка є одним із параметрів для конкретизації і реалізації графічного конструктора. Іншим його параметром є кут $\alpha = 20^\circ$. На рис. 4.6, а) представлена блискавка, що є реалізацією конструктора:

$$\langle C_{GF}(\Omega_3, 20.0, l \hookleftarrow X = 1, h \hookleftarrow X = 1 l \hookleftarrow L = 2, h \hookleftarrow L = 1, l \hookleftarrow T = 2, h \hookleftarrow T = 1, l \hookleftarrow F = 7, h \hookleftarrow F = 2,$$
$$l \hookleftarrow B = 20, h \hookleftarrow F = 1) C_B \rangle_{R^1} \Omega_4(C_{GF}).$$

Зауважимо, що дана блискавка складається з елементів, які відтворюють блискавку з реалізацією Ω_2. Таким чином, Ω_4 реалізує розгалужену структури блискавки, що представлена рис. 4.5, а).

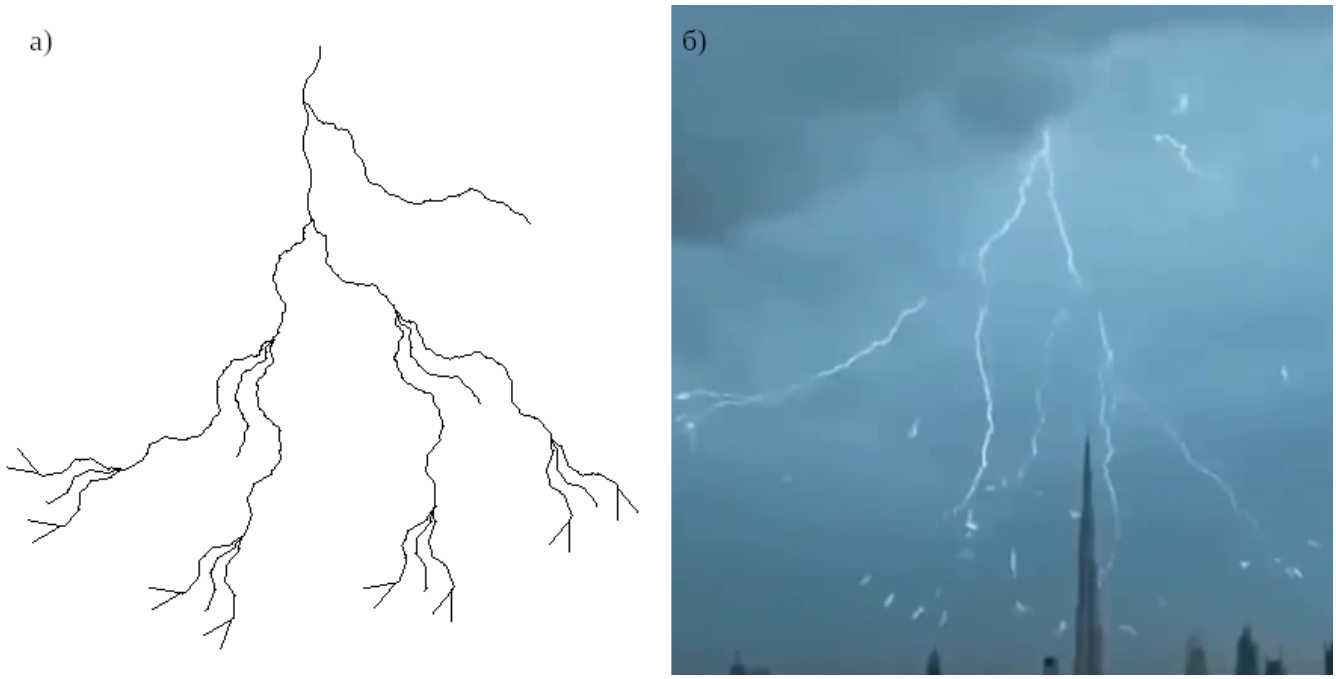

Рисунок 4.6 – Реалізація конструктора Ω_4 – а) та його прототип у природі – б)

Визначимо значення P_3 для побудови другого варіанту блискавки, який представлений на рис. 4.6, б).

$$P_3 = \{ X \to ++FF - F + + + T + L$$
$$F \to T - T - T + T + T$$
$$L \to L + L - L$$
$$T \to TT \}.$$

В результаті реалізації $\langle C_{MS}(X, P_3, 5), C_A \rangle_R \Omega_5(C_{MS})$ отримаємо багатосимвольну конструкцію $\Omega_5(C_{MS})$, яка є одним із параметрів для конкретизації і реалізації графічного конструктора. Іншим його параметром є кут $\alpha = 14.7^\circ$. На рис. 4.6, а) представлена блискавка, що є реалізацією конструктора $\langle C_{GF}(\Omega_5, 14.7, l \hookleftarrow L = 8, h \hookleftarrow L = 5, l \hookleftarrow T = 3, h \hookleftarrow T = 5), C_B \rangle_R \Omega_6(C_{GF})$.

Порівнюючи правила P_1 та P_3 можна побачити значні подібності, що говорить про те, що блискавки в цілому є схожі та подібні одна до одної. Дане спостереження підводить до висновку, що в подальшому можна розробити стохастичний конструктор, який б генерував безліч блискавок типу хмара-земля.

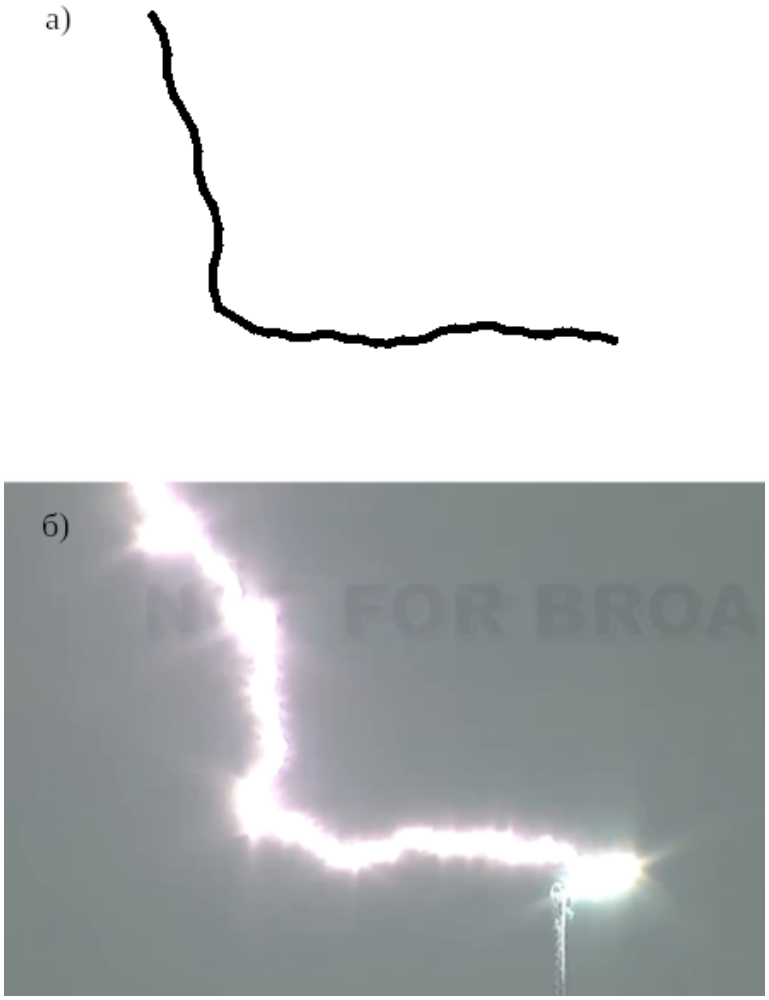

Рисунок 4.7 – Реалізація конструктора – а) та його прототип у природі – б)

Висновки до розділу 4

Проведені дослідження щодо ефективності розробленої методології виокремлення блискавок з кадрів відео, створених супутниками NASA, які демонструють як статичний фон, так і високодинамічну хмарність, показали, що результати є адекватними та придатні для подальшого використання. Результати виокремлення кадрів з блискавками, подальшої їх апроксимації та відтворення дозволили перевірити адекватність моделі поширення грозового фронту [1][2].

За результатами досліджень ефективності використання конструкторів для будуванні моделі блискавки типу хмара-земля виведено наступні тези:

– дана концепція дозволяє будувати блискавки різноманітної форми;
– правила конструкцій є простими;

- правила конструкції є подібними один до одного;
- відтворенні внаслідок реалізації конструкторів блискавки є подібними до своїх прототипів.

Для подальшого розвитку з метою покращення ефективності даного методу слід розглянути можливість використання стохастичних конструкторів.

Таким чином, конструктивно-продукційний підхід до завдань моделювання дозволяє формувати блискавки шляхом перетворень двох параметричних сімейств конструкторів.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Вимоги безпеки при виконанні робіт на робочому місці

Під час експлуатації ЕОМ на працівників можуть діяти такі небезпечні і шкідливі виробничі фактори:

Фізичні:

– електромагнітне випромінювання, при наближенні до екрану чи задньої частини відеотерміналу ближче 0,6 - 0,7 м.;
– м’яке рентгенівське випромінювання, при наближенні до ВДТ ближче 0,05 м.;
– ультрафіолетове й інфрачервоне випромінювання;
– електростатичне поле між екраном і працівником;
– можлива наявність шуму та вібрації при роботі принтерів застарілої модифікації;
– нерівномірність розподілу яскравості в полі зору;
– підвищена яскравість світлового зображення;
– ураження електричним струмом при порушенні вимог електробезпеки.

Психофізіологічні:

– напруження зору;
– напруження уваги;
– інтелектуальні навантаження;
– емоційні навантаження;
– тривалі статичні навантаження.

Через масовий характер робіт, що виконуються працівниками за допомогою комп’ютера, законодавством України чітко врегульовано норми та вимоги до використання комп’ютерної техніки на підприємстві безпосередньо, й охорона праці при роботі з комп'ютером. Основним таким документом у галузі програмної інженерії є НПАОП 0.00-7.15.18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [59].

Згідно з цим документом висуваються певні вимоги безпеки до робочих місць працівників з екранними пристроями, а саме:

– простір для зміни робочого положення та рухів;

– зведене до гранично допустимого рівня випромінювання від екранних пристроїв (вплив на людину факторів довкілля, які не спричиняють соматичних або психічних розладів, та не призводять змін стану здоров’я, працездатності, поведінки, що виходять за межі пристосувальних реакцій);

– зручна організація робочого місця для працівника з можливістю адаптувати положення пристроїв вводу та виводу даних індивідуально для кожного працівника;

– робочий стіл достатнього розміру з низькою відбивною здатністю;

– робоче крісло, що може регулюватися по висоті та мати спинку, що регулюється за нахилом, має бути стійким і дозволяти легко рухатися та займати зручне положення.

Мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Державних санітарних норм мікроклімату виробничих приміщень [60]. Згідно з яким, приміщення повинно бути укомплектоване системами центрального або індивідуального опалення, кондиціонування чи вентиляції повітря, та забезпечувати мікрокліматичні показники приведені в таблиці 5.1.

Таблиця 5.1 – Санітарні норми мікроклімату виробничих приміщень

Пора року	Категорія робіт	Температура повітря, град. С	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодний	легка-1а	22–24	40–60	0,1
	легка-1б	21–23	40–60	0,1
Теплий	легка-1а	23–25	40–60	0,1
	легка-1б	22–24	40–60	0,2

Освітлення робочого місця працівника з екранними пристроями має створювати відповідний контраст між екраном і навколишнім середовищем (з урахуванням виду роботи) та відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних

машин ДСанПІН 3.3.2.007-98 [61]. Згідно з цими вимогами освітлення приміщення повинно відповідати наступним параметрам:

– освітлення природне та штучне – розміщення екранних пристроїв у підвальних приміщеннях, на цокольних поверхах заборонено;
– вікна орієнтовані переважно на північ чи північний-схід з КПО >= 1,5%, обладнані жалюзями чи шторами;
– джерела штучного освітлення – переважно люмінесцентні лампи типу ЛБ;
– освітленість екрана має не перевищувати 300 лк, освітленість поверхні робочого столу 300-500 лк.

Вимоги безпеки під час роботи з екранними пристроями [59]:

– щодня перед початком роботи необхідно очищати екранні пристрої від пилу та інших забруднень;
– після закінчення роботи екранні пристрої слід відключати від електричної мережі;
– у разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.

Не допускається:

– виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника під час роботи з екранними пристроями;
– відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;
– працювати з екранними пристроями, у яких під час роботи виникають нехарактерні сигнали, нестабільне зображення на екрані та інші несправності.

Електробезпека будівель та приміщень, де розміщені робочі місця операторів, повинна відповідати вимогам НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів» [62]. Вплив електричного струму на людину проявляється по-різному, в залежності від його величини:

– струм до 0,6 мА не відчувається людиною;

– струм завбільшки 6 мА призводить до скорочення м'язів тієї частини, тіла, що піддалася його впливу;

– струм 50 мА вражає органи дихання та серцево-судинну систему;

– при досягненні струму порога 100мА настає фібриляція серця і, потім, його зупинка;

– при впливі на тіло людини струму в 3-4 А виникає обвуглювання ділянок тіла.

За способом захисту людини від ураження електричним струмом відео термінали, ЕОМ, периферійні пристосування ЕОМ і оснащення для обслуговування, ремонту і налагодження ЕОМ повинні відповідати I класу захисту. Вимоги електричної і механічної безпеки і засоби випробувань чи повинні бути заземлені.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ і устаткування для обслуговування, ремонту і налагодження ЕОМ виконується як окрема групова, трьох провідна мережа, шляхом прокладки фазового, нульового робочого і нульового захисного провідників. Підключення на розподільному щиті до одного контактного затиску нульового робочого і нульового захисного провідників заборонено. Площа перетину нульового робочого і нульового захисного провідника в груповій трьох провідній мережі повинна бути не менш площі перетину фазового провідника.

При роботі неприпустимо:

– експлуатація кабелів і проводів з ушкодженою ізоляцією або з ізоляцією, яка втратила захисні властивості за час експлуатації;

– застосування саморобних подовжувачів, що не відповідають вимогам ПУЕ до переносних електропроводок;

– використання ушкоджених розеток, розгалужених і сполучних коробок, вимикачів і інших електровиробів, а також ламп, скло яких має затьмарення чи опуклості;

– підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканиною й іншими горючими матеріалами, експлуатація їх зі знятими ковпаками.

5.2 Шкідливі виробничі фактори на робочому місці

Відповідно до вимог [59], робота на ЕОМ пов'язана з наступними шкідливими факторами:

– недостанє освітлення природним світлом;

– м’яке рентгенівське випромінювання;

– електромагнітне випромінювання;

– ультрафіолетове і інфрачервоне випромінювання;

– наявність шуму та вібрації;

– наявність пилу, озону, оксидів азоту й аероіонізації;

– електростатичне поле між екраном і оператором;

– відблиски на екрані монітора.

Під час розробки інструментарію щодо виконання роботи «Конструктивно-продукційне моделювання молнієвої активності» найбільш шкідливими факторами були недолік освітлення природним світлом та відблиски на екрані монітора. Згідно з [61] зазначення освітлення освітленості на поверхні робочого столу в зоні розміщення документів має становити 300-500 лк. Якщо ці значення освітленості неможливо забезпечити системою загального освітлення, допускається використовувати місцеве освітлення. При цьому світильники місцевого освітлення слід встановлювати таким чином, щоб не створювати бліків на поверхні екрана, а освітленість екрана має не перевищувати 300лк. Необхідно обмежувати нерівномірність розподілу яскравості в полі зору працюючих з ВДТ. При цьому співвідношення яскравостей робочих поверхонь має бути не більшим ніж 3:1, а співвідношення яскравостей робочих поверхонь та поверхонь стін, обладнання тощо – 5:1. Саме тому при роботі у вечірній час та похмурі дні використовувалось джерело місцевого освітлення, однак з LED-лампочкою, що є небажаним при напруженій роботі зорового апарату.

Відблиски на екрані монітора, що виникають при неправильному освітленні, приводить до погіршення зору, а у випадку тривалого впливу даного небезпечного фактора, може призвести до повної втрати зору. З метою зниження рівня впливу на

робітника даного шкідливого фактора, варто дотримуватись норми щодо природного і штучного освітлення ДБН В.2.5-28:2018 «Природне і штучне освітлення» [63] або застосовувати рідкокристалічні монітори, які в силу своєї конструкції і використовуваних матеріалів мають менший коефіцієнт відбиття світла, ніж електронно-променевий монітор, тому відблисків на них практично не буває. При роботі над програмою використовувався рідкокристалічний монітор, однак вікна у робочій кімнаті орієнтовані на південь. Проблема з надлишковим прямим сонячним освітлення у яскраві дні була вирішена шляхом обладнання вікон жалюзями.

Саме тому для усунення шкідливого впливу на зір необхідно виконувати комплекс вправ для очей, який рекомендований [61]:

– закрити очі на 6 с, різко відкрити, подивитись вгору та вниз;

– колові рухи очима: вниз-вліво-вгору-вправо;

– закрити очі, різко відкрити, подивитись на кінчик носа;

– швидко моргати;

– діагональні рухи очима;

– дивитись у вікно;

– з закритими очима переводити погляд вправо-вліво, вгору-вниз;

– просто сидіти з заплющеними очима.

Визначається [62], що інженери-програмісти виконують роботу переважно з відеотерміналом та документацією при необхідності і інтенсивного обміну інформацією з ЕОМ і високою частиною прийняття рішень. Робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті верхніх кінцівок. Робота виконується в режимі діалогу з ЕОМ у вільному темпі з періодичним пошуком помилок в умовах дефіциту часу. Тому рекомендовано для розробників програм із застосуванням ЕОМ організовувати перерву під час роботи тривалістю 15 хвилин через кожну годину роботи. Забороняється працювати більше 4 годин поспіль. Для ліквідації шкідливих факторів, пов'язаних з вимушеною

робочою позою від час перерв рекомендовано виконувати вправи для для поліпшення мозкового кровообігу, зняття психоневрологічного навантаження, для рук та спини. Розглянемо найбільш актуальні під час роботи над дипломним проектом.

Комплекс вправ для поліпшення мозкового кровообігу:

– нахили і повороти голови підвищують кровоносних судин еластичність;
– тренування вестибулярного апарату сприяє розширенню кровоносних судин головного мозку;
– дихальні вправи, особливо дихання через ніс, збільшують їх кровонаповнення;
– все це підсилює мозковий кровообіг, тим самим полегшуючи розумову діяльність.

Комплекс вправ для рук. Вправи можна робити в будь-який час протягом дня, спочатку по 2-3 рази, поступово збільшуючи навантаження до 6-10 разів:

– згинати й розгинати пальці та руки в ліктьових суглобах;
– обертати кулаки за годинниковою стрілкою і проти;
– згинати і розгинати руки;
– підняти руки в сторони до рівня плечей, потім опустити, обертати їх;
– масажувати пальцями кісті з тильного і зовнішнього боків.

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях, обладнаних ВДТ ЕОМ і ПЕОМ, мають відповідати вимогам ДСН 3.3.6.037-99 «Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку» [64]. Устаткування, що становить джерело шуму (АЦП, принтери тощо), слід розташовувати поза приміщенням для роботи ВДТ ЕОМ і ПЕОМ. Для забезпечення допустимих рівнів шуму на робочих місцях слід застосовувати засоби звукопоглинання, вибір яких має обґрунтовуватись спеціальними інженерно-акустичними розрахунками [61].

Іонізуючі електромагнітні випромінювання на відстані 0,05 м від екрана до корпуса відеотермінала при будь-яких положеннях регулювальних пристроїв не

повинна перевищувати 7,74 х 10 в ст.-12 А/кг, що відповідає еквівалентній дозі 0,1 мбер/год (табл. 5.2).

Таблиця 5.2 – Рівні іонізації повітря приміщень при роботі на ЕОМ

Рівні	Число іонів в 1 см куб. повітря	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Максимальний рівень рентгенівського випромінювання на робочому місці оператора комп'ютера звичайно не перевищує 10 мкбер/год, а інтенсивність ультрафіолетового і інфрачервоного випромінювань від екрану монітора лежить в межах 10-100 мВт/м2.

5.3 Дії працівників у аварійних ситуаціях

Згідно з Положеннями щодо розробки планів локалізації та ліквідації аварійних ситуацій і аварій [65] аварійна ситуація – стан потенційно небезпечного об'єкта, що характеризується порушенням меж та/або умов безпечної експлуатації, але не перейшов в аварію, при якому всі несприятливі впливи джерел небезпеки на персонал, населення та навколишнє середовище утримуються у прийнятних межах за допомогою відповідних технічних засобів, передбачених проектом.

Виробнича аварія – це раптова зупинка роботи або порушення установленого процесу виробництва на об'єкті, яка призводить до пошкодження або знищення матеріальних цінностей, травмування або загибелі людей.

Джерелами небезпечних і шкідливих виробничих чинників можуть бути:

– нерегламентовані режими роботи технологічного устаткування, транспортні засоби;

– механізми, обладнання;

– електромережі, електрифіковане устаткування та інструменти;

– інженерні комунікації;

– роботи, що призводять до психофізіологічних перевантажень;

– токсичні речовини;

– помилкові дії працівників;

– аварії.

У випадку виникнення **аварійних ситуацій** або пожежі працівник мусить:

– припинити роботу;

– якнайшвидше сповістити про аварію (пожежу) керівництва об’єкта;

– приступити до ліквідації (локалізації) аварії (пожежі) наявними засобами;

– за необхідності викликати інші аварійно-рятувальні (пожежні, медичні тощо) підрозділи.

У разі **нещасного випадку**:

– працівник, який його виявив, або сам потерпілий повинен терміново повідомити безпосереднього керівника об’єкта або його заступника;

– організувати надання невідкладної медичної допомоги потерпілому, та викликати бригаду швидкої медичної допомоги (доставити потерпілого до лікувального закладу);

– зберегти до прибуття комісії з розслідування обстановку на робочому місці та устаткування в такому стані, у якому вони були на момент події (якщо це не загрожує життю й здоров'ю інших працівників і не призведе до більш тяжких наслідків).

– У разі виникнення **пожежі** або її ознаки (задимлення, запах горіння або тління різних матеріалів, підвищення температури в приміщенні тощо) необхідно:

– негайно повідомити про це службу порятунку за телефоном: 101 (при цьому слід чітко назвати адресу об'єкта, місце виникнення пожежі, а також свою посаду та прізвище);

– організувати оповіщення працівників та відвідувачів про пожежу;

– організувати евакуацію людей з будівлі до безпечного місця;

– повідомити керівництво про виникнення пожежі;

– вжити заходів для збереження матеріальних цінностей та гасіння (локалізації) пожежі наявними засобами пожежогасіння;

– організувати зустріч пожежних підрозділів;

– у разі необхідності викликати інші аварійно-рятувальні служби (медичну, газову та ін.);

– виходячи з приміщення, де виникла пожежа, потрібно щільно зачинити двері, щоб зменшити надходження кисню до приміщення.

У випадку **припинення подачі електроенергії** відключити електричні прилади від мережі. Виявивши обрив електропроводів, пошкодження їхньої ізоляції негайно повідомити керівництво та відповідальну особу за стан цивільного захисту (техногенної безпеки) та охорони праці на підприємстві.

ВИСНОВКИ

Було проведено дослідження моделювання процесів динамічної поведінки спалахів блискавок на статичному фоні та у рухомих грозових фронтах з високодинамічною хмарністю на супутникових відео, створених погодними супутниками. В результаті були розроблені методологія та програмне забезпечення для виявлення блискавок на статичному фоні, а також ореолів і самих блискавок на відеозображеннях погодних супутників, що демонструють суттєво динамічний хмарний покрив.

Особливість відео, що демонструють рухому грозові фронти – фільтрація кадрів під час зйомки таким чином, що блискавка виникає в оточенні кольорового контуру. Особливі якості відображення контуру в кольорових моделях Lab та LCH були враховані при розробці фільтрів для ідентифікації контурів блискавки та самих блискавок. В результаті було запропоновано два фільтри для вирішення завдання. Серія експериментів показала, що квадратичний є найбільш ефективним. Незважаючи на це, лінійний має меншу обчислювальну складність, і тому його можна використовувати в тих випадках, коли потрібно ідентифікувати блискавку за короткий час.

Для подальшого моделювання блискавок типу хмара-земля була розроблена множина конструкторів. Реалізація параметричних багатосимвольних конструкторів дозволяє формувати фрактальні послідовності символів. Конструктор-перетворювач із символьного рядка в графічне представлення формує зображення блискавки. Для автоматизації даного процесу було розроблено програмне забезпечення. За результатами досліджень ефективності використання конструкторів для будуванні моделі блискавки типу хмара-земля виведено наступні тези:

- дана концепція дозволяє будувати блискавки різноманітної форми;
- правила конструкцій є простими та подібними один до одного;
- відтворенні внаслідок реалізації конструкторів блискавки є подібними до своїх прототипів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Shynkarenko, V. Modeling of Lightning Flashes in Thunderstorm Front by Constructive Production of Fractal Time Series / V. Shynkarenko, K. Lytvynenko, R. Chyhir, I. Nikitina // In: Shakhovska, N., Medykovskyy, M. (eds) Advances in Intelligent Systems and Computing IV. – Springer, 2020. С. 173-185. Режим доступу: https://doi.org/10.1007/978-3-030-33695-0_13. Перевірено: 25.11.2020.
2. Shynkarenko, V. Constructive-synthesizing Modeling of Lightning Flashes in the Dynamic Thunderstorm Front / V. Shynkarenko, I. Nikitina, R. Chyhir // In: Shakhovska, N., Medykovskyy, M. (eds) Advances in Intelligent Systems and Computing V. – Springer, 2020. – 19 с.
3. Shynkarenko, V. Constructive Modeling of Lightning Activity in Thunderstorm Front / V. Shynkarenko, K. Lytvynenko, R. Chyhir, I. Sansiieva // In: 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT). – 2019. С. 92–95. Режим доступу: https://doi.org/110.1109/STC-CSIT.2019.8929754. Перевірено: 25.11.2020.
4. Shynkarenko, V. Lightning Recognition on the Filtered Videos in the Dynamic Clouds / V. Shynkarenko, I. Nikitina // In: IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies. – 2020.
5. Williams, E.R. Lightning and climate: A review [Текст] / E.R Williams // In: Atmospheric Research. – 2005. – №76(1-4). – С. 272–287. Режим доступу: https://doi.org/10.1016/j.atmosres.2004.11.014. Перевірено: 25.11.2020.
6. Predicting the potential for lightning activity in Mediterranean storms based on the Weather Research and Forecasting (WRF) model dynamic andmicrophysical fields [Текст] / Y. Yair and others // In: J. Geophys. Res. – 2010. – 115 с. Режим доступу: https://doi.org/10.1029/2008JD010868. Перевірено: 25.11.2020.
7. Lightning, convective rain and solar activity – Over the South/Southeast Asia [Текст] / S. Devendraa and others // In: Atmospheric Research. – 2013. – №120. – С. 99–111. Режим доступу: https://doi.org/10.1016/j.atmosres.2012.07.026. Перевірено: 25.11.2020.

8. A ten-year analysis of cloud-to-ground lightning activity over the Eastern Mediterranean region [Текст] / E. Galanaki and others // In: Atmospheric Research. – 2015. – №166. – С. 213–222.
9. Ahrens, M. Lightning fires and lightning strikes [Текст] / M. Ahrens // In: National Fire Protection Association, Quincy. – 2013. – 31 с.
10. Sátoria, G. Variability of global lightning activity on the ENSO time scale [Текст] / G. Sátoria, E.Williams, I. Lempergeraams // In: Atmospheric Research. – 2009. – №91(2–4). – С. 500–507. Режим доступу: https://doi.org/10.1016/j.atmosres. 2008.06.014. Перевірено: 25.11.2020.
11. The National –Severe Storms Laboratory Research: Thunderstorms. Режим доступу: https://www.nssl.noaa.gov/research/thunderstorms/. Перевірено: 25.11.2020.
12. Fuchs, B. R. Climatological analyses of LMA data with an open-source lightning flashclustering algorithm [Текст] / B. R. Fuchs // In: Journal of Geophysical Research: Atmospheres. – 2016. – №121(14). – С. 8625–8648.
13. Sen, H. An Investigation On Modelling And Simulation Of Real Lightning Strikes [Текст] / H, Sen, C. Uzunoglu, A. Yilmaz // Manager's Journal on Electrical Engineering; Nagercoil. – 2019. – №12(4). – С. 1–9. Режим доступу: https://doi.org/10.26634/jee.12.4.15833. Перевірено: 25.11.2020.
14. Fractal model of lightning channel for simulating lightning strikes to transmission lines [Текст] / J. He and others // Science in China Series E: Technological Sciences. – 2009. – №52(11). – С. 3135–314. Режим доступу: https://doi.org/10.1007/s11431-009-0259-1. Перевірено: 25.11.2020.
15. Simulation of cloud-to-ground lightning strikes to structures based on an improved stochastic lightning model [Текст] / R. Jiang and others // Journal of Atmospheric and Solar–Terrestrial Physics. – 2020. – №203. Режим доступу: https://doi.org/10.1016/j.jastp.2020.105274. Перевірено: 25.11.2020.
16. Oklahoma Lightning Mapping Array. Режим доступу: http://lightning.nmt.edu/oklma/view_rthour.php?date=200522&time=0800. Перевірено: 25.11.2020.

17. The National Severe Storms Laboratory Research: Lightning. Режим доступу: https://www.nssl.noaa.gov/research/lightning/. Перевірено: 03.12.2020.
18. RAMSDIS Online. Режим доступу: http://rammb.cira.colostate.edu/ramsdis/online/. Перевірено: 03.12.2020.
19. Sainte, V. Satellite Image Time Series Classification with Pixel-Set Encoders and Temporal Self-Attention [Текст] / V. Sainte and others // arXiv preprint. – 2019. – arXiv: 1911.07757.
20. Fu, H. Cloud Detection for FY Meteorology Satellite Basedon Ensemble Thresholds and Random Forests Approach [Текст] / H. Fu and others // In: Remote Sensing. – 2019. – №11(1). – 44. – С. 1–28. Режим доступу: https://doi.org/10.3390/rs11010044. Перевірено: 25.11.2020.
21. Peterson, M. Thunderstorm Cloud-Type Classification from Space-Based Lightning Images [Текст] / M. Peterson, S. Rudlosky and D. Zhang // In: Monthly Weather Review. – 2020. – С. 1891–1898. Режим доступу: https://doi.org/10.1175/MWR-D-19-0365.1. Перевірено: 25.11.2020.
22. Jin, W. Cloud Types Identification for Meteorological Satellite Image Using Multiple Sparse Representation Classifiers via Decision Fusion [Текст] / W. Jin and others // In: IEEE Access. – 2019. – 7. С. 8675–8688. Режим доступу: https://doi.org/10.1109/ACCESS.2018.2890295. Перевірено: 25.11.2020.
23. Zheng, X. Detecting Comma-Shaped Clouds for Severe Weather Forecasting Using Shape and Motion [Текст] / X. Zheng // In: IEEE Transactions on Geoscience and Remote Sensing. 2019. – №57(6). – С. 1–14. Режим доступу: https://doi.org/10.1109/TGRS.2018.2887206. Перевірено: 25.11.2020.
24. Radhika, K., A Neural Network Based Classification of Satellite Images for Change Detection Applications [Текст] / K. Radhika, S. Varadarajan // Cogent Engineering. – 2018. – №5. – С. 1–9. Режим доступу: https://doi.org/10.1080/23311916.2018.1484587. Перевірено: 25.11.2020.
25. Chen, G. Use of VIIRS DNB Satellite Images to Detect Nighttime Fishing Vessel Lights in Yellow Sea [Текст] / G. Chen // In: Proceedings of the 3rd International

Conference on Computer Science and Application Engineering. – 2019. – С. 1–5. Режим доступу: https://doi.org/10.1145/3331453.3361661. Перевірено: 25.11.2020.

26. Elvidge, D. Automatic Boat Identification System for VIIRS Low Light Imaging Data / D. Elvidge // Remote Sensing. – 2015. – №7(3). – С. 3020–3036. Режим доступу: https://doi.org/10.3390/rs70303020. Перевірено: 25.11.2020.
27. GLM Flash Event Density vs Centroids. Режим доступу: https://inside.nssl.noaa.gov/ewp/2019/05/16/glm-flash-event-density-vs-centroids/. Перевірено: 03.12.2020.
28. Cannons J. Modelling and simulationof lightning discharge patterns / J. Cannons. 2000. – 120 с.
29. L-SYSTEMS GENERATOR 2.4. Режим доступу: http://www.digitalpoiesis.org/. Перевірено: 03.12.2020.
30. Конушин А. А. Выделение объектов в видеопотоке на основеразрезов графов / Конушин, Н. Четвериков // The 22nd International Conference on Computer Graphics and Vision. – 2009.
31. KaewTraKulPong, P. An Improved Adaptive Background Mixture Model for Real-time Tracking with Shadow Detection / KaewTraKulPong, R. Bowden // The 2nd European Workshop on Advanced Video Based Surveillance Systems. – 2001.
32. Zivkovic, Z. Improved Adaptive Gaussian Mixture Model for Background Subtraction / Z. Zivkovic // ICPR '04 Proceedings of the Pattern Recognition. – 2004. – 28–31 с.
33. Алпатов Б.А. Алгоритм автоматического обнаружения, выделения и оценки динамических объектов возникающих в последовательности телевизионных кадров / Б.А. Алпатов, К.А. Бохан // 3-я Международная Конференция DSPA-2000. – 2001.
34. First Images from GOES-16 Lightning Mapper. Режим доступу: https://www.americaspace.com/2017/03/07/goes-16-satellite-returns-first-lightning-mapping-images-like-never-seen-before. Перевірено: 03.12.2020.

35. Бутаков, Е. А. Обработка изображений на ЭВМ / Е. А. Бутаков, В. И. Остроский, И. Л. Фадеев. – М.: Радио и связь. – 1978. – 240 с.
36. The International Commission on Illumination [Електронний ресурс]. Режим доступу: http://cie.co.at/. Перевірено: 03.12.2020.
37. Sayed, M. An accurate method to calculate the color difference in a single image / M. Sayed, M. Albashier // International Conference on Robotics, Automation and Sciences (ICORAS). – 2017. – С. 1–3. Режим доступу: https://doi.org/10.1109/ICORAS.2017.8308079. Перевірено: 03.12.2020.
38. Color Equations. RGB to XYZ [Електронний ресурс]. Режим доступу: http://www.brucelindbloom.com/index.html?Eqn_RGB_to_XYZ.html. Перевірено: 03.12.2020.
39. Color Equations. XYZ to Lab [Електронний ресурс]. Режим доступу: http://www.brucelindbloom.com/index.html?Eqn_XYZ_to_Lab.html. Перевірено: 03.12.2020.
40. Color Equations. Lab to LCH [Електронний ресурс]. Режим доступу: http://www.brucelindbloom.com/index.html?Eqn_Lab_to_LCH.html. Перевірено: 03.12.2020.
41. Nocturnal thunderstorms near Lake Maracaibo are known for their «Catatumbo Lightning», 2018-08-15 [Online]. Режим доступу: http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes16/20180815000000&number_of_images_to_display=100&loop_speed_ms=15. Перевірено: 03.12.2020.
42. The next storm to hit the U.S. West Coast is already producing lightning, 2019-01-16 [Online]. Available: Режим доступу: http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes16/20190116000000&number_of_images_to_display=120&loop_speed_ms=100. Перевірено: 03.12.2020.
43. Фершильд, М.Д Модели цветового восприятия / М.Д. Фершильд. – 2004. – 430 с.
44. Thanh, D. A Skin Lesion Segmentation Method for Dermoscopic Images Based on Adaptive Thresholding with Normalization of Color Models / D. Thanh and others //

In: 2019 6th International Conference on Electrical and Electronics Engineering (ICEEE). – 2019. – С. 116–120. Режим доступу: https://doi.org/10.1109/ ICEEE2019.019.00030. Перевірено: 03.12.2020.

45. Patil, J. Screening of Damage Regions in Retinopathy Using Segmentation-Color Space Selection / J. Patil and S. Chaudhari // In: International Journal Multimedia and Image Processing (IJMIP). – 2017. – №7(1), С. 362–365. Режим доступу: https://doi.org/10.20533/ijmip.2042.4647.2017.0044. Перевірено: 03.12.2020.
46. Thanh L. An Adaptive Local Thresholding Roads Segmentation Method for Satellite Aerial Images with Normalized HSV and Lab Color Models [Текст] / L. Thanh, D. Thanh // In: V. Solanki V., Hoang M., Lu Z., Pattnaik P. (eds) Intelligent Computing in Engineering. Advances in Intelligent Systems and Computing. – Springer, 2020. – С. 865–872.
47. Kolkur S. Human skin detection using RGB, HSV and YCbCr color models [Текст] / S. Kolkur and others // arXiv preprint. – 2017. arXiv:1708.02694.
48. Mporas, I. Color Models for Skin Lesion Classification from Dermatoscopic Images [Текст] / I. Mporas, I. Perikos and M. Paraskevas // In: Advances in Integrations of Intelligent Methods. Smart Innovation, Systems and Technologies. – С. 85–98. Режим доступу: https://doi.org/10.1007/978-981-15-1918-5_5. Перевірено: 03.12.2020.
49. Chuang, C. Soil Moisture Estimation Using Multiple Color Spaces in Digital Image Analysis [Текст] / C. Chuang // In: Geophysical Research Abstracts. – 2019. – 21 с.
50. Shynkarenko, V. Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part I. Generalized Formal Constructive-Synthesizing Structure [Текст] / V. Shynkarenko, V. Illman // In: Cybernetics and Systems Analysis. – 2014. – №50(5), С. 665–662. Режим доступу: https://doi.org/10.1007/ s10559-014-9655-z. Перевірено: 03.12.2020.
51. Shynkarenko, V. Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part II. Refining Transformations [Текст] / V. Shynkarenko, V. Illman

// In: Cybernetics and Systems Analysis. – 2014. – №50(6), С. 829–841. Режим доступу: https://doi.org/10.1007/s10559-014-9674-9. Перевірено: 03.12.2020.

52. Shynkarenko, V. Structural models of algorithms in problems of applied programming. I. Formal algorithmic structures [Текст] / V. Shynkarenko, V. Illman, V. Skalozub // In: Cybernetics and Systems Analysis. – 2009. – №45(3). – С. 329–339. Режим доступу: https://doi.org/10.1007/s10559-009-9118-0. Перевірено: 03.12.2020.
53. Skalozub, V. Development of ontological support of constructive-synthesizing modeling of information systems [Текст] / V. Skalozub, V. Illman, V. Shynkarenko // In: Eastern-European Journal of Enterprise Technologies. – 2017. – №6/4(90). – С. 58–69. Режим доступу: https://doi.org/10.15587/1729-4061.2017.119497. Перевірено: 03.12.2020.
54. Lindenmayer, A. Mathematical models for cellular interaction in development [Текст] / A. Lindenmayer // In: Journal of Theoretical Biology. – 1968. – №18. – С. 280–315.
55. Qt documentation [Електронний ресурс]. Режим доступу: https://doc.qt.io/. Перевірено: 03.12.2020.
56. Швец, А. Погружение в паттерны проектирования [Текст] / А. Швец – 2018. – 402с.
57. Тидвелл, Д. Разработка пользовательских интерфейсов [Текст] / Д. Тидвелл. – Питер, 2008. – 480 с.
58. Prusinkiewicz, P. The Algorithmic Beauty of Plants [Текст] / P. Prusinkiewicz, A. Lindenmayer – New York, 1990. – 240 с.
59. НПАОП 0.00-7.15.18. Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями [Електронний ресурс] – Режим доступу: http://zakon.rada.gov.ua/laws/show/z0508-18. Перевірено: 03.12.2020.
60. ДСН 3.3.6.042-99. Державні санітарні норми мікроклімату виробничих приміщень [Електронний ресурс] – Режим доступу: http://zakon.rada.gov.ua/rada/show/va042282-99. Перевірено: 03.12.2020.

61. ДСанПІН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин [Електронний ресурс] – Режим доступу: http://mozdocs.kiev.ua/view.php?id=2445. Перевірено: 03.12.2020.
62. НПАОП 40.1-1.21-98. Правила безпечної експлуатації електроустановок споживачів [Електронний ресурс] – Режим доступу: https://dnaop.com/html/2029/doc-НПАОП_40.1-1.21-98. Перевірено: 03.12.2020.
63. ДБН В.2.5-28:2018. Природне і штучне освітлення [Електроний ресурс] – Режим доступу: https://dbn.co.ua/load/normativy/dbn/dbn_v_2_5_28/1-1-0-1188. Перевірено: 03.12.2020.
64. ДСН 3.3.6.037-99. Державні санітарні норми виробничого шуму, ультразвуку та інфразвуку [Електронний ресурс] – Режим доступу: https://dnaop.com/html/1642/doc-ДСН_3.3.6.037-99/. Перевірено: 03.12.2020.
65. НПАОП 0.00-4.33-99. Положення щодо розробки планів локалізації та ліквідації аварійних ситуацій і аварій [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0424-99/ed20121114. Перевірено: 03.12.2020.

ДОДАТКИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Дніпровського
національного університету
залізничного транспорту імені
академіка В. Лазаряна
_____ Б. Є. Боднар
«03» грудня 2020 р.

АВТОМАТИЗОВАНА СИСТЕМА КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

Технічне завдання

ЛИСТ ЗАТВЕРДЖЕННЯ

01116130.01188-01-ЛЗ

Завідувач кафедри КІТ
_____ В. І. Шинкаренко
«03» грудня 2020 р.

Керівник розробки
_____ В. І. Шинкаренко
«03» грудня 2020 р.

Виконавець
_____ І. М. Нікітіна
«03» грудня 2020 р.

Нормоконтролер
_____ О. С. Куроп'ятник
«03» грудня 2020 р.

2020

ЗАТВЕРДЖЕНО

01116130.01188-01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

Технічне завдання

01116130.01188-01

Листів 21

2020

АНОТАЦІЯ

Документ 01116130.01188-01 «Автоматизована система конструктивно-продукційного моделювання молнієвої активності. Технічне завдання» входить до складу програмної документації до програми, яка реалізує розпізнавання блискавок на кадрах поширення грозового фронту та моделює поширення блискавки типу хмара-земля у атмосфері.

У даному документі представлене призначення та область застосування програмного продукту, основні вимоги, стадії та строки виконання проекту, технічні та техніко-економічні показники, що пред’являються до програмного продукту.

ЗМІСТ

ВСТУП

Програмний продукт «Автоматизована система конструктивно-продукційного моделювання молнієвої активності» призначено для пошуку спалахів блискавок на кадрах відео з метеосупутників та моделювання блискавки типу хмара-земля у атмосфері.

На сьогодні блискавка продовжує складати суттєву загрозу для людей та обладнання. Прогнозування місць вірогідного влучення блискавок дозволило б завчасно провести профілактичні роботи на обладнані щодо захисту від грозового імпульсу та бути готовими вжити заходів з ліквідації ймовірних руйнівних наслідків від ударів блискавок шляхом:

– передбачення резервних джерел електропостачання у випадках знаходження електричних підстанцій та електростанцій в зонах ризику ураження;
– посилення протипожежних заходів на об'єктах з підвищеною вогненебезпекою (газосховища, нафтосховища, ліси, електричні підстанції та електростанції).

Пошук зони найбільш ймовірного ураження виконується на основі математичних моделей розповсюдження грозового фронту та поширення блискавки від хмари до землі. Для побудування моделі розповсюдження фронту необхідно проаналізувати характер молнієвої активності, використовуючи дані з погодних супутників, що дозволить розрахувати координати виникнення розряду. Автоматизована система конструктивно-продукційного моделювання молнієвої активності дозволила б зібрати такі дані. Подальше моделювання блискавки у вертикальному напрямку дозволить розрахувати координати її влучення. Розроблювана програма дозволила б побудувати необхідні моделі.

Основна термінологія: БЛИСКАВКА, ГРОЗОВИЙ ФРОНТ, РОЗПІЗНАВАННЯ ОБРАЗІВ, КОЛЬОРОВІ СХЕМИ, L-СИСТЕМИ.

Причинами виникнення розробки є відсутність повноцінних аналогів у відкритому доступі.

Область застосування: метеорологічні центри України.

1 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ ректора Дніпропетровського національного університету залізничного транспорту імені академіка В. Лазаряна проф. Пшінька О. М. №779ст від 10.10.2019р. «Про призначення наукових керівників та затвердження тем магістерських робіт» факультету «Комп'ютерні технології і системи» за спеціальністю 121 «Інженерія програмного забезпечення».

Тема роботи: розробка програми «Конструктивно-продукційне моделювання молнієвої активності», керівник розробки проф. Шинкаренко В.І.

2 ПРИЗНАЧЕННЯ РОЗРОБКИ

2.1 Функціональне призначення

Функціональним призначенням розробки є:

– виокремлення блискавок з відеофайлів, створених метрологічними супутниками, які демонструють процес поширення грозового фронту;
– розрахунок заданих параметрів виокремлених блискавок;
– моделювання поширення блискавки типу хмара-земля у атмосфері з подальшою візуалізацією.

2.2 Експлуатаційне призначення

Експлуатаційне призначення полягає у дослідженні характеру поширення грозового фронту та блискавок у атмосфері, що дає можливості передбачення поведінки блискавок і може бути використано для:

– попередження аварійних відключень електромереж;
– планування графіків, що забезпечують безпечне виконання робіт на електричних підстанціях та електростанціях;
– складання графіків проведення профілактичних робіт на обладнані по захисту від грозового імпульсу (громовідводи, заземлення обладнання, справність розрядників та обмежувачів перенапруг) перед грозовим сезоном;
– посилення протипожежних заходів на об'єктах з підвищеною вогненебезпекою (газосховища, нафтосховища, лісові масиви).

3 ВИМОГИ ДО ПРОГРАМИ

3.1 Вимоги до функціональних характеристик

Розроблювана програма розрахована на використання на ЕОМ та спроектована з урахуванням можливості подальшого розвитку методом додавання нових алгоритмів розпізнавання блискавок.

Вимоги до функціональних характеристик програмного забезпечення:

– завантаження кадрів з відеофайлів, створених метеосупутниками;

– виокремлення блискавок з вхідних зображень, результатом якого є нове зображення з блискавками на однотонному фоні (рис. 3.1);

– розрахунок параметрів блискавок з серії кадрів з виокремленими блискавками таких як: координати виникнення відносно верхнього лівого кута вхідного кадру, розмір у пікселях та номер кадру оброблюваної блискавки;

– конструювання моделі (у двовимірному просторі з можливістю вказати довжину і початкову ширину не-терміналу) поширення блискавки типу хмара-земля у атмосфері з подальшою візуалізацією (блискавка має бути чорного кольору на білому фоні).

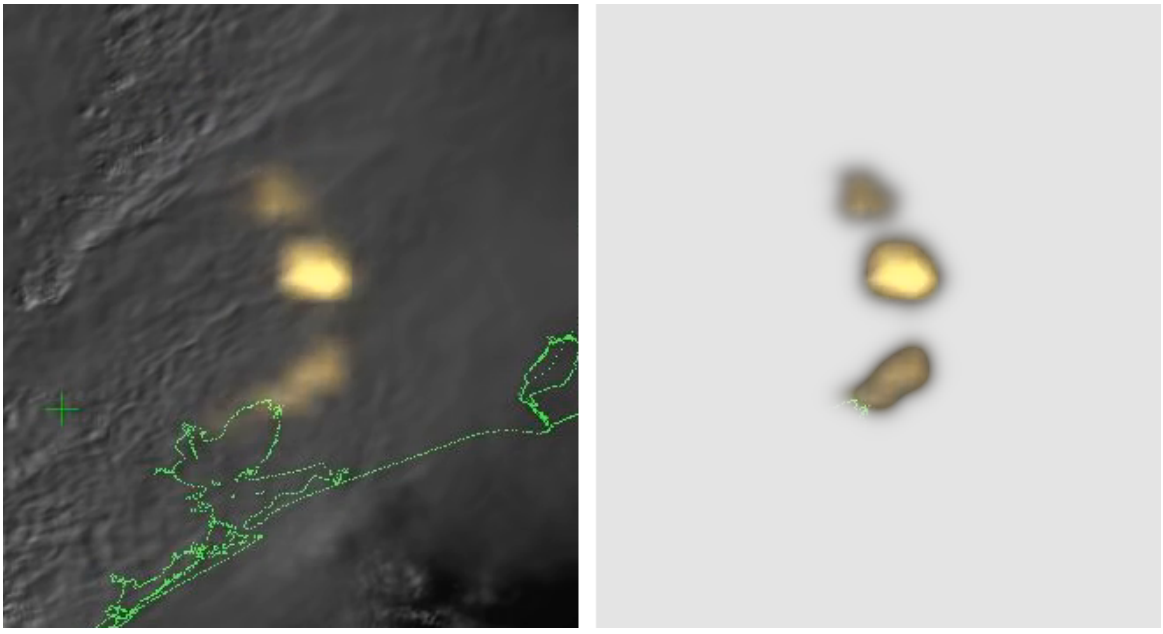

Рисунок 3.1 – Приклад виокремлення блискавок:

а – вхідне зображення, б – результат

Часові характеристики:

– реакція програми на запит не повинна перевищувати 0.8 сек.;

– обробка одного кадру програми не повинна перевищувати 30 сек.

Вхідні дані наступні:

– кадри відео, відзнятого супутником у форматі .jpg або .png, що показують процес поширення грозового фронту та мають статичний фон або, у разі динамічного фону, демонструють блискавки у вигляді яскравої плями з чітким кольоровим ореолом;

– назва директорії для збереження результату виокремлення блискавок;

– назва файлу для збереження результату розрахунку параметрів блискавок;

– параметри обробки файлу з параметрами виокремлених блискавок:

– шлях до директорії збереження результату;

– кількість кадрів, які необхідно пропустити з початку, беззнакове ціле;

– кількість вхідних кадрів, які необхідно розмістити на результатному, беззнакове ціле;

– через яку кількість кадрів змінювати колір;

– правила продукції для моделювання блискавки типу хмара-земля.

Вихідні дані наступні:

– кадри з виокремленими блискавками з вхідних зображень зображень з форматом та ім'ям, які відповідають вхідним;

– файл формату .txt з розрахованими параметрами блискавок для серії оброблених кадрів, де параметри кожної блискавки є рядок файлу з наступними даними, розділеними пробілом: координати виникнення блискавки відносно верхнього лівого кута вхідного кадру (x та y) у пікселях, радіус блискавки у пікселях, розрахований за (3.1) та порядковий номер кадру оброблюваної блискавки у серії, що обробляється; рядки мають бути впорядковані за номером кадру у порядку зростання; порядок блискавок з однаковим номером неважливий;

$$r = \sqrt{\frac{S}{\pi}}\ , \qquad (3.1)$$

де S – кількість пікселів, що складає блискавку;

– кадри, на які нанесені блискавки апроксимовані до кола згідно з розрахованими параметрами;

– зображення блискавки типу хмара-земля, сконструйоване відповідно до розроблених правил продукції, виведене у відведене вікно у програмі.

3.2 Вимоги до надійності

Основні вимоги надійності:

– контроль за введеними даними, не допускаючи тим самим введення до полів даних типу, що не відповідає очікуваному, а також таких значень, що виходять за межі очікуваного діапазону;

– не допускання виконання операції при наявності незаповнених обов'язкових полів даних;

– допускається аварійне завершення програми з подальшою втратою введених користувачем значень до полів один раз на 100 запусків програми;

– програма не повинна допускати втрату або пошкодження даних на пристрої.

3.3 Умови експлуатації

Даний програмний продукт може використовуватись в умовах, які відповідають вимогам документу [1].

Для нормального функціонування програмного продукту необхідно виконання наступних вимог:

– ЕОМ повинні відповідати вимогам чинних в Україні стандартів, нормативних актів з охорони праці [2];

– програма повинна експлуатуватись у приміщенні, призначеному для роботи з ЕОМ, з відповідними кліматичними умовами: температура 21^{o} – 25^{o}С та вологість 40 - 60%

– стан технічних засобів повинен задовольняти відповідним нормам та вимогам;

Для забезпечення надійного функціонування програмного продукту необхідно дотримуватися таких умов:

– програма призначена для роботи на ЕВМ з операційною системою сімейства GNU/Linux;
– працювати з програмою може людина, яка володіє навичками роботи з ОС у графічному режимі та ознайомилась з керівництвом користувача.

3.4 Вимоги до складу і параметрів технічних засобів

Пристрій, що має щонайменше 128 Мб вільного місця на вбудованому носії інформації, 1024 Мб оперативної пам’яті та центральний процесор з тактовою частотою 1 ГГц та більше для коректної роботи. Для зручної роботи з програмою необхідно мати монітор з роздільною здатністю 800*600 і вище, маніпулятор «миша» та клавіатуру. Для встановлення ПЗ необхідна або наявність CD/DVD приводу чи USB роз’єму, або підключення до мережі Інтернет.

3.5 Вимоги до інформаційної та програмної сумісності

Вимоги до інформаційної та програмної сумісності: ОС сімейства GNU/Linux 3 та вище.

3.6 Вимоги до маркування і упаковки

Програмний продукт може маркуватися як зображено на рис. 3.1:

Автоматизована система конструктивно-продукційного моделювання молнієвої активності, 1.0
Нікітіна Ірина Михайлівна
кафедра КІТ, ДНУЗТ 2020

Рисунок 3.1 – Маркувальний штамп

3.7 Вимоги до транспортування і зберігання

Транспортувати програмний продукт можна наступними способами:

– через всесвітню систему взаємосполучених комп'ютерних мереж, що базуються на комплекті Інтернет-протоколів – Інтернет;
– за допомогою портативних носіїв інформації – CD/DVD-дисків або USB-накопичувачів.

Термін збереження обумовлений збереженням інформації на носії.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації має входити технічне завдання та робочий проект.

До складу робочого проекту мають входити:

– специфікація;

– текст програми;

– опис програми;

– керівництво користувача. Керівництво з виокремлення блискавок;

– керівництво користувача. Керівництво з моделювання блискавок типу хмара-земля.

Вся документація до програми повинна задовольняти вимогам державного стандарту до оформлення програмних документів [3].

5 ТЕХНІКО-ЕКОНОМІЧНІ ПОКАЗНИКИ

Основна мета розробки техніко-економічного обґрунтування (ТЕО) – дати фінансову оцінку передбачуваних витрат та одержуваного корисного результату, а також оцінити прибутковість проекту і, в кінцевому підсумку, економічну доцільність його розробки та впровадження.

Початковим етапом розрахунку величини трудових витрат розробників є оцінка розміру програмного забезпечення. Основні відмінності методик, що застосовуються в оцінці трудовитрат, полягають у використовуваному типі критерію оцінки якості (кількісний або якісний) [4].

Згідно моделі COCOMO, розмір проекту S вимірюється в рядках коду LOC (KLOC), а трудовитрати в людино-місяцях.

$$E = a \cdot S^b \cdot EAF \tag{5.1}$$

де E – витрати праці на проект (в людино-місяцях);

S^b – розмір коду (в KLOC);

EAF – фактор уточнення витрат (effort adjustment factor).

Для простих систем, a = 2,4; b = 1,05

Припустимо, що розмір програмного коду програмного засобу – 2300 рядків, оскільки розроблювана програма містить багато розрахунків та повинна мати графічний інтерфейс користувача:

$$E = 2{,}4 \cdot 2{,}3^{1{,}05} \cdot 1 = 5{,}75 \tag{5.2}$$

Отже, згідно моделі COCOMO, орієнтовні трудовитрати на проект складуть приблизно 5,75 людино-місяці.

Нижче наведені розрахунки вартості розробки «Автоматизована система конструктивно-продукційного моделювання молнієвої активності». Основними статтями витрат прийняті:

– основна заробітна плата;

– відрахування на соціальні потреби;

– накладні витрати;

– витрати на персональний комп’ютер і ліцензійні базові програмні засоби.

Основна заробітна плата (ОЗП) оцінює працю інженера-програміста зі створення програмного продукту і визначається виходячи з кількості розробників, часу виконання розробки (годин), а також заробітної плати в розрахунку на одну годину. Розрахунок заробітної платні проводиться по формі табл. 5.1.

Таблиця 5.1 – Фонд місячної заробітної плати

№ п/п	Посада виконавця	Оклад, грн/міс	Кількість		Сума зарплати, грн
			осіб	місяців	
1	інженер-програміст	19634 [5]	1	6	117 804

Описаний в проекті програмний продукт буде розроблений одним програмістом в період з 3.02.20 до 24.07.20, що складає 119 робочих днів з розрахунку (20 – у лютому, 21 – у березні, 21 – у квітні, 19 – у травні, 20 – у червні та 18 – у липні) та 6 вихідних днів (1 – у березні, 1 – у квітні, 2 – у травні, 2 – у червні) або 25 робочих тижнів. Витрати робочого часу прийняті за 40 годин на тиждень, оплата за святкові дні нараховується як за робочий день. Погодинна ставка кваліфікованого інженера–програміста складає 118,73 грн/год [6]. Таким чином, витрачено робочого часу:

$$t_{розробки} = N_{чол} \cdot N_{тиж} \cdot N_{год}\,, \tag{5.3}$$

де $N_{чол}$ – кількість виконавців, *чол*;

$N_{тиж}$ – тривалість розробки;

$N_{год}$ – витрати робочого часу, год;

$$t_{розробки} = 1 \cdot 25 \cdot 40 = 1000 \text{ чол/год.} \tag{5.4}$$

ОЗП визначається за формулою:

$$ОЗП = t_{розробки} \cdot N \cdot K_{КВ}\,, \tag{5.5}$$

Де $t_{розробки}$ – витрати праці у чол/год*;*

N – погодинна ставка;

$K_{КВ}$ – коефіцієнт кваліфікації програміста, приймається 0,75.

ОЗП складає:

$$ОЗП = 1000 \cdot 118{,}73 \cdot 0{,}75 = 89\,048 \text{ грн.} \quad (5.6)$$

Відрахування на соціальні потреби встановлюються у відсотках від суми заробітної плати та на сьогоднішній день складає 22% [7]:

$$C_{соц} = \frac{89\,048 \cdot 22\,\%}{100\,\%} = 19\,561 \text{ грн.} \quad (5.7)$$

Отримані результати за (6) та (7) підсумовуються. Вони складають 108 639 грн. та визначають основні прямі витрати.

Накладні витрати враховують загальногосподарчі витрати по забезпеченню проведення роботи: витрати на опалення, електроенергію, амортизація будівель, зарплату адміністративного персоналу та інше. Вони визначаються в процентах (30 – 40%) від суми прямих витрат:

$$C_{накл} = \frac{(ОЗП + C_{соц}) \cdot 35\,\%}{100\,\%}; \quad (5.8)$$

$$C_{накл} = \frac{108\,639 \cdot 35\,\%}{100\,\%} = 38\,024 \text{ грн.} \quad (5.9)$$

На протязі усього терміну використання нової техніки підприємство щорічно витрачає певні кошти, пов’язані з її експлуатацією.

Експлуатаційні витрати на персональний комп’ютер визначаються протягом терміну розробки програмного засобу в залежності від вартості комп’ютеру. В експлуатаційні витрати входять:

– вартість витратних матеріалів;

– витрати на ремонт;

– заробітна плата ремонтника;

– оренда приміщення;

– додаткові витрати – прибирання приміщення, охорона, оренда, комунальні послуги;

– амортизаційні витрати на персональний комп'ютер і програмне забезпечення.

Витрати на електроенергію ($C_{ел}$) визначаються за формулою:

$$C_{ел} = P \cdot B \cdot T_{розр}, \quad (5.10)$$

де P – потужність комп’ютера та допоміжних споживачів електричної енергії, приймається 0,45 кВт/год;

B – вартість 1 кВт/година, складає 2,73 грн [8];

$T_{розр}$ – час роботи з ЕВМ, приймається рівним робочому часу.

Витрати на електроенергію визначаються так:

$$C_{ел} = 0{,}45 \cdot 2{,}73 \cdot 1000 = 1229 \text{ грн.} \quad (5.11)$$

Витрати на матеріали ($C_{вм}$) протягом всього терміну експлуатації приблизно 10% від вартості комп’ютеру. Вартість робочої станції приймається 22 000 грн. [9], термін експлуатації – 5 років. Отже, можна визначити ці витрати за період створення програмного засобу:

$$C_{вм} = B_{ком} \cdot \frac{N_Д}{N_{експ} \cdot 365} \cdot \frac{10\%}{100\%} \ , \quad (5.12)$$

де $B_{КОМ}$ – вартість персонального комп’ютеру;

$N_Д$ – кількість днів розробки програмного продукту;

$N_{експ}$ – термін експлуатації персонального комп’ютеру.

Витрати на витратні матеріали визначаються так:

$$C_{вм} = 22\,000 \cdot \frac{119}{5 \cdot 365} \cdot \frac{10}{100} = 143 \text{ грн.} \quad (5.13)$$

Заробітна плата ремонтника ($C_{рем}$) визначена наступним чином: на ремонт 50 комп’ютерів потрібен один інженер-системотехнік. Його середньомісячна заробітна плата приймається 12000 грн. [10] Тоді в перерахунку на один комп’ютер його заробітна плата складає:

$$C_{рем} = \frac{C'_{рем}}{N_{КОМ}} \cdot T_{міс} \ , \quad (5.14)$$

де $C'_{рем}$ – середньомісячна заробітна плата;

$N_{КОМ}$ – кількість комп’ютерів на одного ремонтника;

$T_{міс}$ – час розробки програмного продукту, міс.

Заробітна плата ремонтника ($C_{рем}$) буде складати:

$$C_{рем} = \frac{12000}{50} \cdot 6 = 1440 \text{ грн.} \quad (5.15)$$

За статистикою витрати на комплектуючи вироби ($C_{КОМ}$) для ремонту персонального комп'ютера складає 10% від його вартості за термін його експлуатації, тобто рівні витратам на витратні матеріали:

$$C_{КОМ} = C_{ВМ} = 143 \text{ грн.} \quad (5.16)$$

Амортизаційні відрахування на персональний комп'ютер (АПК) визначені з положення, що амортизаційний період в даний час дорівнює терміну морального старіння обчислювальної техніки і складає 3 роки. Отже, за 3 роки амортизаційні відрахування на персональний комп'ютер дорівнюють вартості комп'ютера. За період проектування амортизаційні відрахування складуть:

$$АКП = B_{КОМ} \cdot \frac{N_Д}{N_{експ} \cdot 365} = 22000 \cdot \frac{119}{3 * 365} = 2391 \text{ грн.} \quad (5.17)$$

Амортизаційні відрахування на програмне забезпечення (АПЗ) залежать від його циклу заміни. Якщо прийняти термін морального старіння таким же, як у персонального комп'ютера, то амортизаційні відрахування на програмне забезпечення за 3 роки дорівнюють його вартості.

Для функціонування ПК використовувалася операційна система «Ubuntu», для написання програми – «Qt Creator». Розрахунок АПЗ зведений в табл. 5.2

Таблиця 5.2 – Використовуване програмне забезпечення

Найменування програмного забезпечення	Вартість програмного забезпечення, грн	Джерело придбання	Амортизаційні відрахування, грн
«Fedora Linux»	0	fedoraproject.org	0
«Qt Creator»	0	qt.io	0
Всього:			0

Додаткові витрати ($C_{дод}$): прибирання приміщень, охорона, оренда, комунальні послуги важко оцінити точно і прийняти рівними 50% заробітної плати інженера–системотехніка, тобто 6000 гривень на місяць.

Оренду приміщень для однієї людини приймемо рівною 1 050 гривень на місяць (загальна вартість оренди приміщення на 10 інженер-програмістів 10 500 грн. на місяць). Тобто за весь період розробки – 6 300 грн.

Сумарні експлуатаційні витрати на один персональний комп’ютер складають:

$$C_{експ} = C_{ел} + C_{ВМ} + C_{рем} + C_{ком} + АПК + АПЗ + C_{ор} + C_{дод} \ ; \quad (5.18)$$

$$C_{експ} = 1229 + 143 + 1440 + 143 + 2391 + 0 + 6\,300 + 6\,000 = 17\,646 \text{ грн.} \quad (5.19)$$

Результати розрахунків зведено у табл. 5.3.

Таблиця 5.3 – Експлуатаційні витрати на ПК і ПЗ.

Найменування витрат	Витрати, грн
Витрати на електроенергію	1229,00
Вартість витратних матеріалів	143,00
Витрати на ремонт	1440,00
Комплектуючи вироби	143,00
Амортизація персонального комп’ютера	2 391,00
Амортизація програмного забезпечення	0,00
Оренда приміщення	6 300,00
Додаткові витрати	6 000,00
Всього	17 646,00

Таким чином, витрати на створення програмного продукту складають:

$$C_{розробки} = ОЗП + C_{соц} + C_{накл} + C_{експ} \ ; \quad (5.20)$$

$$C_{розробки} = 89\,048 + 19\,561 + 38\,024 + 17\,646 = 164\,279 \text{ грн.} \quad (5.21)$$

Розрахунок витрат зведено у табл. 5.4.

Таблиця 5.4 – Кошторис витрат на розробку програмного засобу

Найменування витрат	Витрати, грн
Основна заробітна плата	89 048,00
Відрахування на соціальні потреби	19 561,00
Накладні витрати	38 024,00
Експлуатаційні витрати	17 646,00
Всього	164 279,00

За отриманими значеннями техніко-економічних показників проекту складено кошторис витрат на розробку сучасного програмного забезпечення для оцінки схожості програм. За результатами розрахунків, приблизна вартість розробки складає 164 279,00 грн.

6 СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

Стадії та етапи розробки приводяться в табл. 6.1.

Таблиця 6.1 – Стадії та етапи розробки

№	Стадії розробки	Етап розробки	Термін
1	Технічне завдання	Постановка задачі	23.12.19 – 27.12.19
		Розробка структур вхідних і вихідних даних	30.12.19 – 05.01.20
		Розробка вимог до програми	06.01.20 – 24.01.20
		Затвердження технічного завдання	25.01.20 – 2.02.20
2	Робочий проект	Розробка і програмування логіки програми	3.02.20 – 15.05.20
		Розробка і програмування користувацького інтерфейсу	18.05.20 –19.06.20
		Відлагодження програми	22.06.20 – 03.07.20
		Розробка програмної документації	06.07.20 – 13.11.20
3	Впровадження	Підготовка і передача програми і програмної документації для супроводу	16.11.20 – 30.11.20

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙОМУ

Контроль здійснюється за допомогою виконання набору тестів з метою знаходження помилок в програмному продукті та його специфікації. Контроль виконання роботи забезпечується головним керівником розробки.

Прийом програмного продукту здійснюється уповноваженою комісією.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. ДСанПІН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».
2. Закон Міністерства охорони здоров'я України від 09.10.2000 № 247 (у редакції наказу МОЗ від 14.03.2006 № 120) «Про затвердження Тимчасового порядку проведення державної санітарно-гігієнічної експертизи».
3. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. – 38 с.
4. Методики оценки трудозатрат по разработке программного обеспечения информационных систем [Текст]: Є. Борисенков; ТарГУ ім. М.Х.Дулати.
5. Середня заробітна плата за видами економічної діяльності по місяцях. Архів. – Режим доступу: http://www.ukrstat.gov.ua/operativ/operativ2005/gdn/Zarp_ek_m/Zp_ek_m_u/arh_zpm_u.html. Перевірено: 14.10.2020.
6. Середня заробітна плата за відпрацьовану годину за видами економічної діяльності за квартал. Архів. – Режим доступу: http://www.ukrstat.gov.ua/operativ/operativ2018/gdn/szp_vg/szp_vg_ek/szp_vg_ek_u.htm. Перевірено: 14.10.2020.
7. Закон України «Про збір та облік єдиного внеску на загальнообов'язкове державне соціальне страхування», стаття 8, пункт 5. Режим доступу: https://zakon.rada.gov.ua/laws/show/2464-17#Text. Перевірено: 14.10.2020.
8. Офіційний сайт Національної комісії, що здійснює державне регулювання у сферах енергетики та комунальних послуг. Тарифи на електроенергію для непобутових споживачів. Режим доступу: http://www.nerc.gov.ua/. Перевірено: 14.10.2020.
9. Ноутбук HP ProBook 455 G7 (7JN01AV_V2) Pike Silver. Режим доступу: https://rozetka.com.ua/hp_7jn01av_v2/p243835963/. Перевірено: 14.10.2020.
10. Інженер комп'ютерних систем. Пошук робочих місць. Режим доступу: https://www.work.ua/ru/resumes/3794515/. Перевірено: 14.10.2020.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Дніпровського
національного університету
залізничного транспорту імені
академіка В. Лазаряна
_____ Б. Є. Боднар
«3» грудня 2020 р.

АВТОМАТИЗОВАНА СИСТЕМА КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

Робочий проект

ЛИСТ ЗАТВЕРДЖЕННЯ

Завідувач кафедри КІТ
_____ В. І. Шинкаренко
«3» грудня 2020 р.
Керівник розробки
_____ В. І. Шинкаренко
«3» грудня 2020 р.
Виконавець
_____ І. М. Нікітіна
«3» грудня 2020 р.
Нормоконтролер
_____ О. С. Куроп'ятник
«3» грудня 2020 р.

2020

ЗАТВЕРДЖЕНО

01116130.01188-01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

Специфікація

01116130.01188-01

Листів 2

2020

Позначення	Найменування	Примітка
	Документація	
01116130.01188-01-ЛЗ	Лист затвердження	
01116130.01188-01 12 01-ЛЗ	Лист затвердження	
01116130.01188-01 12 01	Текст програми	
01116130.01188-01 13 01-ЛЗ	Лист затвердження	
01116130.01188-01 13 01	Опис програми	
01116130.01188-01 ІЗ 01	Керівництво користувача. Керівництво з виокремлення блискавок	
01116130.01188-01 ІЗ 02	Керівництво користувача. Керівництво з моделювання блискавок типу хмара-земля	

ЗАТВЕРДЖЕНО

01116130.01188-01 13 01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

Опис програми

01116130.01188-01 13 01

Листів 26

2020

АНОТАЦІЯ

Документ 01116130.01188-01 13 01 «Автоматизована система конструктивно-продукційного моделювання молнієвої активності. Опис програми» входить до складу програмної документації до програми.

Програмний продукт надає можливість автоматизувати процес виокремлення блискавок на кадрах поширення грозового фронту та будувати конструктори блискавки типу хмара-земля з подальшою візуалізацією. В документі міститься опис програми та її функціональних можливостей. Програма реалізована на мові C++ у програмному середовищі Qt Creator 5.0.

ЗМІСТ

1 ЗАГАЛЬНІ ВІДОМОСТІ

Програмний продукт має назву «Автоматизована система конструктивно-продукційного моделювання молнієвої активності»; його призначено для дослідження характеру поширення грозового фронту та блискавок у атмосфері.

До програмних засобів, які потрібні для функціонування даного програмного продукту, слід віднести операційну систему сімейства GNU/Linux 3 та вище з графічним інтерфейсом.

Програма реалізована на мові C++ у програмному середовищі QT Creator 5.0.

2 ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Функціональним призначенням розробки є:

– виокремлення блискавок з відеофайлів, створених метрологічними супутниками, які демонструють процес поширення грозового фронту;

– розрахунок заданих параметрів виокремлених блискавок;

– моделювання поширення блискавки типу хмара-земля у атмосфері з подальшою візуалізацією.

Програма не призначена для роботи з відеофайлами, що були записані камерою, що рухається.

3 ОПИС ЛОГІЧНОЇ СТРУКТУРИ

3.1 Алгоритм програми

Алгоритми організації роботи програми з користувачем є очевидним і зрозумілим. Щоб зрозуміти логічну структуру програми доцільно розглянути основні алгоритми роботи програми – процес отримання кадрів з виокремленими блискавками з кадрів відео, створених метеосупутниками та алгоритм візуалізації моделі з заданих правил конструкції.

3.1.1 Процес отримання кадрів з виокремленими блискавками з кадрів відео

Задана послідовність дій: видалення колірного каналу, видалення фону, видалення шумів, заповнення пустот, видалення шумів, пошук блискавок. Послідовність змінюватись не може, можливо лише вмикати або вимикати дану операцію над зображенням. Схему обробки послідовності дій показано на рис. 3.1.

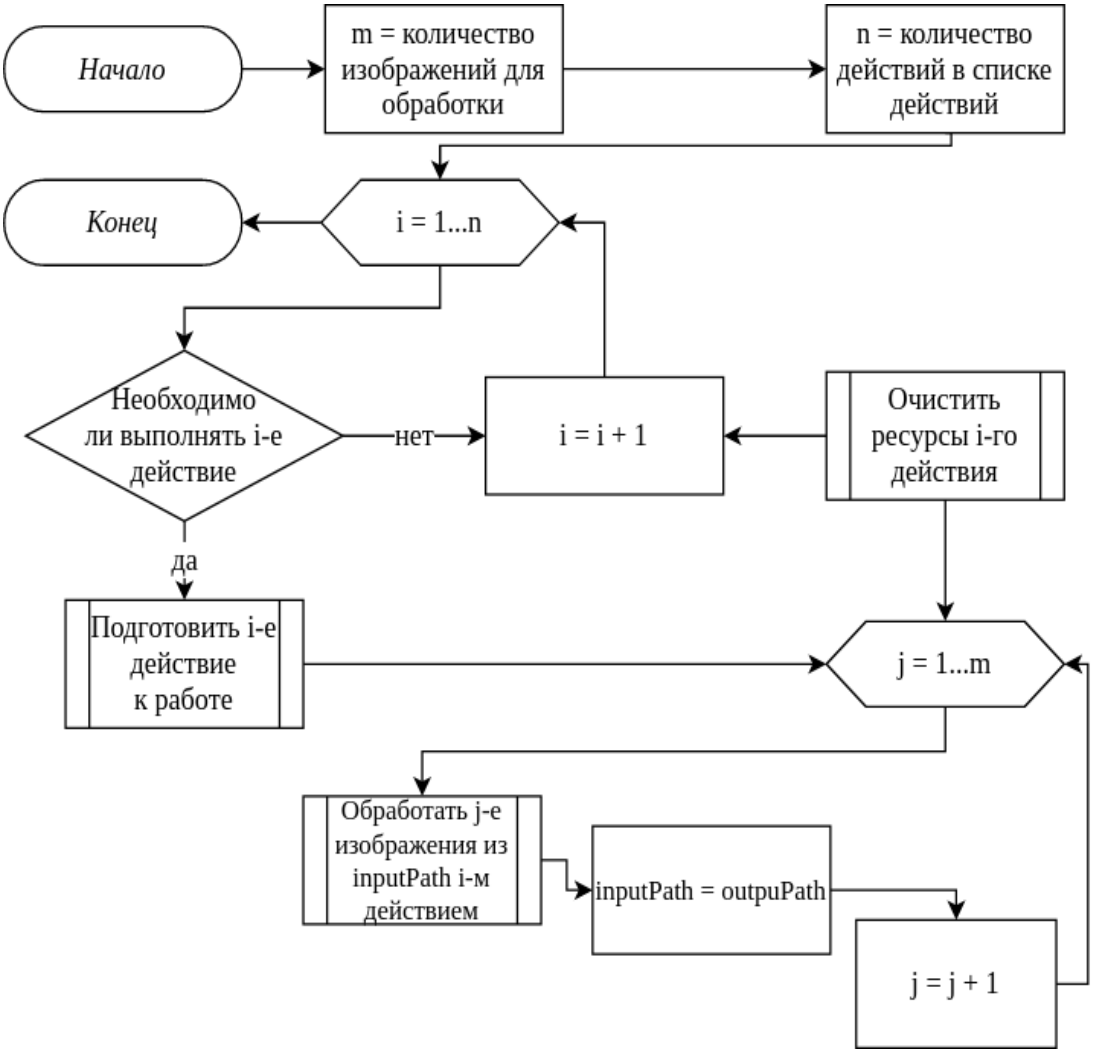

Рисунок 3.1 – Алгоритм обробки кадрів

Підготовка і-ї дії до роботи та чистка залежить від типу дії та явно визначена лише для пошуку блискавок, що являє собою відкриття та закриття потоку виводу у файл. Розглянемо алгоритм обробки j-го зображення дії видалення фону з зображення (рис. 3.2).

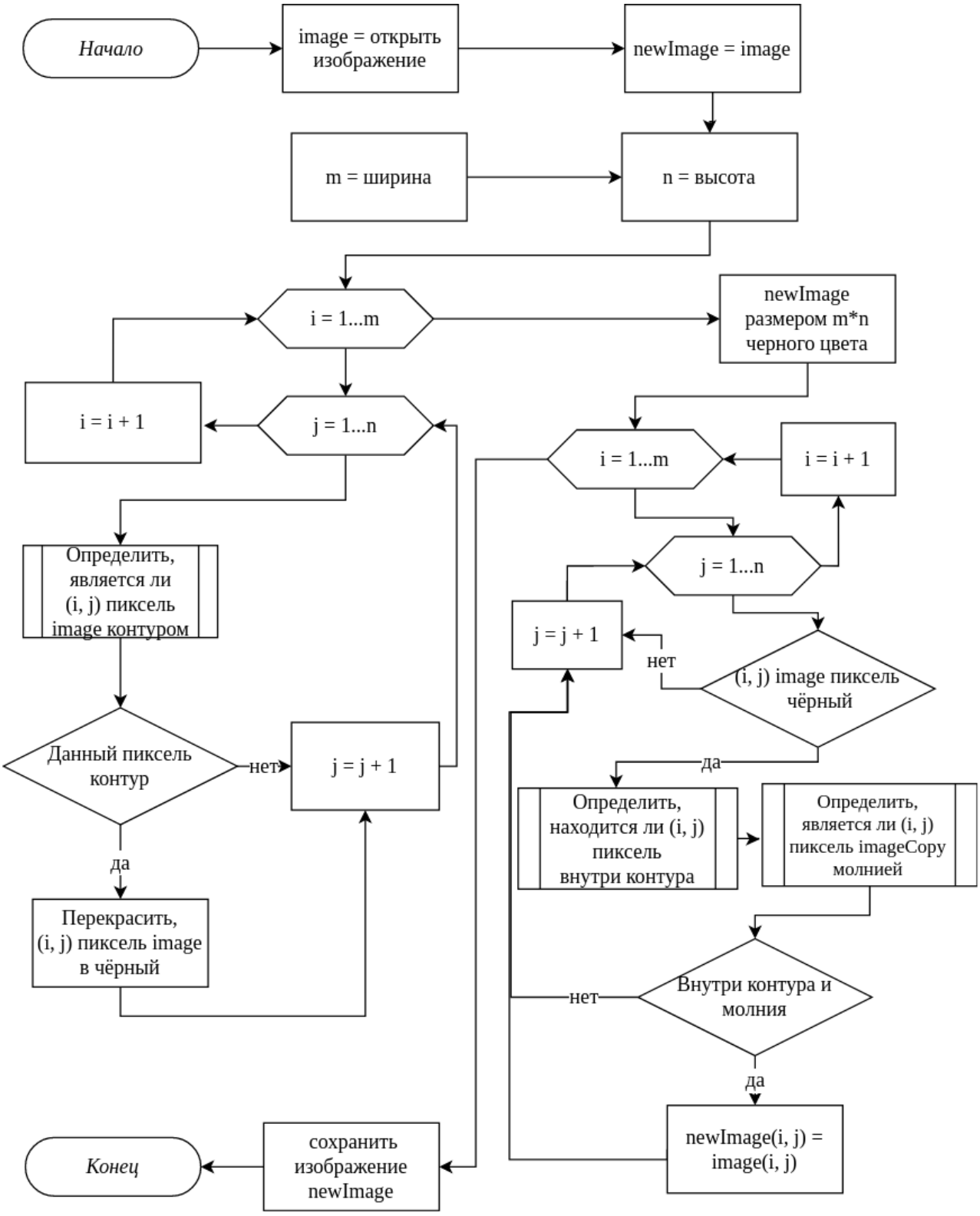

Рисунок 3.2 – Схема алгоритму видалення фону

3.1.2 Процес візуалізації моделі з заданих правил конструкції

Для відтворення моделі блискавки застосовуються наступні дії. Існує таблиця терміналів з їх правилами, один з терміналів є аксіомою. Для відтворення правила для візуалізації використовується алгоритм, схема якого представлена на рис. 3.3.

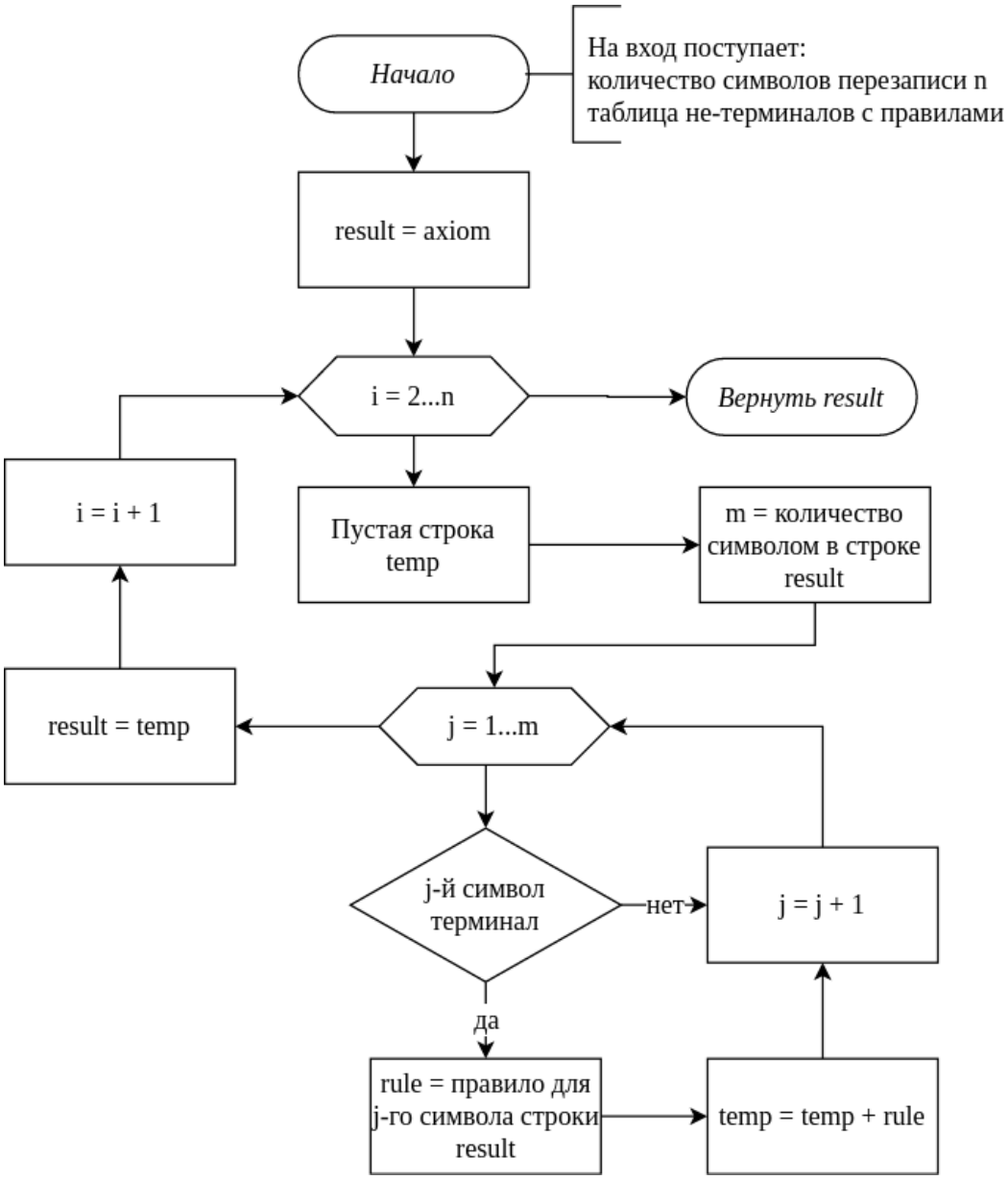

Рисунок 3.3 – Алгоритм відтворення правила

3.2 Використані методи

Програма використовує наступні методи:

– для пошуку блискавки на кадрах з видаленим фоном використовується алгоритм пошуку в глибину [1];

– для розрахунку центру блискавки обчислюється середнє за кожною координатою;

– для порівняння кольорів використовується евклідова міра схожості [2];

– заповнення пустот та видалення шумів описано у джерелі [2];

– для роботи з правилом використовується алгоритм перезапису вузлів, а для відтворення моделі за заданим правилом використовується метод черепахи [3].

3.3 Структура програми з описом функцій складових частин і зв'язки

Програма складається з наступних модулей: MainWindow, Dialog, Utils, Actions та Poductions.

Модуль MainWindow є головним модулем програми і організує взаємодію всіх інших модулів між собою. Відповідає за отримання даних від користувача (Dialogs), передачу їх до логіки програми (модуль Actions), а також демонстрацію даних користувачеві.

Модуль Dialog включає в себе різноманітні діалогові вікна для взаємодії з користувачем (прийом даних та оповіщення). Він використовує модуль Actions, оскільки отримує від користувача необхідні параметри для їх створення та, власне, створює їх.

Модуль Utils представляє собою модуль, який містить класи, що реалізують загальновживані складні логічні задачі. Дозволяє уникнути дублювання коду.

Модуль Actions представляє собою основну логіку програми. Даний модуль реалізує можливість застосовувати різноманітні методи обробки кадрів з відео, створених метеосупутниками, для пошуку потрібних даних на них а для відтворення спрощеного представлення блискавок грозового фронту. Використовує операції модуля Utils та Spots.

Spots необхідний для вибору типу спрощеного представлення блискавок. Модуль виокремлений для більш простого масштабування програми (додавання нової апроксимованої форми блискавки у грозовому фронті).

Модуль Poductions містить класи, які дозволяють будувати правила продукції блискавок типу хмара-земля і їх подальше відтворення.

На рис. 3.4 представлена схема взаємодії основних модулів програми.

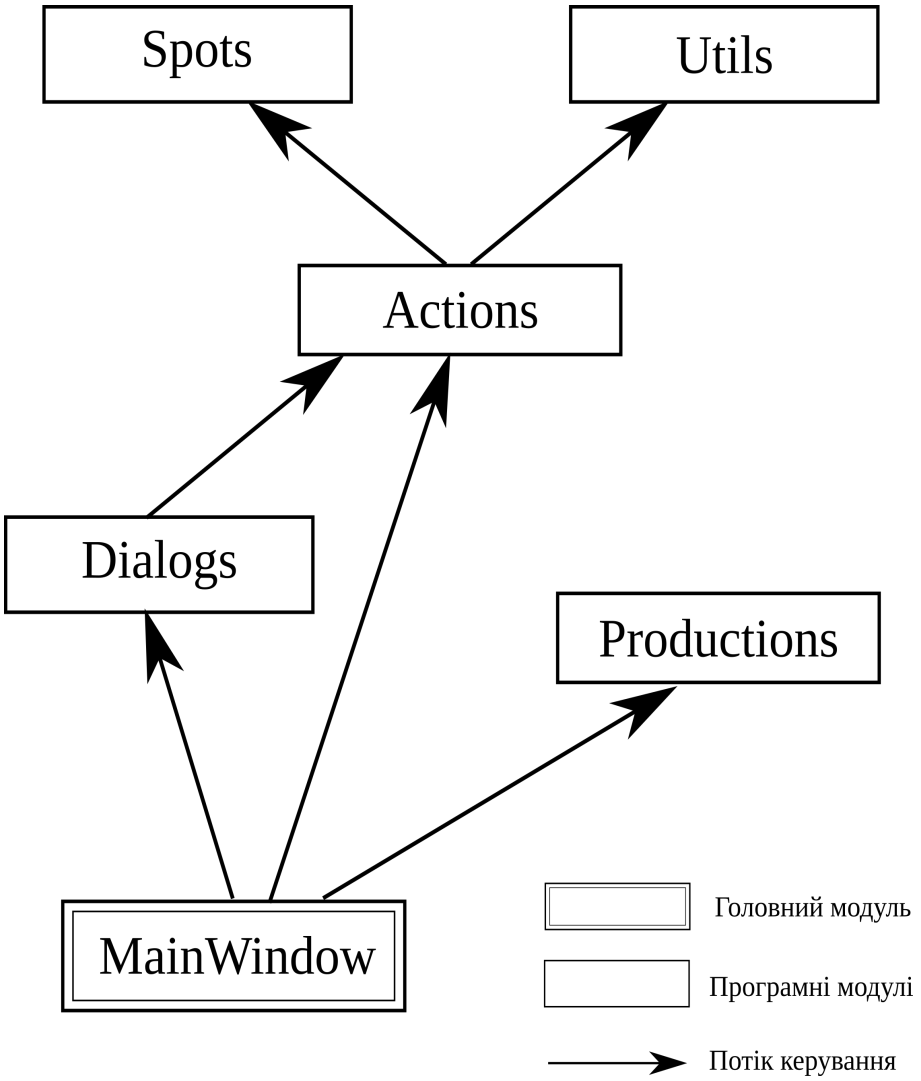

Рисунок 3.4 – Схема взаємодії основних модулів програми

Більш детальну структуру програми можна представити за допомогою діаграми класів. Діаграма класів складається з множини елементів, які в сукупності відображають декларативні знання про предметну галузь. Ці знання інтерпретуються в базових поняттях мови UML, таких як класи, інтерфейси та відношення між ними. На діаграмі класів, класи відображаються у вигляді прямокутників, у верхній частині яких відображається назва в середині атрибути, а в нижній – методи цих класів. Розглядається окремо модуль Actions. Його основу складає абстрактний базовий клас IAction, від якого наслідуються класи-дії. Клас IAction має три чистих віртуальні методи:

– process() – виконати обробку над об'єктом як того передбачає операція;

– getName() – отримати назву операції, яка непредставлена даним класом

та три віртуальних методи:

– reset() – повернути значення параметрів до значення параметрів за замовченням;

– prepare() – виконати підготовку певних параметрів, до старту циклу обробки;

– clear() – для очищення ресурсів, відкритих методом prepare().

Кількість класів обумовлена операціями, які необхідно виконувати для виокремлення блискавок та їх подальшої апроксимації:

– DeleteColorChannel – клас, що реалізує видалення колірного каналу;

– DeleteBackground – видалення фону, на якому розташовані спалахи блискавок грозового фронту;

– DeleteNoises – видалення шумів після видалення фону;

– FillEmpties – заповнення пустот, що утворились після видалення фону;

– FindSpots – пошук плям блискавок, їх апроксимація до більш простої форми та подальший розрахунок їх параметрів.

– CreateApproximateLightning – створення блискавок заданої апроксимованої форми, за отриманими раніше параметрами з подальшим розміщенням на зображеннях та їх збереженням.

Всі описані вище класи реалізують методи process() та getName(), а також перевизначають метод reset(), який викликає й метод reset() базового класу. Методи prepare() та clear() перевизначають лише класи FindSpots та CreateApproximateLightning. Дані класи використовують їх відкриття та закриття потоків роботи з файлом. Дані методи у базовому класі мають порожнє тіло.

При роботі над реалізацією методу process() класу DeleteBackground знадобилися наступні класи, які поміщені до модуля Utils:

– DecisiveFunction – клас, що реалізує визначення належності пікселя до заданого класу образів за встановленими параметрами кольору у моделях Lab або LCH та заданими правилами;

– ColorConvertor – клас, який реалізує конвертацію кольорів RGB до моделей Lab та LCH.

Діаграма класів модулю Actions та Spot представлена на рис. 3.5. Дані модулі представлені окремо, оскільки містять багато класів та зв'язків. Для того, щоб не перевантажувати діаграму класів програми представмо їх окремо.

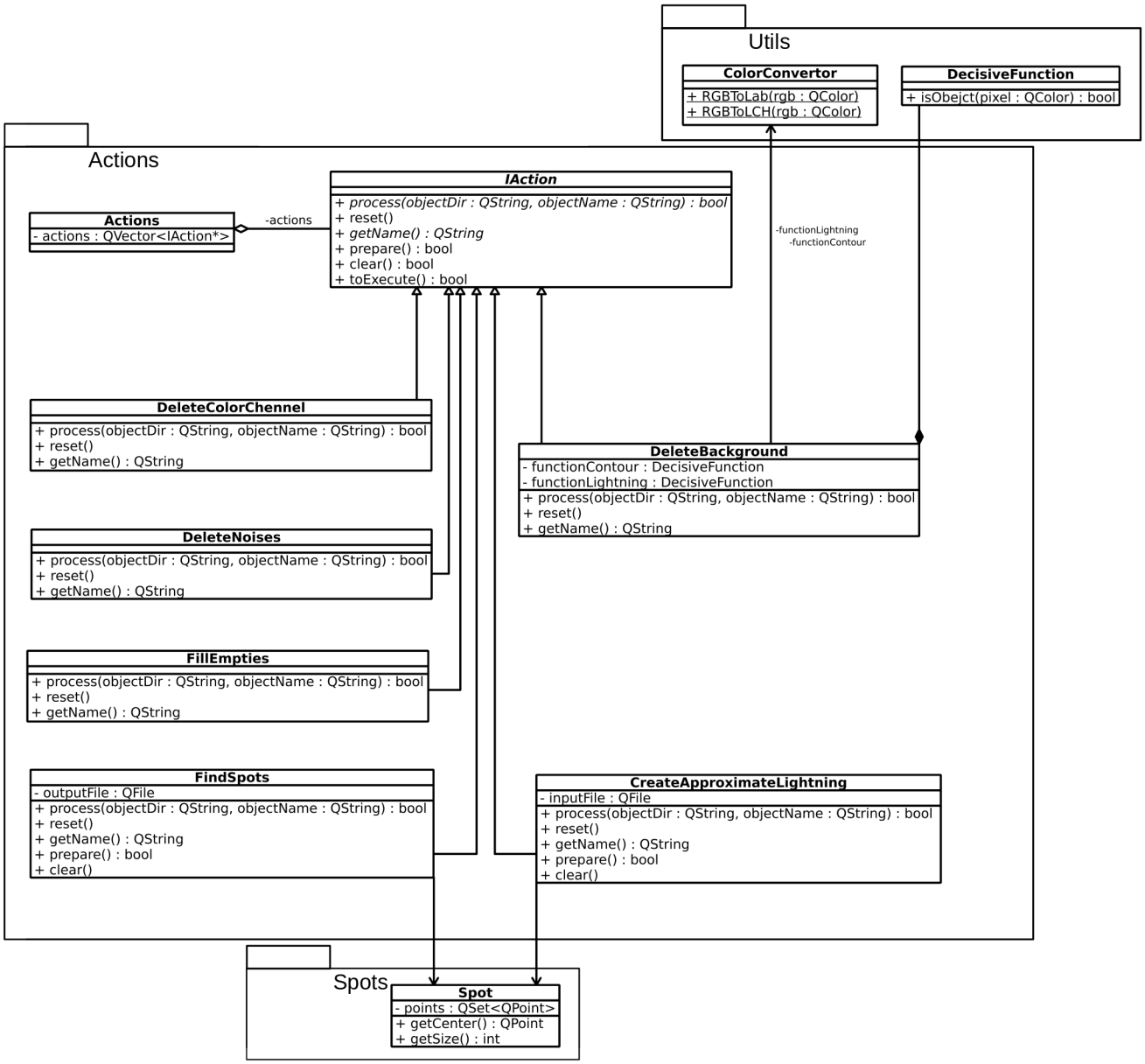

Рисунок 3.5 – Діаграма класів модулів Actions та Spot та використовувані ними класи Utils

Для приведення дій у виконання використовується клас ActionRunner. Для отримання імен зображень використовується структура Source. Основу механізму генерації імен файлів складає абстрактний базовий клас INameGenerator. Клас має дві реалізації: FixedNameGenrator та SimpleNameGenerator. Модуль Productions:

– ProdRules – для зберігання та оперування правилами продукцій;

– DrawableModel – для відтворення згенерованої послідовності графічним способом.

На основі описаної вище архітектури програми була побудована діаграма класів, спрощене представлення якої знаходиться на рис. 3.6.

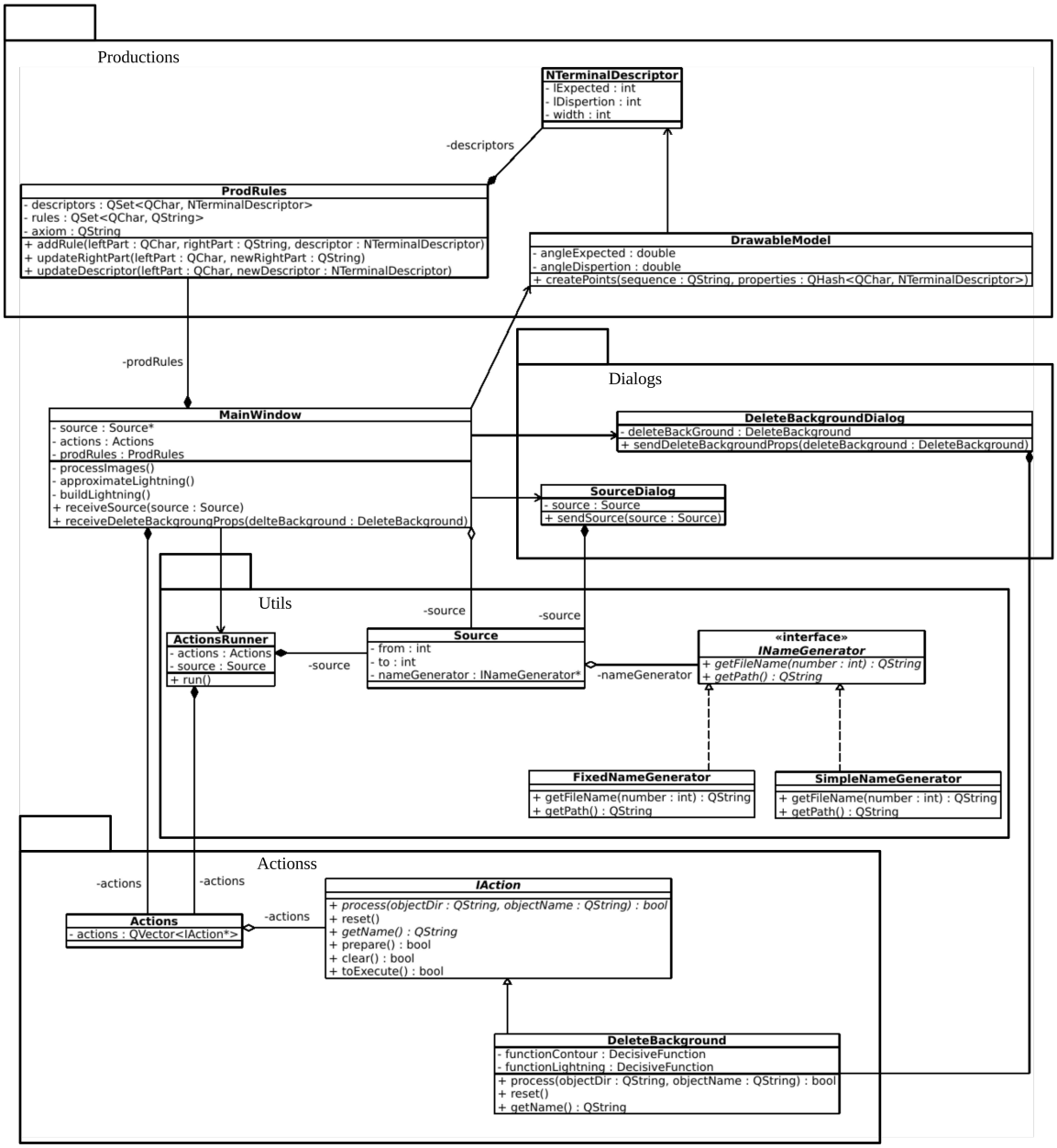

Рисунок 3.6 – Спрощена діаграма класів програми

3.4 Зв'язки програми з іншими програмами

Розроблюваний програмний продукт не має залежностей від сторонніх програми та може працювати автономно, використовуючи лише функціонал використовуваної ОС.

4 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Для експлуатації програмного продукту необхідний будь-який пристрій з параметрами, які задовольняють нормальному функціонуванню операційної системи GNU/Linux версії 3.0 або вище та має графічний інтерфейс.

Мінімальна конфігурація пристрою, що забезпечить нормальну роботу програмного продукту:

– щонайменше 128 Мб вільного місця на вбудованому носії інформації;

– щонайменше 1024 Мб оперативної пам’яті;

– монітор з роздільною здатністю 800*600 і вище;

– центральний процесор з тактовою частотою 1 ГГц та більше;

– маніпулятор «миша»;

– стандартна клавіатура;

– для встановлення ПЗ необхідна або наявність CD/DVD приводу чи USB роз’єму, або підключення до мережі Інтернет.

5 ВИКЛИК ТА ЗАВАНТАЖЕННЯ

Для запуску програмного продукту «Автоматизована система конструктивно-продукційного моделювання молнієвої активності» необхідно виконати запуск файлу DynamicProcessor. Після запуску програми користувачеві представляється головне вікно програми (рис. 5.1).

Об’єм програми у неархівованому вигляді складає 8.7 Мбайт. Програма призначена для функціонування у середовищі GNU/Linux версії 3.0 та вище.

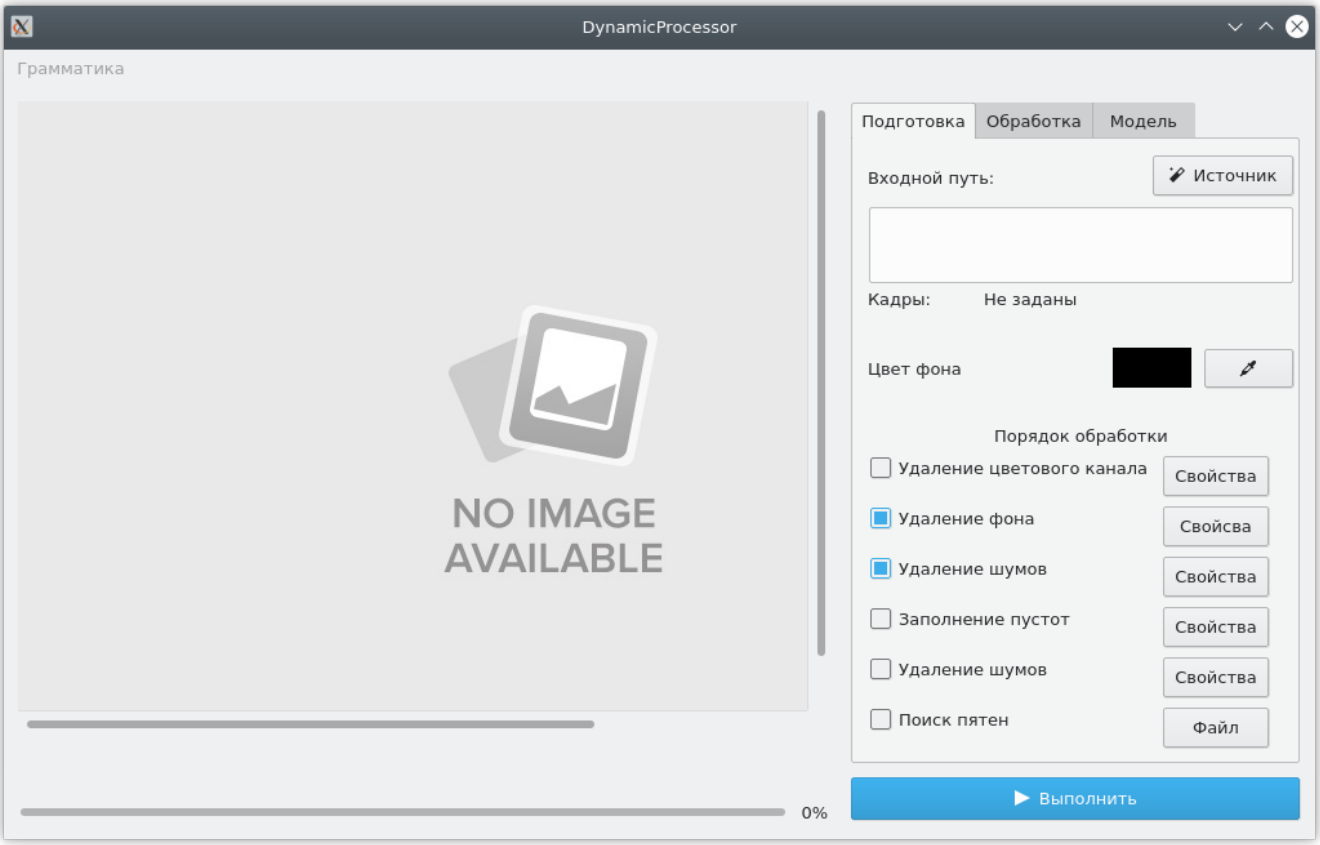

Рисунок 5.1 – Головне вікно програми

6 ВХІДНІ ДАНІ

Вхідними даними програми є:

– кадри відео, відзнятого супутником у форматі .jpg або .png, що показують процес поширення грозового фронту та мають статичний фон [4] або, у разі динамічного фону, демонструють блискавки у вигляді яскравої плями з чітким кольоровим ореолом [5, 6];

– параметри обробки відео, отримані від користувача у діалоговому режимі:

  – ознака виконання дії, що включається відповідним прапором;

  – колір фону у форматі RGB, який обирається через стандартне вікно вибору кольору;

  – рівень чутливості до різниці кольорів у відсотках для «Видалення фону», беззнакове ціле;

  – опція пошуку контуру, блискавок та виконання статичного порівняння кольорів;

  – лінійна чи квадратична функція для оцінки контуру та блискавки для «Видалення фону», включаються радіокнопками;

  – кількість циклів проходження по зображенню для «Усунення шумів», беззнакове ціле;

  – колір для пікселя у форматі RGB, який обирається через стандартне вікно вибору кольору для «Добудова пошкоджених частин»;

  – назва каналу, який необхідно видалити; текстове значення, яке обирається зі списку;

– назва директорії для збереження результату виокремлення блискавок;

– назва файлу для збереження результату розрахунку параметрів блискавок;

– параметри обробки файлу з виокремленими параметрами блискавок:

  – назва файлу з розрахунком параметрів блискавок;

  – шлях до директорії збереження результату;

  – кількість кадрів, які необхідно пропустити з початку, беззнакове ціле;

  – кількість вхідних кадрів, які необхідно розмістити на результатному, беззнакове ціле;

– через яку кількість кадрів змінювати колір;

– правила продукції для моделювання блискавки типу хмара-земля:

– математичне очікування кута повороту, дійсне число подвійної точності;

– дисперсія кута повороту, дійсне число подвійної точності;

– список правил продукції;

– не-термінал, великий символ латинського алфавіту;

– ліва частина правил, рядок, що складається з не-терміналів;

– математичне очікування довжини, беззнакове ціле;

– дисперсія кута повороту, беззнакове ціле;

– ширина 0-го рівня графічного представлення.

7 ВИХІДНІ ДАНІ

Результатом роботи програми є наступні вихідні дані:

– кадри з виокремленими блискавками з вхідних зображень зображень з форматом та ім'ям, які відповідають вхідним;

– файл формату .txt з розрахованими параметрами блискавок для серії оброблених кадрів, де параметри кожної блискавки є рядок файлу з наступними даними, розділеними пробілом: координати виникнення блискавки відносно верхнього лівого кута вхідного кадру (x та y) у пікселях, радіус блискавки у пікселях та порядковий номер кадру оброблюваної блискавки у серії, що оброблюється; рядки впорядковані за номером кадру у порядку зростання; порядок блискавок з однаковим номером неважливий;

– кадри, на які нанесені блискавки апроксимовані до кола згідно з розрахованими параметрами;

– зображення блискавки типу хмара-земля, сконструйоване відповідно до розроблених правил продукції, виведене у відведене вікно у програми.

8 ОПИС ПРИЗНАЧЕНОГО ДЛЯ КОРИСТУВАЧА ІНТЕРФЕЙСУ

На рис. 5.1 представлений зовнішній вигляд програми після відкриття. Для того, щоб отримати файл з розрахованими параметрами блискавки з вхідної послідовності, необхідно вказати параметри джерела. Кнопка «Джерело» переводить користувача на нове діалогове вікно (рис. 8.1), де можна вказати наступні параметри:

– шлях до директорії;

– формат імені файлу, який складається з префіксу, формату номеру та розширення зображень (.jpg або .png);

– номера початкового та кінцевого зображень.

Обрані параметри зберігаються у пам’яті та підставляються у поля форми при наступних запусках.

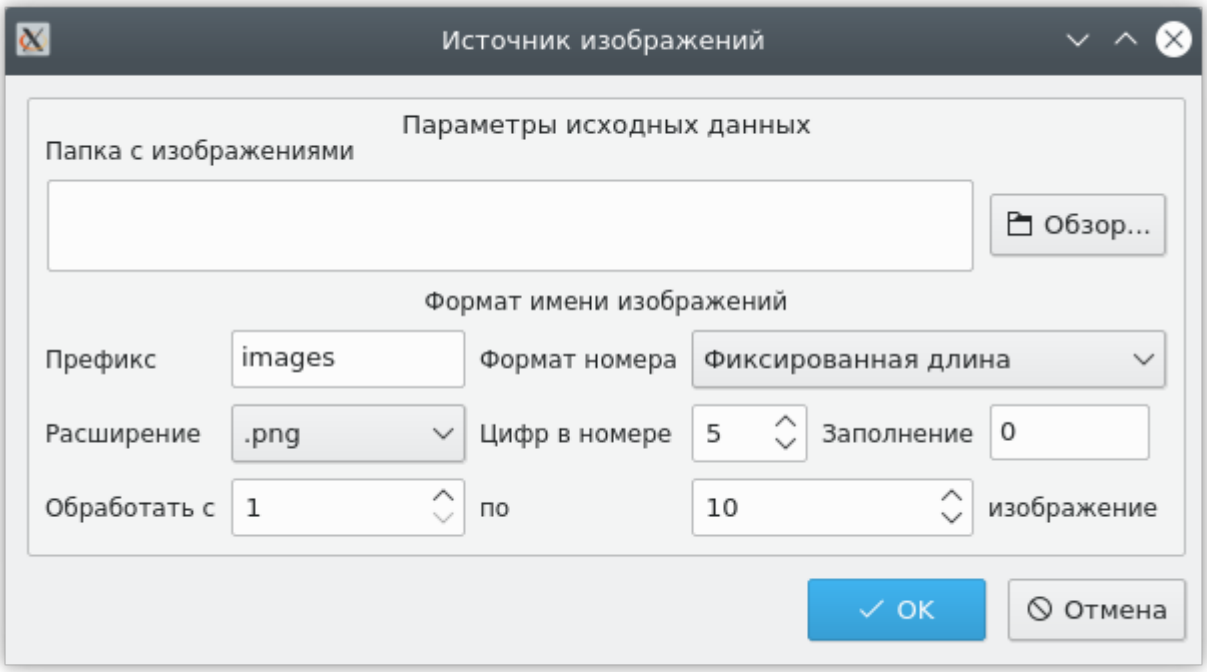

Рисунок 8.1 – Параметри джерела

Якщо користувач вказав параметри джерела коректно та натиснув на кнопку «OK», вікно закривається, а на головній формі програми з’являються введені параметри та відкриється перше зображення директорії (рис. 8.2).

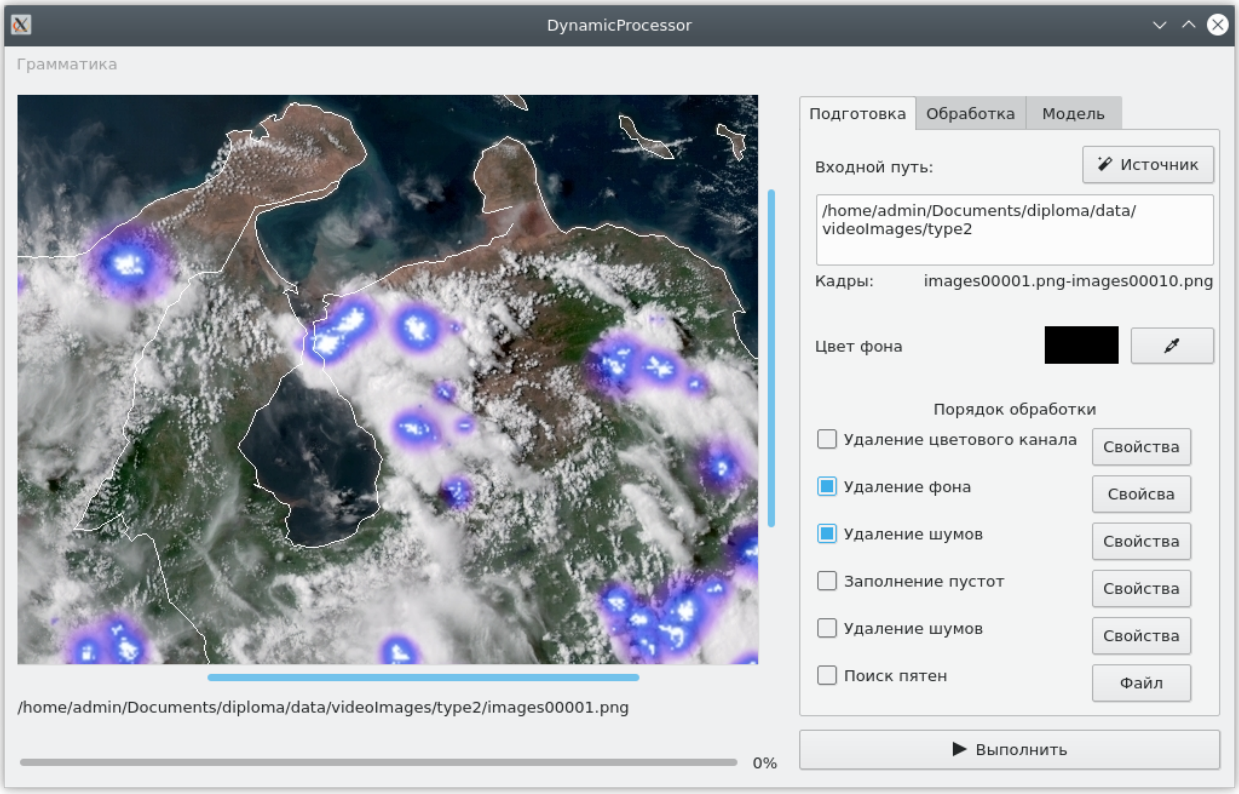

Рисунок 8.2 – Головне вікно після вибору параметрів джерела

Далі можливо обрати фон, який буде фоном для зображень з виокремленими блискавками, а також фоном вхідних зображень для дії «Пошук плям» шляхом натискання на кнопку зі значком піпетки.

Відмічені прапорцем дії зі списку будуть виконані над зображеннями. Для досягнення бажаного результату необхідно задати їх параметри при натисканні на кнопку поруч.

Видалення колірного каналу являє собою позбавлення кожного пікселя певних складових у форматі RGB. Користувачу пропонується зробити вибір із списку серед запропонованих варіантів: червоний, синій, зелений, червоний та зелений, червоний та синій, синій та зелений.

Видалення фону являє собою основну операцію з виокремлення блискавок. Видалення колірного каналу виконує роль попередньої обробки для нього, а видалення шумів та заповнення пустот необхідні для досягнення більш якісного результату. При натисненні на кнопку «Властивості» відкриється діалогове вікно (рис. 8.3), на якому розміщенні поля для вводу параметрів для видалення фону.

За допомогою прапорців можна обрати необхідні операції видалення фону.

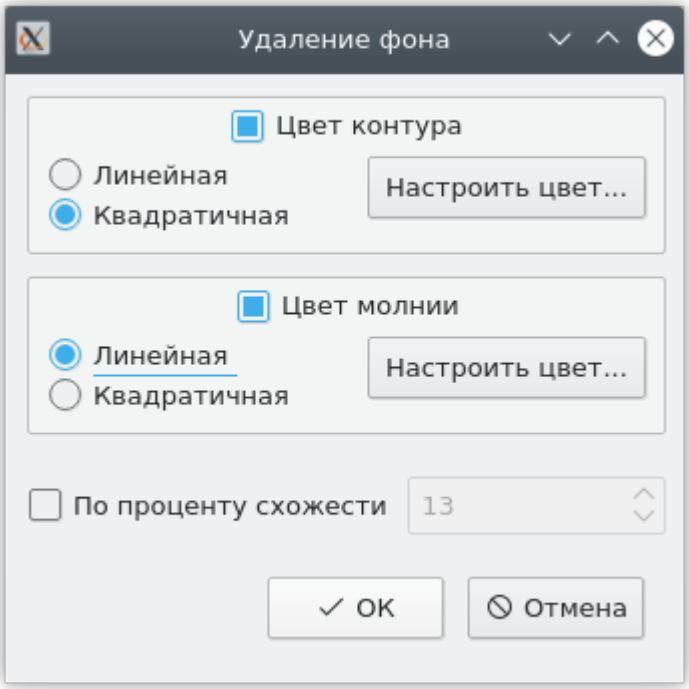

Рисунок 8.3 – Параметри видалення фону

Натискання на кнопку «Налаштувати колір...» переводить на форму (8.4).

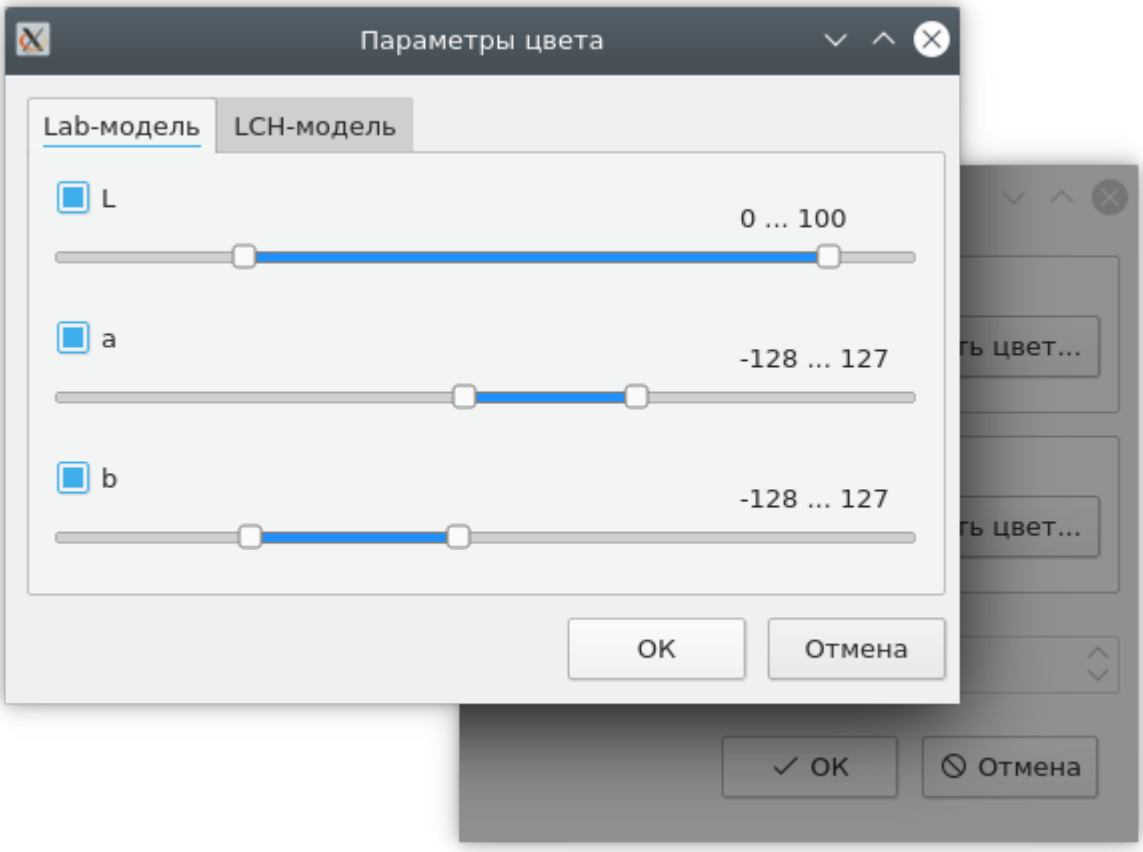

Рисунок 8.4 – Параметри колірних інтервалів

Якщо після отримання виокремлених блискавок ще й необхідно отримати їх параметри, то необхідно підняти прапорець пошуку плям та обрати файл для збереження результатів. Слід відмітити, що ви можете виконати дану дію окремо від безпосередньо виокремлення, якщо необхідні кадри є в наявності.

При натисканні на кнопку «Виконати», почнеться процес обробки. Програма буде інформувати яка дія зараз виконується та скільки кадрів уже оброблено. Кадри з виокремленими блискавки з'являються в обраній директорії та показуються в якості зображення на формі. Файл з розрахованими параметрами у кожному рядку містить такі дані: номер кадру, координата центру (x та y) та радіус даної блискавки.

Для обробки файлу, що містить апроксимовані дані блискавок, призначена вкладку «Обробка» та заповніть необхідні поля (рис. 8.5).

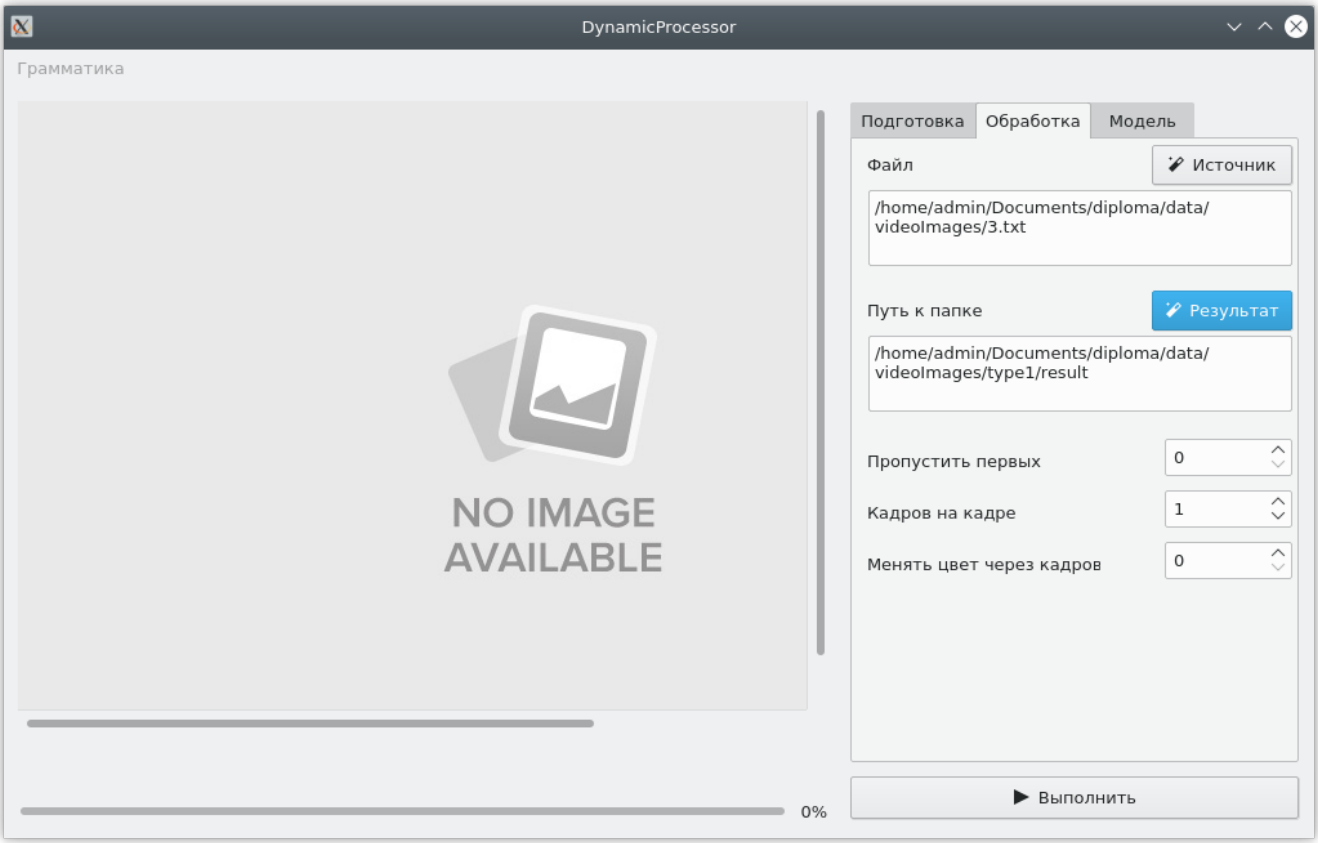

Рисунок 8.5 – Параметри відтворення апроксимованих блискавок

Поле «Пропустити перших» показує скільки кадрів необхідно пропустити від початку файлу, перш ніж почати наносити кадри на зображення у полі. Поле «Кадрів на кадрі» показує яка кількість кадрів з файлу буде розміщена на одному вихідному зображені. Кнопка «Виконати» запускає процес обробки.

Перехід на вкладку «Модель» (рис. 8.6) забезпечує інтерфейсом для будування правил продукції та їх графічного відтворення.

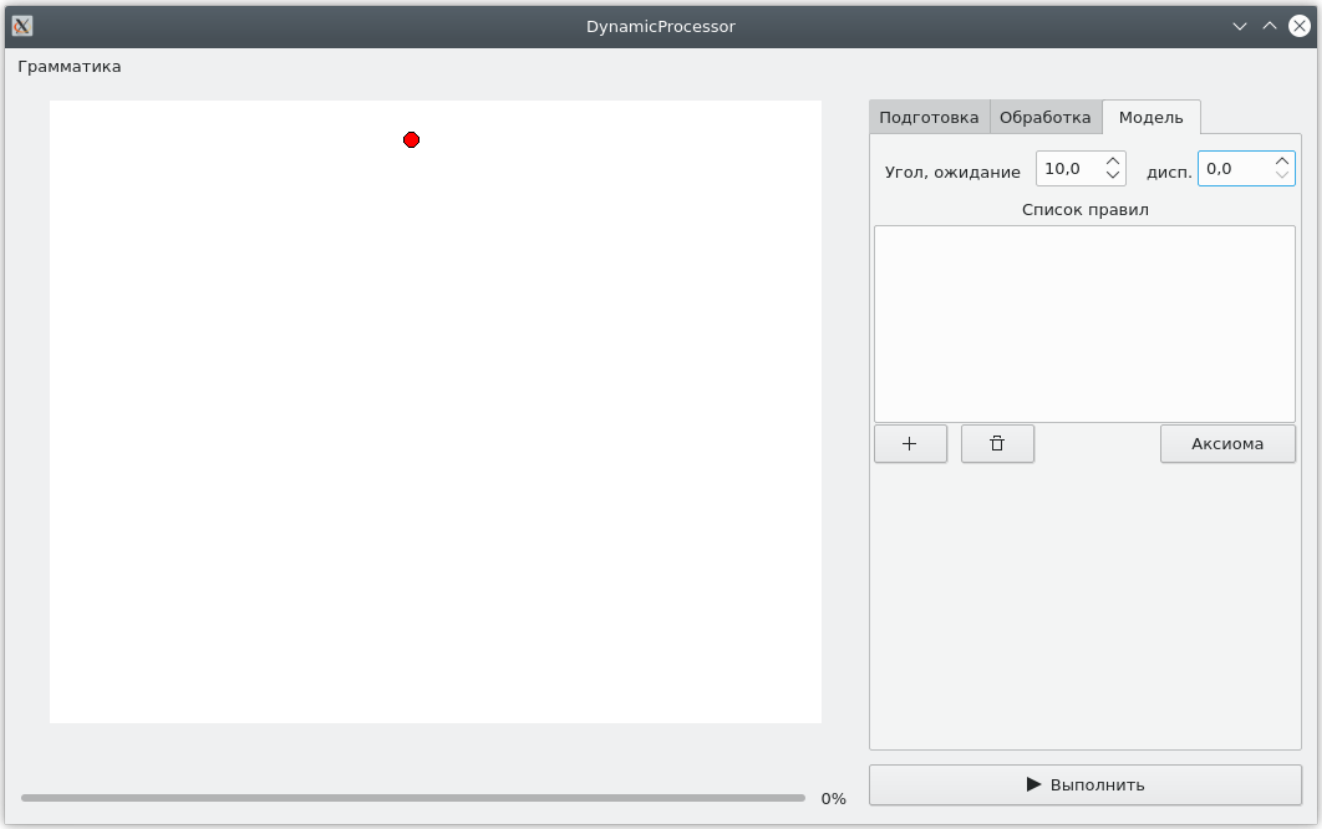

Рисунок 8.6 – Інтерфейс вкладки «Модель»

Для побудови графічного представлення блискавки необхідно створити список правил продукції (за допомогою кнопки зі знаком «Плюс» можна додати новий не-термінал) визначити його параметри(математичне очікування та дисперсію довжини, товщину ліній графічного представлення не-терміналу та, власне, праву частину правила).

Введені правила можна зберігати та потім завантажувати у програму. Для цього необхідно обрати пункт меню «Граматика», потім пункт «Зберегти» або «Імпортувати» та обрати файл для збереження або імпорту.

Щоб отримати графічне представлення розроблених правил продукції, необхідно натиснути на кнопку «Виконати» та ввести кількість циклу перезапису правила. Після чого згенерована послідовність буде відтворена (рис. 8.7).

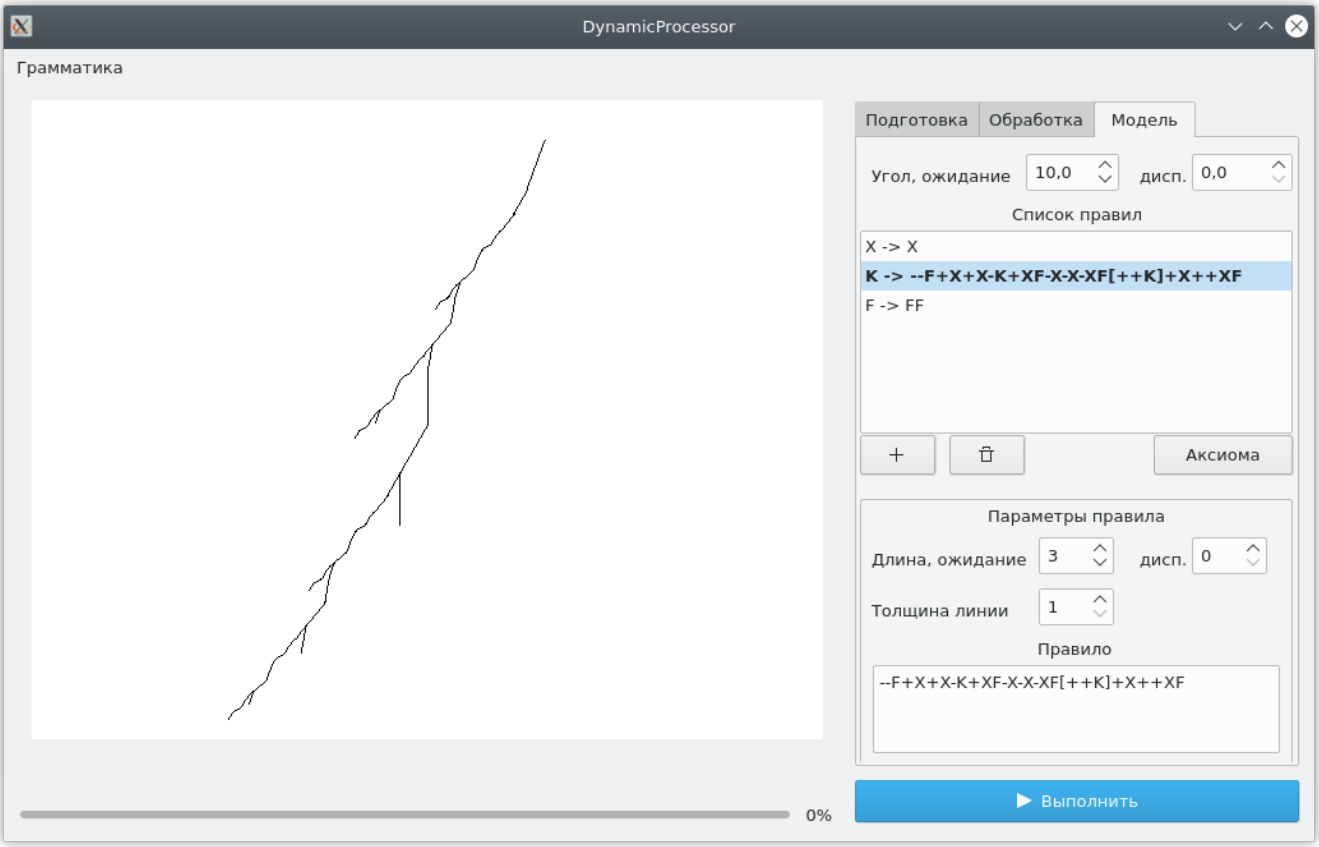

Рисунок 8.7 – Графічне представлення правил продукції

9 ПОВІДОМЛЕННЯ

У табл. 9.1 наведені повідомлення виключно користувачу, що можуть з’явитись у процесі роботи з програмою.

Таблиця 9.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Зображення <ім'я відео> не вдалось відкрити	Зображення з даним іменем не вдалось відкрити	Правильно вказуйте параметри джерела, перевірте, що зображення коректне
Вкажіть, будь ласка, <ім'я параметра>	Необхідно обрати значення відповідного параметра, наприклад, шлях до директорії, колір фону	Задайте необхідні параметри
Аксіома не обрана	Немає початкового символу для будування моделі	Оберіть аксіому відведеними засобами
Список правил відсутній	Список правил продукції не заповнений	Натисніть на кнопку додати правило та заповніть відведені поля
Даний не-термінал вже існує	Спроба створити раніше доданий не-термінал	Оберіть інший символ або змініть правило для того, що існує
Виконується дія <ім'я дії>	Програма виконує дію	Зачекайте завершення обробки
Обробка завершена успішно	Обробка відео завершена успішно	Оберіть ім'я файлу для збереження результатів
Не вдалось зберегти результати у <ім'я файлу>	Результати не можна зберегти до файлу з даним іменем	Перевірте, що даний файл доступний для запису та не використовується іншими програмами
Результати успішно збережені	Результати обробки були збережені до обраного файлу	Закрийте вікно або оберіть нові кадри для обробки

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Левитин, А. В. Метод уменьшения размера задачи: Поиск в глубину. Алгоритмы. Введение в разработку и анализ [Текст] / А. В. Левитин – М.: Вильямс, 2006. – С. 212 – 215.
2. Гонсалес, Р. Принципы распознавания образов [Текст] / Р. Гонсалес – М.: Мир, 1978. – С. 50-60, С. 103 – 104.
3. Prusinkiewicz, P. The Algorithmic Beauty of Plants [Текст] / P. Prusinkiewicz, A. Lindenmayer – New York, 1990. – 240 с.

ЗАТВЕРДЖЕНО

01116130.01188-01 І3 01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

Керівництво користувача. Керівництво з виокремлення блискавок

01116130.01188-01 І3 01

Листів 15

2020

АНОТАЦІЯ

Документ 01116130.01188-01 І3 01 «Автоматизована система конструктивно-продукційного моделювання молнієвої активності. Керівництво користувача. Керівництво з виокремлення блискавок» входить до складу програмної документації до програми.

Програмний продукт надає можливість автоматизувати процес виокремлення блискавок на кадрах поширення грозового фронту та будувати конструктори блискавки типу хмара-земля з подальшою візуалізацією. В документі міститься керівництво користувача з виокремлення блискавок.

ЗМІСТ

ВСТУП

Областю застосування програмного продукту є комп'ютерне моделювання процесів розповсюдження грозового фронту. Програмний продукт призначено для для дослідження характеру поширення грозового фронту. Функції програми полягають у виокремленні блискавок з відеофайлів, створених метрологічними супутниками, які демонструють процес поширення грозового фронту з подальшим розрахунком параметрів виокремлених блискавок (відбувається їх приведення до до кола, розраховуються координати центру відносно лівого верхнього кута та радіус у пікселях). За отриманими даними будуються нові кадри відео шляхом розміщення апроксимованих блискавок на фоні певного кольору.

Продукт розраховано на базовий рівень підготовки користувача для роботи з персональним комп’ютером.

Перед використанням продукту користувачеві необхідно ознайомитися з відповідним керівництвом.

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Функціональне призначення розробки охоплює такі аспекти:

– виокремлення блискавок з відеофайлів, створених метеорологічними супутниками, які демонструють процес поширення грозового фронту;

– розрахунок заданих параметрів виокремлених блискавок;

– створення кадрів відео, що містять апроксимоване представлення блискавок.

Експлуатаційне призначення продукту полягає в усуненні людини від задачі розрахунку параметрів блискавок з відеофайлів, створених метрологічними супутниками, в ручному режимі, що знизить час на дослідження характеру поширення грозового фронту.

Програмний продукт розрахований на персональні комп’ютери, що мають такі мінімальні характеристики:

– щонайменше 128 Мб вільного місця на вбудованому носії інформації;

– щонайменше 1024 Мб оперативної пам’яті;

– монітор з роздільною здатністю 800*600 і вище;

– центральний процесор з тактовою частотою 1 ГГц та більше;

– маніпулятор «миша»;

– стандартна клавіатура;

– для встановлення ПЗ необхідна або наявність CD/DVD приводу чи USB роз’єму, або підключення до мережі Інтернет.

Для роботи з програмним продуктом на ПК має бути встановлена ОС GNU/Linux 3.0 і вище, що має графічний інтерфейс.

2 ПІДГОТОВКА ДО РОБОТИ

Підготовка до роботи включає: ввімкнення ПК та завантаження ОС, перенесення ПЗ на робочу машину.

Для того, щоб перенести програму з носія на робочу машину, її потрібно скопіювати на вільний дисковий простір. Якщо програма представлена в стиснутому виді (дані розархівовано), то потрібно скопіювати архів у вільний дисковий простір, розархівувати на диск.

Для того, щоб запустити програмний продукт, необхідно відкрити виконуваний файл DynamicDrocessor.

Користувачеві необхідно мати директорію з розкадрованим відео-файлом, що демонструє процес поширення грозового фронту, створеним супутником NASA на зображення формату .png або .jpg.

3 ОПИС ОПЕРАЦІЙ

Робота в програмі виконується за такими напрямками:

– підготовка вхідних даних для розпізнавання об’єктів;

– розпізнавання об’єктів та запис у файл їх параметрів;

– створення нових кадрів для відео за отриманим файлом з параметрами.

На рис. 3.1 представлений зовнішній вигляд програми після відкриття.

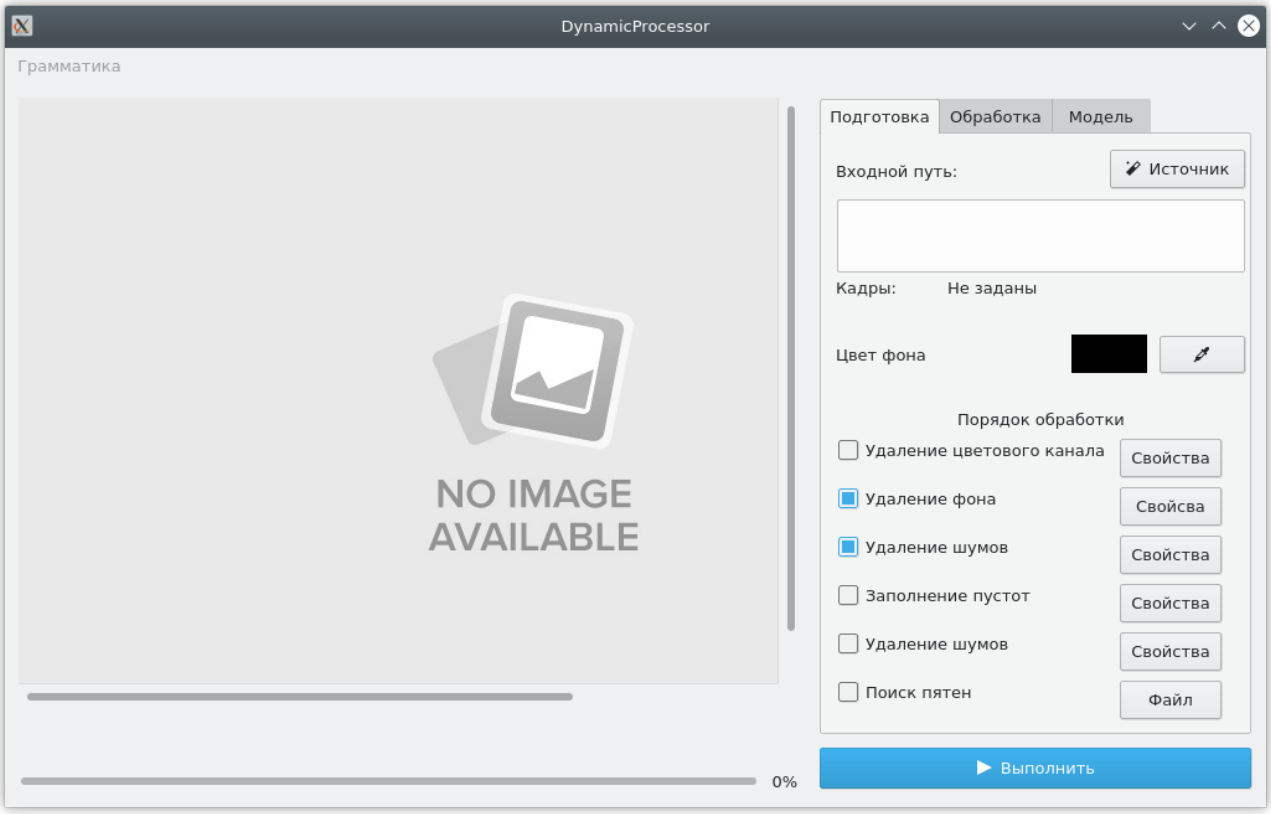

Рисунок 3.1 – Головне вікно програми

Для того, щоб отримати файл з розрахованими параметрами блискавки з вхідної послідовності, необхідно вказати параметри джерела:

– шлях до директорії;

– формат імені файлу, який складається з префіксу, формату номеру та розширення зображень (.jpg або .png);

– номера початкового та кінцевого зображень.

Для цього натисніть на кнопку «Джерело», після чого відкриється діалогове вікно (рис. 3.2).

Зверніть увагу, що обрані параметри зберігаються у пам’яті та підставляються у поля форми при наступних запусках.

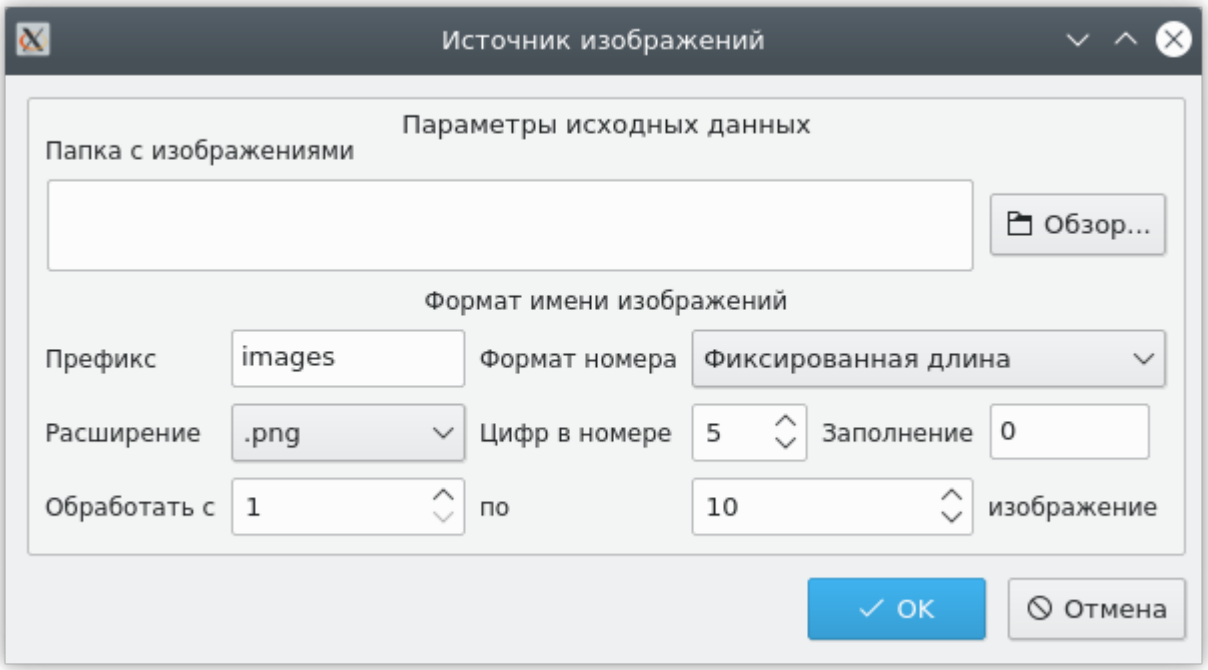

Рисунок 3.2 – Параметри джерела

Після вводу необхідних параметрів – натисніть на кнопку «ОК». Якщо параметри джерела вказані коректно, вікно закривається, а на головній формі програми з’являться введені параметри та відкриється перше зображення (рис. 2.3).

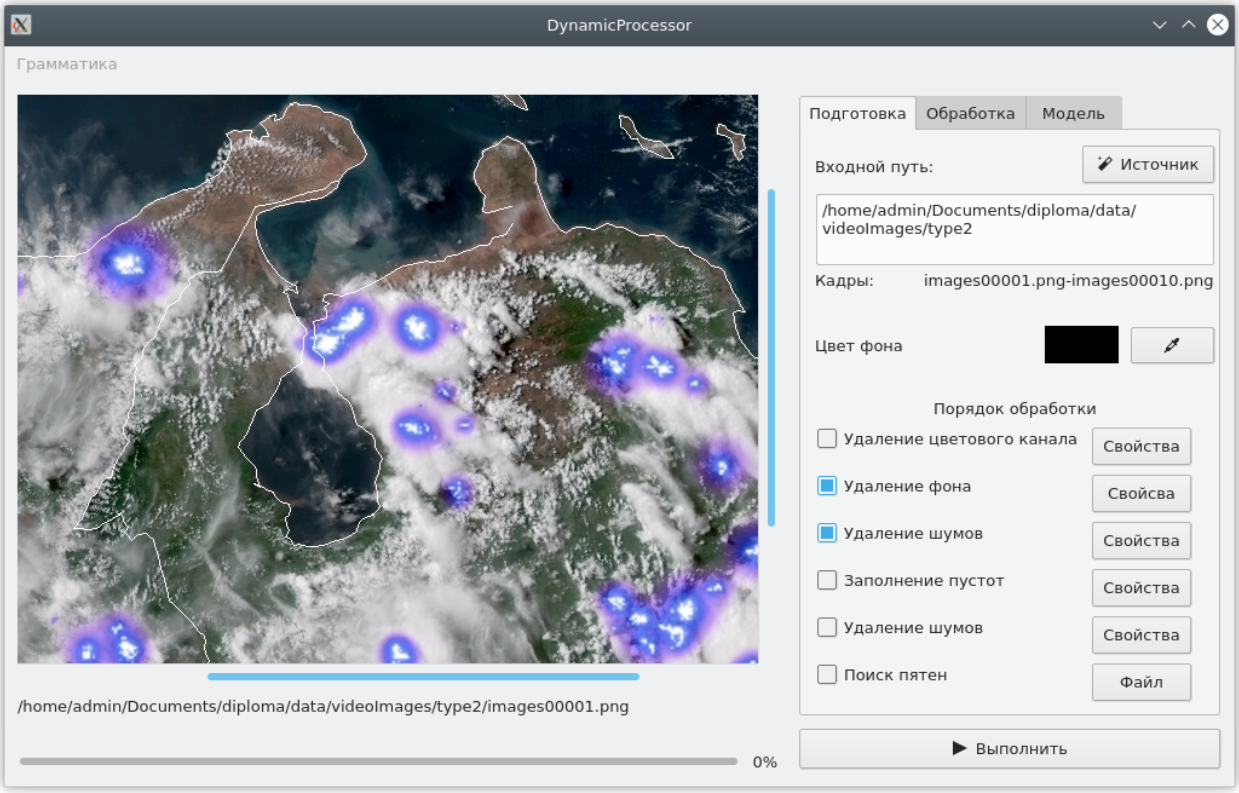

Рисунок 3.3 – Головне вікно після вибору параметрів джерела

Оберіть фон, який буде фоном для зображень з виокремленими блискавками, а також фоном вхідних зображень для дії «Пошук плям» шляхом натискання на кнопку зі значком піпетки.

Із списку дії оберіть ті, які необхідно виконати для досягнення бажаного результату, задайте їх параметри. Включення дії досягається шляхом встановлення відповідного прапорця. Для визначення параметрів дії натисніть на кнопку поруч.

Видалення колірного каналу являє собою позбавлення кожного пікселя певних складових у форматі RGB. Зробіть вибір із списку серед запропонованих варіантів: червоний, синій, зелений, червоний та зелений, червоний та синій, синій та зелений.

Видалення фону являє собою основну операцію з виокремлення блискавок. Видалення колірного каналу виконує роль попередньої обробки для нього, а видалення шумів та заповнення пустот необхідні для досягнення більш якісного результату. При натисненні на кнопку «Властивості» відкриється діалогове вікно (рис. 3.4), на якому розміщенні поля для вводу параметрів для видалення фону.

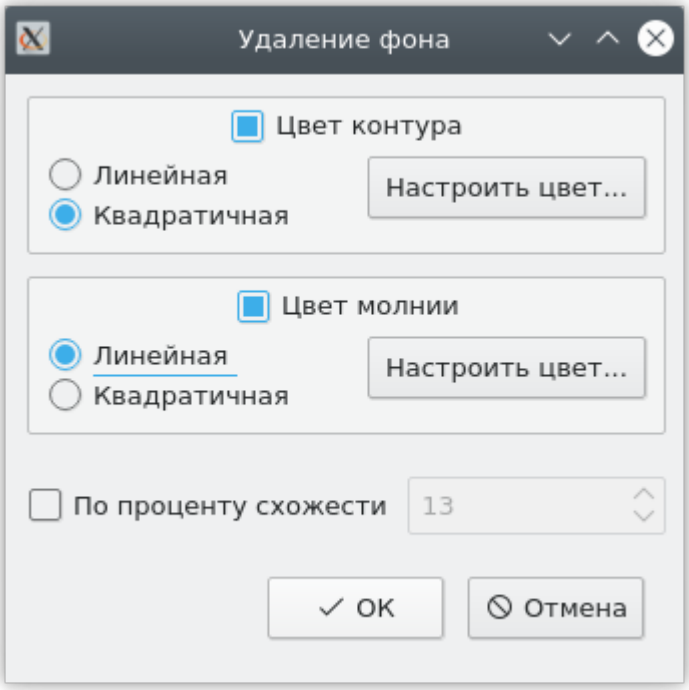

Рисунок 3.4 – Параметри видалення фону

За допомогою прапорців оберіть необхідні операції при видаленні фону.

Вони будуть виконані над кожним зображенням у порядку представленому на формі, тобто спочатку відбудеться пошук контуру за його кольором (якщо прапорець піднятий), потім програма на основі знайденого контуру знайде блискавки, а після – перейде до порівняння за процентом схожості вже отриманих кадрів з блискавками. Якщо прапорець кольору контуру опущений, програма одразу почне шукати блискавки за встановленими критеріями кольору.

Для налаштування кольору натисніть на кнопку «Налаштувати колір...» та у новому діалоговому вікні (рис. 3.5) визначте інтервали кожної складової у моделях Lab та LCH на відповідних вкладках. Вимкнення або включення перевірки за певною складовою досягається за допомогою відповідного прапорця. Пересувайте лівий та правий повзунок для задання відповідних меж інтервалу.

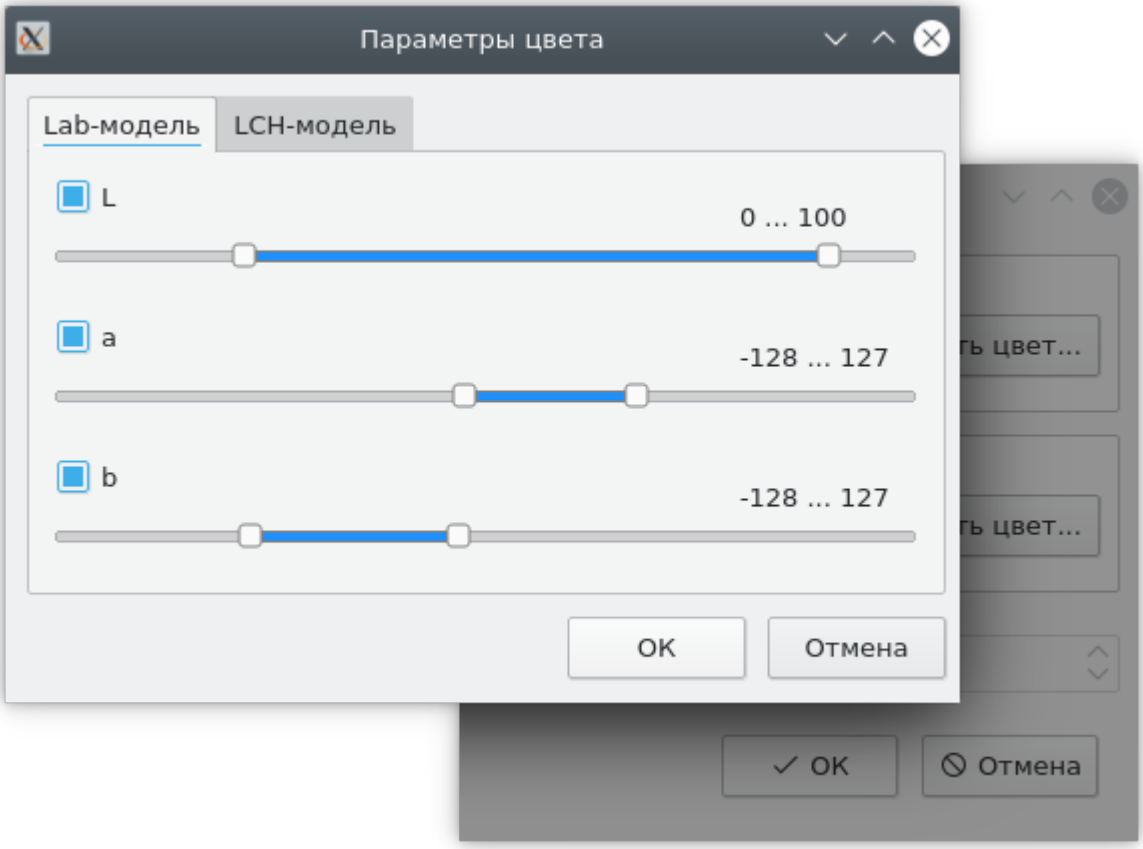

Рисунок 3.5 – Параметри колірних інтервалів

Введіть кількість циклів видалення шумів для кожного кадру та оберіть колір для заповнення пустот на головній формі (рис. 3.1) після натиснення на відповідну кнопку. Дані для виокремлення блискавок підготовлені.

Якщо після отримання виокремлених блискавок ще й необхідно отримати їх параметри, то необхідно підняти прапорець пошуку плям та обрати файл для збереження результатів. Слід відмітити, що ви можете виконати дану дію окремо від безпосередньо виокремлення, якщо необхідні кадри є в наявності (просто вкажіть їх в джерелі). Для виконання цієї дії оберіть файл для збереження результату (натисніть на кнопку «Файл»).

Натисніть на кнопку «Виконати», коли будуть введені всі параметри. Якщо була обрана дія з виокремлення блискавок, то перед початком виконання програма запропонує обрати директорію для збереження проміжних результатів. Програма буде інформувати яка дія зараз виконується та скільки кадрів уже оброблено.

Кадри з виокремленими блискавки з'являються в обраній директорії та показуються в якості зображення на формі. Файл з розрахованими параметрами у кожному рядку містить такі дані: номер кадру, координата центру (x та y) та радіус даної блискавки.

Для обробки файлу, що містить апроксимовані дані блискавок, перейдіть на вкладку «Обробка» та заповніть необхідні поля (рис. 3.6).

Для вибору, власне, файлу натисніть на кнопку «Джерело» та оберіть файл.

Для вибору директорії для збереження вихідних зображень, натисніть на кнопку»результат.

Вкажіть кількість кадрів, які необхідно пропустити від початку файлу, перш ніж почати наносити кадри на зображення у полі «Пропустити перших».

Поле «Кадрів на кадрі» показує яка кількість кадрів з файлу буде розміщена на одному вихідному зображені.

Щоб на вихідному зображені були нанесені блискавки різного кольору (для рідних вхідних кадрів) введіть у поле «Змінювати колір через кадрів» необхідне натуральне число. Якщо така опція не є необхідною залиште нульове значення.

Для запуску процесу обробки натисніть на кнопку «Виконати».

Щоб завершити роботу з програмую натисніть на хрестик у правому верхньому куті форми.

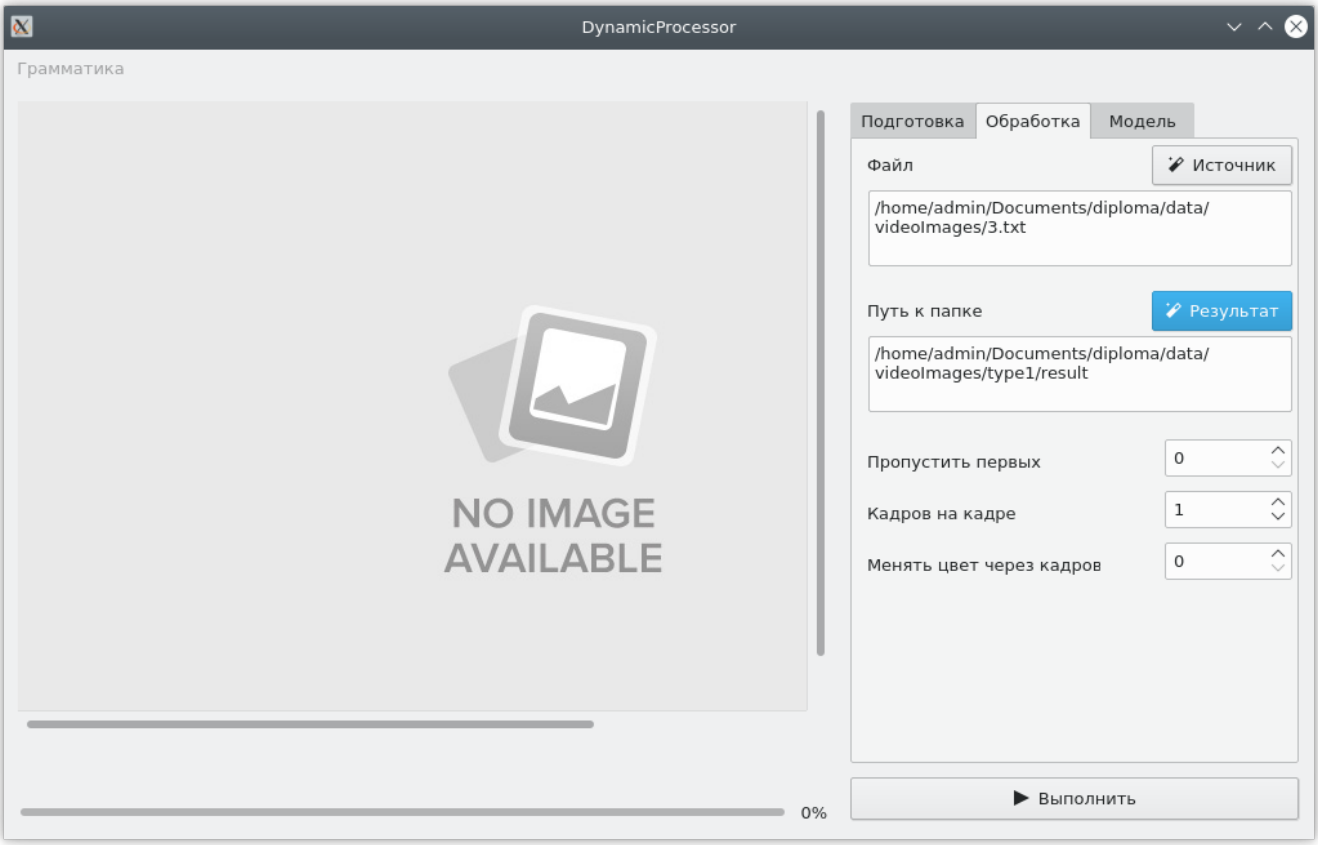

Рисунок 3.6 – Параметри відтворення апроксимованих блискавок

4 АВАРІЙНІ СИТУАЦІЇ

При відмові продукту через несправність зовнішнього середовища виконання – ОС – необхідно завершити процес програми засобами системи та знову запустити програму. При відмові ОС перезавантажте ПК.

При відмові функцій програми виконайте перезавантаження програми шляхом закриття і повторного запуску або виконайте повторне копіювання програми з носія та (або) зверніться до служби підтримки або (і) розробників продукту.

5 РЕКОМЕНДАЦІЇ ЩОДО ОСВОЄННЯ ТА ЕКСПЛУАТАЦІЇ

Для найшвидшого освоєння продукту користувачеві потрібно ознайомитися з керівництвом користувача.

Для безпечної експлуатації завершуйте роботу з програмою, використовуючи методи, вказані у керівництві користувача, завжди читайєте повідомлення, що з’являються при роботі з продуктом та дотримуйтесь вказівок, які в них містяться.

Контрольній приклад. Відкрийте програму та оберіть директорію до каталогу з кадрами відео зі статичним фоном. Відмітьте дії: видалення каналу, видалення фону, видалення шумів, заповнення пустот, видалення шумів. У властивостях видалення фону відмітьте лише прапорець «За відсотком схожості» та поставте значення рівним 13. Використовуйте всі інші параметри за замовченням. Натисніть на кнопку «Виконати» та вкажіть директорію для збереження файлів, дочекайтесь результатів обробки.

6 ПОВІДОМЛЕННЯ

У табл. 6.1 наведені повідомлення виключно користувачу, що можуть з’явитись у процесі виокремлення блискавок.

Таблиця 6.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Зображення <ім'я відео> не вдалось відкрити	Зображення з даним іменем не вдалось відкрити	Правильно вказуйте параметри джерела, перевірте, що зображення коректне
Вкажіть, будь ласка, <ім'я параметра>	Необхідно обрати значення відповідного параметра, наприклад, шлях до директорії, колір фону	Задайте необхідні параметри
Виконується дія <ім'я дії>	Програма виконує дію	Зачекайте завершення обробки
Обробка завершена успішно	Обробка відео завершена успішно	Оберіть ім'я файлу для збереження результатів
Не вдалось зберегти результати у <ім'я файлу>	Результати не можна зберегти до файлу з даним іменем	Перевірте, що даний файл доступний для запису та не використовується іншими програмами
Результати успішно збережені	Результати обробки були збережені до обраного файлу	Закрийте вікно або оберіть нові кадри для обробки

ЗАТВЕРДЖЕНО

01116130.01188-01 І3 02-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

Керівництво користувача. Керівництво з моделювання блискавок типу хмара-земля

01116130.01188-01 І3 02

Листів 13

2020

АНОТАЦІЯ

Документ 01116130.01188-01 І3 02 «Автоматизована система конструктивно-продукційного моделювання молнієвої активності. Керівництво користувача. Керівництво з моделювання блискавок типу хмара-земля» входить до складу програмної документації до програми.

Програмний продукт надає можливість автоматизувати процес виокремлення блискавок на кадрах поширення грозового фронту та будувати конструктори блискавки типу хмара-земля з подальшою візуалізацією. В документі міститься керівництво користувача з моделювання блискавок типу хмара-земля.

ЗМІСТ

ВСТУП

Областю застосування програмного продукту є комп'ютерне моделювання процесів розповсюдження блискавок типу хмара-земля. Програмний продукт призначено для для дослідження характеру поширення грозового фронту. Функції програми полягають у надані користувачу можливості побудувати чорну блискавку типу хмара-земля на білому фоні у двовимірному за заданими правилами продукції з можливістю визначити графічні параметри не-терміналу.

Продукт розраховано на базовий рівень підготовки користувача для роботи з персональним комп’ютером.

Перед використанням продукту користувачеві необхідно ознайомитися з відповідним керівництвом та правилами побудови граматик L-систем [1].

1 ПРИЗНАЧЕННЯ ТА УМОВИ ЗАСТОСУВАННЯ

Функціональне призначення розробки являє собою конструювання моделі (у двовимірному просторі з можливістю вказати довжину і початкову ширину нетерміналу) поширення блискавки типу хмара-земля у атмосфері з подальшою візуалізацією (блискавка має бути чорного кольору на білому фоні).

Експлуатаційне призначення продукту полягає в усуненні людини від задачі побудови графічного представлення блискавки типу хмара-земля, що дозволить значно швидше визначити як можна формалізувати дане явище, оскільки процес графічного відтворення граматик є складною та тривалою роботою.

Програмний продукт розрахований на персональні комп’ютери, що мають такі мінімальні характеристики:

– щонайменше 128 Мб вільного місця на вбудованому носії інформації;

– щонайменше 1024 Мб оперативної пам’яті;

– монітор з роздільною здатністю 800*600 і вище;

– центральний процесор з тактовою частотою 1 ГГц та більше;

– маніпулятор «миша»;

– стандартна клавіатура;

– для встановлення ПЗ необхідна або наявність CD/DVD приводу чи USB роз’єму, або підключення до мережі Інтернет.

Для роботи з програмним продуктом на ПК має бути встановлена ОС GNU/Linux 3.0 і вище, що має графічний інтерфейс.

2 ПІДГОТОВКА ДО РОБОТИ

Підготовка до роботи включає: ввімкнення ПК та завантаження ОС, перенесення ПЗ на робочу машину.

Для того, щоб перенести програму з носія на робочу машину, її потрібно скопіювати на вільний дисковий простір. Якщо програма представлена в стиснутому виді (дані розархівовано), то потрібно скопіювати архів у вільний дисковий простір, розархівувати на диск.

Для того, щоб запустити програмний продукт, необхідно відкрити виконуваний файл DynamicDrocessor та перейти на вкладку «Модель».

3 ОПИС ОПЕРАЦІЙ

Робота в програмі виконується за такими напрямками:

– конструювання правил продукції блискавок;

– відтворення блискавок за даними правилами.

На рис. 3.1 представлений зовнішній вигляд програми на вкладці «Модель».

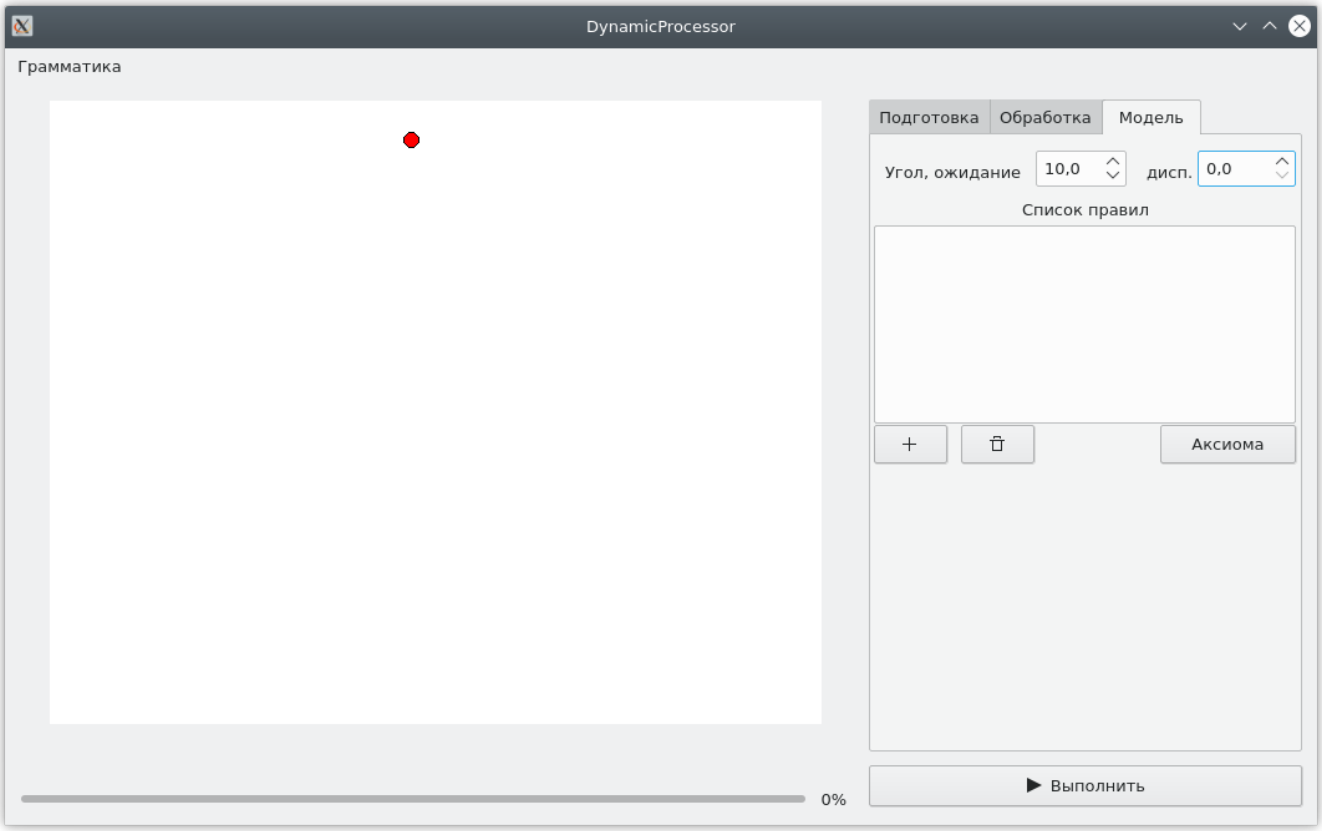

Рисунок 3.1 – Інтерфейс вкладки «Модель»

Для того, щоб визначити градусну міру кута повороту введіть його математичне очікування та дисперсію у відповідні поля.

Створіть список правил продукції. Для того, щоб додати новий не-термінал, натисніть на кнопку зі знаком «Плюс» та введіть ім'я не-терміналу у діалогове вікно. Слід зауважити, що у випадку, якщо таке ім'я не-терміналу вже використовується, введене значення не буде додано та на екран виведеться відповідне попередження.

За замовчуванням ліва частина правила еквівалентна правій. Для того, щоб визначити правило необхідно клацнути лівою кнопкою миші на правило та ввести математичне очікування та дисперсію довжини, товщину ліній графічного представлення не-терміналу у відповідні поля та, власне, праву частину правила

(поле «Правило). При вводі правила можна використовувати не-термінали та наступні термінали:

– «+» – лівий поворот (проти годинникової стрілки); поточний кут змінюється на число рівне поточному куту повороту;
– «-» – правий поворот (за годинниковою стрілкою); поточний кут змінюється на число рівне поточному куту повороту;
– «%» – згенерувати новий кут повороту за вказаним математичним очікуванням та дисперсією (нормальний розподіл);
– «#» – згенерувати нову довжину за вказаним математичним очікуванням та дисперсією (нормальний розподіл) для наступного не-терміналу у послідовності для відтворення;
– «[» – початок гілки; зберегти поточний кут повороту та позицію і продовжити розбір послідовності для відтворення;
– «]» – кінець гілки; присвоїти позиції та куту повороту останні збережені значення та продовжити розбір послідовності для відтворення.

Зверніть увагу, що початковий кут рівний 270 градусам, а початкове значення кута повороту і довжини не-терміналу визначені їх відповідними математичними очікуваннями.

Якщо бажаєте видалити правило із списку, натисніть на кнопку зі значком смітника, обравши необхідну строку у списку.

Ви можете зберегти введені правила. Для цього оберіть у пункті меню «Граматика» пункт «Зберегти» та вкажіть ім'я файлу для збереження через файловий діалог, що з'явиться.

Для того, щоб завантажити та продовжити працювати із раніше розробленою граматикою, оберіть у пункті меню «Граматика» пункт «Імпортувати» та оберіть файл, у який відбувалось збереження раніше. Зауважте, що якщо обрати файл, що не був створений даною програмою, поведінка програми буде невизначеною.

Щоб отримати графічне представлення розроблених правил продукції, натисніть на кнопку «Виконати». Вам буде запропоновано ввести кількість циклу перезапису правила. Після згенерована послідовність буде відтворена (рис. 3.2).

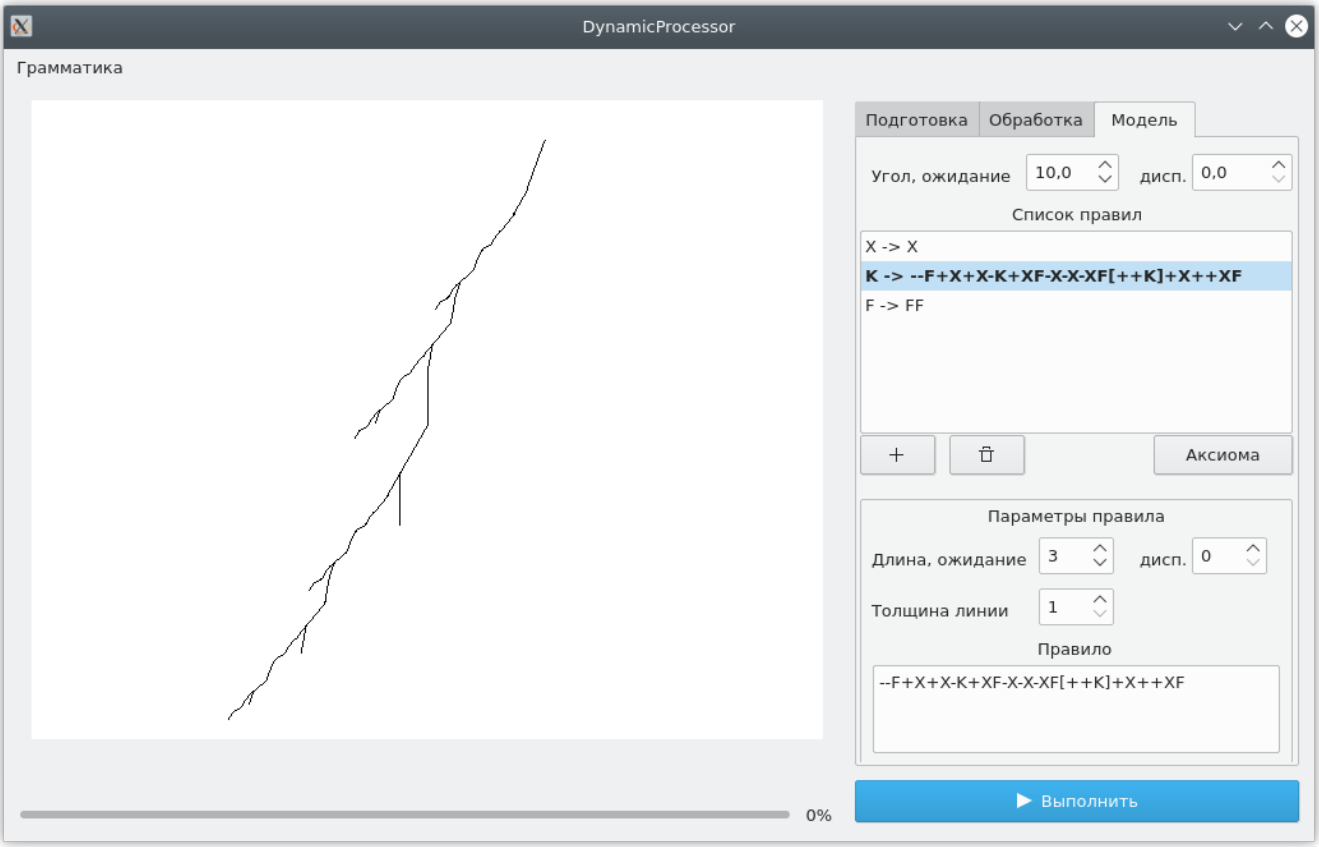

Рисунок 3.2 – Графічне представлення правил продукції

Ви можете збільшувати та зменшувати отримане зображення за допомогою клавіш «W» та «S» відповідно.

Щоб завершити роботу з програмою натисніть на хрестик у правому верхньому куті форми.

4 АВАРІЙНІ СИТУАЦІЇ

При відмові продукту через несправність зовнішнього середовища виконання – ОС – необхідно завершити процес програми засобами системи та знову запустити програму. При відмові ОС перезавантажте ПК.

При відмові функцій програми виконайте перезавантаження програми шляхом закриття і повторного запуску або виконайте повторне копіювання програми з носія та (або) зверніться до служби підтримки або (і) розробників продукту.

Контрольний приклад. Запустіть проект та перейдіть на вкладку «Модель». Заповніть поля так, як показано на рис. 3.2, використовуючи інформацію щодо опису операцій. Натисніть на кнопку «Виконати» та вкажіть значення кількості циклів перезапису рівним п'яти. На екрані з'явиться результат.

5 РЕКОМЕНДАЦІЇ ЩОДО ОСВОЄННЯ ТА ЕКСПЛУАТАЦІЇ

Для найшвидшого освоєння продукту користувачеві потрібно ознайомитися з керівництвом користувача та правилами побудови граматик L-систем [1].

Для безпечної експлуатації завершуйте роботу з програмою, використовуючи методи, вказані у керівництві користувача, завжди читаєте повідомлення, що з’являються при роботі з продуктом та дотримуйтесь вказівок, які в них містяться.

6 ПОВІДОМЛЕННЯ

У табл. 6.1 наведені повідомлення виключно користувачу, що можуть з’явитись у процесі виокремлення блискавок.

Таблиця 6.1 – Повідомлення користувачу

Текст повідомлення	Опис ситуації	Рекомендовані дії
Аксіома не обрана	Немає початкового символу для будування моделі	Оберіть аксіому відведеними засобами
Список правил відсутній	Список правил продукції не заповнений	Натисніть на кнопку додати правило та заповніть відведені поля
Даний не-термінал вже існує	Спроба створити раніше доданий не-термінал	Оберіть інший символ або змініть правило для того, що існує

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Prusinkiewicz, P. The Algorithmic Beauty of Plants [Текст] / P. Prusinkiewicz, A. Lindenmayer – New York, 1990. – 240 с.

ЗАТВЕРДЖЕНО

01116130.01188-01 12 01-ЛЗ

АВТОМАТИЗОВАНА СИСТЕМА КОНСТРУКТИВНО-ПРОДУКЦІЙНОГО МОДЕЛЮВАННЯ МОЛНІЄВОЇ АКТИВНОСТІ

Текст програми

01116130.01188-01 12 01

Листів 58

2020

АНОТАЦІЯ

Документ 01116130.01188-01 12 01 «Система автоматичного розпізнавання молнієвої активності грозового фронту. Текст програми» входить до складу програмної документації до програми. Програмний продукт надає можливість автоматизувати процес розпізнавання блискавок на кадрах поширення грозового фронту.

В документі міститься опис основних модулів програми та їх текст. Програма реалізована на мові C++ у програмному середовищі QT Creator 5.0.

ЗМІСТ

1 СХЕМА ВЗАЄМОДІЇ ПРОГРАМНИХ МОДУЛІВ

На рис. 1.1 представлена схема взаємодії основних модулів програми.

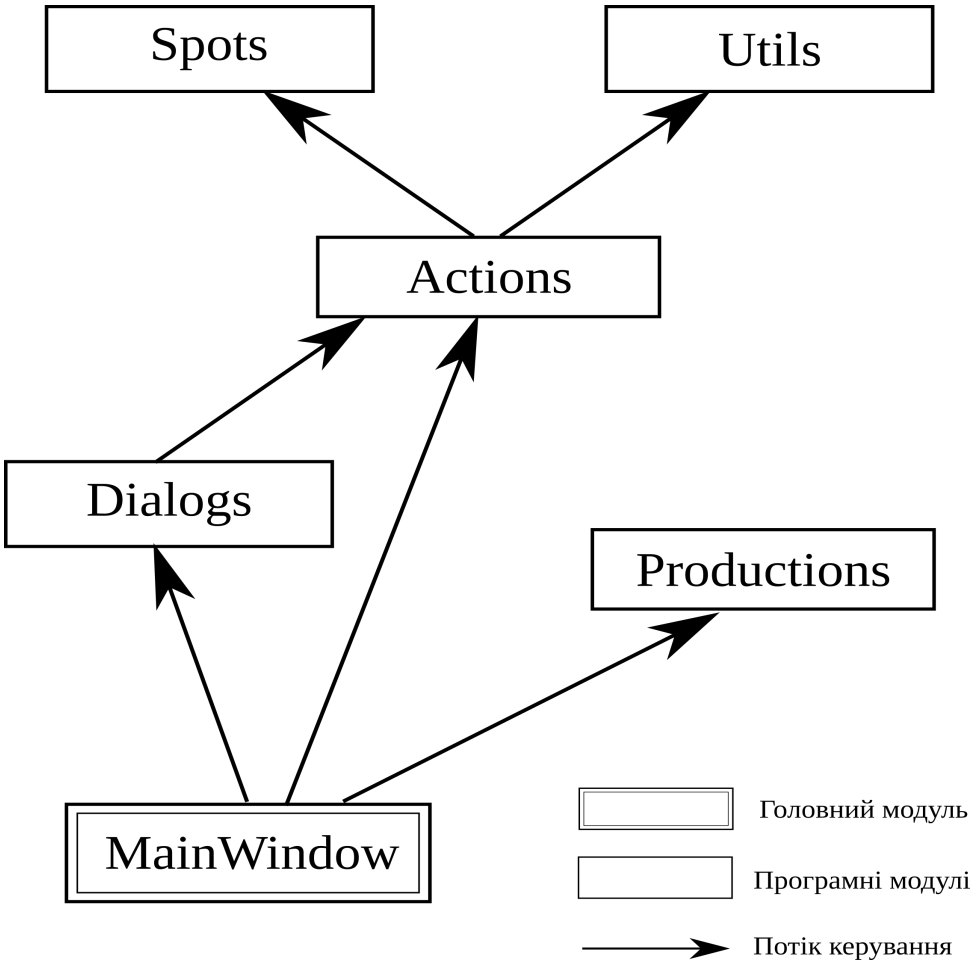

Рисунок 1.1 – Схема взаємодії основних модулів програми

2 ТЕКСТ ПРОГРАМИ

2.1 Текст програми у файлі main.cpp

```
#include "main_window.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

2.2 Текст програми у файлі main_window.h

```
#ifndef MAIN_WINDOW_H
#define MAIN_WINDOW_H

#include <QMainWindow>

#include "forms/image_sourse_form.h"
#include "forms/delete_background_dialog.h"
#include "file_processor.h"
#include "actions/actions.h"
#include "productions/drawablelabel.h"

#include <QImageReader>
#include <QImage>
#include <QKeyEvent>
#include <QScrollArea>
#include <QSizePolicy>
#include <QStandardItem>

namespace Ui {
    class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

protected:
    void keyPressEvent(QKeyEvent *event) override;

private slots:
    void on_pBSource_clicked();
    void receiveSourceData(Source *);

    void on_pushButton_chooseColor_clicked();
    void on_pushButton_background_clicked();
    void on_pushButton_colorChannel_clicked();
    void on_pushButton_noises_clicked();
    void on_pushButton_fill_clicked();
    void on_pushButton_noisesLast_clicked();
    void on_pushButton_fileName_clicked();

    void on_pBSource_2_clicked();
    void on_pBSource_resultDir_clicked();

    void on_selection_index_changed(const QModelIndex& selection);
    void on_pushButtonAdd_clicked();
    void on_pushButtonDelete_clicked();
    void on_pushButtonMain_clicked();
```

```
    void on_pushButton_clicked();

    void changeProgressBar(int processed, int total);
    void changeActionName(const QString &actionName);
    void showErrorMessage(const QString& errorMessage);

    void on_pushButton_saveDir_clicked();



    void on_tabWidget_currentChanged(int);


    void on_importGram_triggered();

    void on_saveGram_triggered();

private:
    Ui::MainWindow *ui;
    Actions preProcActions;
    Actions procActions;
    Source *preProcSrc;
    Source *procSrc;

    DrawableLabel *label_images;
    QScrollArea *scrollArea;

    int currentImageNumber;

    QString preprocessSaveDir;

    void setImage(const QString &imageName);
    void changeLabelColor(QLabel *label, QColor color);
    void runActionsProcessing(Actions &actions, const Source &src, const QString &dirToSave,
bool useDefParams);

    static const QString COMPANY_NAME;
    static const QString SOFT_NAME;

    QStandardItemModel rulesModel;
    SequenceGenerator sq;

    QChar nTerminal;
    int axiomInd;
    void updateValues();
};

#endif // MAIN_WINDOW_H
```

2.3 Текст програми у файлі main_window.cpp

```
#include "main_window.h"
#include "ui_mainwindow.h"
#include <QThread>
#include <QInputDialog>
#include "utils/actions_runner.h"
#include "actions/find_spots_action.h"
#include "actions/delete_color_channel.h"
#include "actions/delete_noises.h"
#include "actions/fill_empties.h"
#include "actions/draw_lightning_action.h"

const QString MainWindow::COMPANY_NAME = "MySoft";
const QString MainWindow::SOFT_NAME = "DynamicProcessor";

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent), ui(new Ui::MainWindow), preProcActions(Actions::CreateForPreProc())
{
    ui->setupUi(this);

    changeLabelColor(ui->label_color, preProcActions.get(0)->getBackgroundColor());
    QSettings settings(COMPANY_NAME, SOFT_NAME);
```

```
    int lastItemIndex = settings.value("deleteColorIndex", 0).toInt();
    static_cast<DeleteColorChannel*>(preProcActions.get(Actions::DELETE_COLOR_CHANNEL_N))-
>setColorChannel(static_cast<ColorChannel>(lastItemIndex));
    int firstCycleN = settings.value("firstDeleteNoisesCycleN", 5).toInt();
    static_cast<DeleteNoises*>(preProcActions.get(Actions::DELETE_NOISES1_N))-
>setCycles(firstCycleN);
    int lastCycleN = settings.value("lastDeleteNoisesCycleN", 5).toInt();
    static_cast<DeleteNoises*>(preProcActions.get(Actions::DELETE_NOISES2_N))-
>setCycles(lastCycleN);

    label_images = new DrawableLabel(this);
    label_images->setMinimumHeight(400);
    label_images->setMinimumWidth(600);
    label_images->setPixmap(QPixmap("./noimage.png"));
    label_images->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
    label_images->resize(label_images->pixmap()->size());

    scrollArea = new QScrollArea();
    scrollArea->setWidget(label_images);
    scrollArea->setBackgroundRole(QPalette::Background);
    scrollArea->setAlignment(Qt::AlignHCenter | Qt::AlignCenter);
    scrollArea->setFrameShape(QFrame::NoFrame);

    ui->gridLayout_2->addWidget(scrollArea, 0, 0);


    currentImageNumber = 0;
    preProcSrc = nullptr;
    procSrc = nullptr;

    //model creation parameters
    ui->listView->setModel(&rulesModel);

    connect(ui->listView->selectionModel(), SIGNAL(currentRowChanged(QModelIndex,
QModelIndex)),
          this, SLOT(on_selection_index_changed(QModelIndex)));

    ui->groupBoxRule->setVisible(false);
    axiomInd = -2;

    ui->tabWidget->setCurrentIndex(0);
}

MainWindow::~MainWindow()
{
    delete ui;
    delete preProcSrc;
    delete procSrc;
}

void MainWindow::keyPressEvent(QKeyEvent *event)
{
    if (ui->tabWidget->currentWidget() == ui->tab_prepare && preProcSrc != nullptr)
    {
        if (event->key() == Qt::Key::Key_D && preProcSrc->to > currentImageNumber)
        {
            setImage(preProcSrc->nameGenerator->getPath() +  "/" +
                     preProcSrc->nameGenerator->getFileName(++currentImageNumber));
        }
        else if (event->key() == Qt::Key::Key_A && preProcSrc->from < currentImageNumber)
        {
            setImage(preProcSrc->nameGenerator->getPath() +  "/" +
                     preProcSrc->nameGenerator->getFileName(--currentImageNumber));
        }
    }
    if (ui->tabWidget->currentWidget() == ui->tab_model)
    {
        switch (event->key()) {
        case Qt::Key_W:
            label_images->zoomIn();
            break;
```

```
        case Qt::Key_S:
            label_images->zoomOut();
            break;
        }
    }

}

void MainWindow::on_pBSource_clicked()
{
    ImageSourseForm* newForm = new ImageSourseForm(COMPANY_NAME, SOFT_NAME);
    connect(newForm, SIGNAL(sendSourceData(Source*)), this, SLOT(receiveSourceData(Source*)));
    newForm->show();
}

void MainWindow::receiveSourceData(Source* source)
{
    if (source == nullptr)
        return;
    delete preProcSrc;
    preProcSrc = source;
    ui->l_fromTo->setText(preProcSrc->nameGenerator->getFileName(preProcSrc->from) + "-" +
                          preProcSrc->nameGenerator->getFileName(preProcSrc->to));
    ui->plainTextEdit_inputPath->setPlainText(preProcSrc->nameGenerator->getPath());

    setImage(preProcSrc->nameGenerator->getPath() + "/" +
             preProcSrc->nameGenerator->getFileName(preProcSrc->from));

    currentImageNumber = preProcSrc->from;
}

void MainWindow::on_pushButton_chooseColor_clicked()
{
    QColor color = QColorDialog::getColor(Qt::black, this, tr("Выберите цвет"));
    changeLabelColor(ui->label_color, color);
    for (int i = 0; i < preProcActions.getSize(); i++)
    {
        preProcActions.get(i)->setBackgroundColor(color);
    }
}

void MainWindow::on_pushButton_colorChannel_clicked()
{
    static const QMap<QString, ColorChannel> NameToValueMap = { {"Красный",
ColorChannel::Red}, {"Зелёный", ColorChannel::Green}, {"Синий", ColorChannel::Blue},
                                                                {"Красный и зелёный",
ColorChannel::RedAndGreen}, {"Зелёный и синий", ColorChannel::GreenAndBlue},
                                                                {"Красный и синий",
ColorChannel::RedAndBlue}};

    QSettings settings(COMPANY_NAME, SOFT_NAME);
    int lastItemIndex = settings.value("deleteColorIndex", 0).toInt();

    QStringList items;
    items << "Красный" << "Зелёный" << "Синий" << "Красный и зелёный" << "Зелёный и синий" <<
"Красный и синий";

    bool ok;
    QString item = QInputDialog::getItem(this, tr("Свойства удаления канала"),
                                         tr("Удалить:"), items, lastItemIndex, false, &ok);

    if (!ok)
        return;

    ColorChannel selectedItem = NameToValueMap[item];
    settings.setValue("deleteColorIndex", static_cast<int>(selectedItem));

    static_cast<DeleteColorChannel*>(preProcActions.get(Actions::DELETE_COLOR_CHANNEL_N))-
>setColorChannel(selectedItem);
}
```

```
void MainWindow::on_pushButton_background_clicked()
{
    DeleteBackgroundDialog* parmsDialog = new DeleteBackgroundDialog(COMPANY_NAME, SOFT_NAME,

static_cast<DeleteBackground*>(preProcActions.get(Actions::DELETE_BACKGROUND_N)));
    parmsDialog->show();
}

void MainWindow::on_pushButton_noises_clicked()
{
    QSettings settings(COMPANY_NAME, SOFT_NAME);
    int firstCycleN = settings.value("firstDeleteNoisesCycleN", 5).toInt();

    bool ok;
    int cycleN = QInputDialog::getInt(this, tr("Свойсва удаления шумов"),
                                      tr("Раз обработки для кадра:"), firstCycleN, 1, 10, 1,
&ok);

    if (!ok)
        return;

    settings.setValue("firstDeleteNoisesCycleN", cycleN);
    static_cast<DeleteNoises*>(preProcActions.get(Actions::DELETE_NOISES1_N))-
>setCycles(cycleN);
}

void MainWindow::on_pushButton_fill_clicked()
{
    QColor color =
QColorDialog::getColor(static_cast<FillEmpties*>(preProcActions.get(Actions::FILL_EMPTIES_N))-
>getObjectColor(), this, tr("Выберите цвет"));
    static_cast<FillEmpties*>(preProcActions.get(Actions::FILL_EMPTIES_N))-
>setObjectColor(color);
}

void MainWindow::on_pushButton_noisesLast_clicked()
{
    QSettings settings(COMPANY_NAME, SOFT_NAME);
    int lastCycleN = settings.value("lastDeleteNoisesCycleN", 5).toInt();

    bool ok;
    int cycleN = QInputDialog::getInt(this, tr("Свойсва удаления шумов"),
                                      tr("Раз обработки для кадра:"), lastCycleN, 1, 10, 1,
&ok);

    if (!ok)
        return;

    settings.setValue("lastDeleteNoisesCycleN", cycleN);
    static_cast<DeleteNoises*>(preProcActions.get(Actions::DELETE_NOISES2_N))-
>setCycles(cycleN);
}

void MainWindow::on_pushButton_fileName_clicked()
{
    QSettings settings(COMPANY_NAME, SOFT_NAME);
    QString fileName = settings.value("fileWithSpots", "/home").toString();

    QString newFileName = QFileDialog::getSaveFileName(this, tr("Save File"), fileName,
tr("TextFiles (*.txt)"));

    if (newFileName.length())
    {
        static_cast<FindSpotsAction*>(preProcActions.get(Actions::FIND_SPOTS_N))-
>setFileName(newFileName);
        settings.setValue("fileWithSpots", newFileName);
    }
}

void MainWindow::on_pBSource_2_clicked()
{
```

```
    QSettings settings(COMPANY_NAME, SOFT_NAME);
    QString fileName = settings.value("fileWithSpots", "/home").toString();

    QString newFileName = QFileDialog::getOpenFileName(this, tr("Save File"), fileName,
tr("TextFiles (*.txt)"));

    if (newFileName.length())
    {
        //static_cast<FindSpotsAction*>(preProcActions.get(Actions::FIND_SPOTS_N))-
>setFileName(newFileName);
        settings.setValue("fileWithSpots", newFileName);
        ui->plainTextEdit_fileName->setPlainText(newFileName);
    }
}

void MainWindow::on_pBSource_resultDir_clicked()
{
    QSettings settings(COMPANY_NAME, SOFT_NAME);
    QString lastDir = settings.value("resultDir", "/home").toString();

    QString dir = QFileDialog::getExistingDirectory(this, tr("Save Directory"), lastDir,
                                                    QFileDialog::ShowDirsOnly |
QFileDialog::DontResolveSymlinks);
    if (dir.length())
    {
        //preprocessSaveDir = dir;
        settings.setValue("resultDir", dir);
        ui->plainTextEdit_saveDirName->setPlainText(dir);
    }
}

void MainWindow::on_selection_index_changed(const QModelIndex &selection)
{
    if (!selection.isValid())
    {
        ui->groupBoxRule->setVisible(false);
        return;
    }
    updateValues();
    nTerminal = rulesModel.item(selection.row())->text()[0];
    ui->groupBoxRule->setVisible(true);
    ui->textEditRule->setText(sq.getValue(nTerminal));
    ui->spinBoxWidth->setValue(sq.getDescriptor(nTerminal).width);
    ui->spinBoxLength->setValue(sq.getDescriptor(nTerminal).lExpected);
    ui->spinBoxLDispertion->setValue(sq.getDescriptor(nTerminal).lDispertion);
    rulesModel.item(selection.row())->setText(QString(nTerminal) + " -> " +
sq.getValue(nTerminal));
}

void MainWindow::on_pushButtonAdd_clicked()
{
    bool ok;
    QString text = QInputDialog::getText(this, tr("Добавление нового правила"),
                                         tr("Символ правила:"), QLineEdit::Normal, "", &ok);
    if (ok && !text.isEmpty())
    {
      if (text.length() != 1)
          return;

      text = text.toUpper();
      if (!sq.addNTerminal(text[0]))
      {
          QMessageBox msgBox;
          msgBox.setText("Такой не-терминал уже существует");
          msgBox.exec();
          return;
      }

      //add to model generator
      QStandardItem* newItem = new QStandardItem(text + " -> " + sq.getValue(text[0]));
      rulesModel.insertRow(rulesModel.rowCount(), newItem);
```

```
    }
}

void MainWindow::on_pushButtonDelete_clicked()
{
    const int selectedRowIndex = ui->listView->selectionModel()->currentIndex().row();
    if (selectedRowIndex != -1)
    {
        nTerminal = rulesModel.item(selectedRowIndex)->text()[0];
        sq.deleteTerminal(rulesModel.item(selectedRowIndex)->text()[0]);
        nTerminal = '\0';
        rulesModel.removeRow(selectedRowIndex);
        if (axiomInd == selectedRowIndex)
        {
            sq.setAxiom('\0');
            axiomInd = -2;
        }
    }
}

void MainWindow::on_pushButtonMain_clicked()
{
    const int selectedRowIndex = ui->listView->selectionModel()->currentIndex().row();
    if (selectedRowIndex != -1 && selectedRowIndex != axiomInd)
    {
        rulesModel.item(selectedRowIndex)->setFont(QFont("Helvetica [Cronyx]", 10,
QFont::Bold));
        sq.setAxiom(rulesModel.item(selectedRowIndex)->text()[0]);
        if (axiomInd != -2)
            rulesModel.item(axiomInd)->setFont(QFont("Helvetica [Cronyx]"));
        axiomInd = selectedRowIndex;
    }
}

void MainWindow::updateValues()
{
    if (nTerminal != '\0' && ui->groupBoxRule->isVisible())
    {
        sq.updateValue(nTerminal, ui->textEditRule->toPlainText());
        sq.updateDescriptor(nTerminal, NTerminalDescriptor(
                                ui->spinBoxLength->value(), ui->spinBoxLDispertion->value(),
                                ui->spinBoxWidth->value()));
    }
}

void MainWindow::on_pushButton_clicked()
{
    if (ui->tabWidget->currentWidget() == ui->tab_process)
    {
        if (preProcActions.getSize() == 0 || procSrc == nullptr)
        {
            QMessageBox msgBox;
            msgBox.setText("Не заданы действия, источник изображений или выходной путь");
            msgBox.exec();
            return;
        }
        runActionsProcessing(procActions, *procSrc, preprocessSaveDir, false);
        return;
    }
    else if (ui->tabWidget->currentWidget() == ui->tab_prepare)
    {
        if (preProcActions.getSize() == 0 || preProcSrc == nullptr ||
preprocessSaveDir.length() == 0)
        {
            QMessageBox msgBox;
            msgBox.setText("Не заданы действия, источник изображений или выходной путь");
            msgBox.exec();
            return;
        }
        runActionsProcessing(preProcActions, *preProcSrc, preprocessSaveDir, false);
        return;
```

```
    }

    bool ok;
    int n = QInputDialog::getInt(this, tr("Количество циклов замены"),
                                 tr("n:"), 5, 1, 100, 1, &ok);

    if (!ok)
        return;

    updateValues();
    if (axiomInd == -2 || rulesModel.rowCount() == 0)
    {
        QMessageBox msgBox;
        msgBox.setText("Не задана аксиома или не назначены правила");
        msgBox.exec();
        return;
    }

    label_images->setAlphaExpected(ui->doubleSpinBoxAlpha->value());
    label_images->setAlphaDispertion(ui->spinBoxADispertion->value());
    label_images->draw(sq.getSequence(n), sq.getDescriptors());
}

void MainWindow::runActionsProcessing(Actions &actions, const Source &src, const QString
&dirToSave, bool useDefParams)
{
    ActionsRunner* runner = new ActionsRunner(actions, src, dirToSave, useDefParams);

    QThread* thread = new QThread();

    connect(thread, SIGNAL(started()), runner, SLOT(run())); //when the thread starts method
process will be called
    connect(runner, SIGNAL(finished()), thread, SLOT(quit())); //when the metod finishes the
thead will be quit
    connect(runner, SIGNAL(processedQuantityChanged(int, int)), this,
SLOT(changeProgressBar(int, int))); //to change the state of progress bar
    connect(runner, SIGNAL(actionNameChanged(const QString&)), this,
SLOT(changeActionName(const QString&)));
    connect(runner, SIGNAL(processError(const QString&)), this, SLOT(showErrorMessage(const
QString&)));

    //check the classes to delete after end of work
    connect(runner, SIGNAL(finished()), runner, SLOT(deleteLater()));
    connect(thread, SIGNAL(finished()), thread, SLOT(deleteLater()));

    runner->moveToThread(thread);
    thread->start();
}

void MainWindow::changeProgressBar(int processed, int total)
{
    ui->progressBar->setMaximum(total);
    ui->progressBar->setValue(processed);
}

void MainWindow::changeActionName(const QString &actionName)
{
    ui->label_actionName->setText(actionName);
}

void MainWindow::showErrorMessage(const QString& errorMessage)
{
    QMessageBox msgBox;
    msgBox.setText(errorMessage);
    msgBox.exec();
}

void MainWindow::on_pushButton_saveDir_clicked()
{
    QSettings settings(COMPANY_NAME, SOFT_NAME);
    QString lastDir = settings.value("PreprocessImagesDir", "/home").toString();
```

```
    QString dir = QFileDialog::getExistingDirectory(this, tr("Open Directory"), lastDir,
                                                    QFileDialog::ShowDirsOnly |
QFileDialog::DontResolveSymlinks);
    if (dir.length())
    {
        preprocessSaveDir = dir;
        settings.setValue("PreprocessImagesDir", dir);
    }
}

void MainWindow::setImage(const QString& imageName)
{
    QPixmap pix(imageName);
    if (pix.isNull())
    {
        label_images->setText(QStringLiteral("Невозможно открыть изображение
%1").arg(imageName));
        ui->label_imageName->setText(imageName);
        return;
    }
    label_images->setText("");
    label_images->setPixmap(pix);
    label_images->resize(label_images->pixmap()->size());
    ui->label_imageName->setText(imageName);
}

void MainWindow::changeLabelColor(QLabel *label, QColor color)
{
    label->setAutoFillBackground(true);
    QPalette pal =  label->palette();
    pal.setColor(QPalette::Window, color);
    label->setPalette(pal);
}

void MainWindow::on_tabWidget_currentChanged(int )
{
    if (ui->tabWidget->currentWidget() == ui->tab_model)
    {
        label_images->setIsDrawMode(true);
        ui->menuGram->setEnabled(true);
        return;
    }
    label_images->setIsDrawMode(false);
    ui->menuGram->setEnabled(false);
    if (ui->tabWidget->currentWidget() == ui->tab_process)
    {
        label_images->setPixmap(QPixmap("./noimage.png"));
        return;
    }
    if (preProcSrc == nullptr)
         label_images->setPixmap(QPixmap("./noimage.png"));
    else
        setImage(preProcSrc->nameGenerator->getPath() + "/" +
                 preProcSrc->nameGenerator->getFileName(currentImageNumber));
}

void MainWindow::on_importGram_triggered()
{
    QSettings settings(COMPANY_NAME, SOFT_NAME);

    //get last directory
    QString input = settings.value("fileLoad", "/home").toString();

    QString fileName = QFileDialog::getOpenFileName(this, tr("Open File"), input, tr("Text
files (*.txt)"));

    if (fileName.length()) {
        settings.setValue("fileLoad", fileName);
        sq.load(fileName);
        rulesModel.clear();
```

```
        QHash<QChar, NTerminalDescriptor> descriptors = sq.getDescriptors();
        for (QHash<QChar, NTerminalDescriptor>::const_iterator it = descriptors.constBegin();
            it != descriptors.constEnd(); ++it) {
            rulesModel.insertRow(rulesModel.rowCount(), new QStandardItem(it.key()));
            if (it.key() == sq.getAxiom()) {
                rulesModel.item(rulesModel.rowCount()-1)->setFont(QFont("Helvetica [Cronyx]",
12, QFont::Bold));
                axiomInd = rulesModel.rowCount()-1;
            }
        }
    }
}

void MainWindow::on_saveGram_triggered()
{
    updateValues();
    if (axiomInd == -2 || rulesModel.rowCount() == 0) {
        QMessageBox msgBox;
        msgBox.setText("Не задана аксиома или не назначены правила");
        msgBox.exec();
        return;
    }
    QSettings settings(COMPANY_NAME, SOFT_NAME);

    //get last save file
    QString output = settings.value("fileSave", "/home").toString();

    QString fileName = QFileDialog::getSaveFileName(this, tr("Save File"), output, tr("Text
files (*.txt)"));

    if (fileName.length())
    {
        settings.setValue("fileSave", fileName);
        sq.save(fileName);
        QMessageBox msgBox;
        msgBox.setText("Грамматика была сохранена");
        msgBox.exec();
    }
}
```

2.4 Текст програми у файлі file_processor.h

```
#ifndef FILEPROCESSOR_H
#define FILEPROCESSOR_H

#include <QObject>
#include <QFile>
#include <QTextStream>
#include <QImage>
#include <QPainter>

#include "structures.h"

class FileProcessor : public QObject
{
    Q_OBJECT
public:
    FileProcessor(const QString &fileName, const QString &saveDir);
public slots:
    void run();
signals:
    void finished();
    void processedQuantityChanged(int processed, int total);
private:
    QString fileName;
    QString saveDir;
};

#endif // FILEPROCESSOR_H
```

2.5 Текст програми у файлі file_processor.cpp

```
#include "file_processor.h"

FileProcessor::FileProcessor(const QString &fileName, const QString &saveDir) :
    fileName(fileName), saveDir(saveDir)
{

}

void FileProcessor::run()
{
    QFile file(fileName);
    if (!file.open(QIODevice::ReadOnly | QIODevice::Text))
        return;
    QTextStream in(&file);
    int delta = 0;

    QFile f("/home/admin/Desktop/b.txt");
    if (!f.open(QIODevice::WriteOnly | QIODevice::Text))
        return;
    QTextStream out(&f);
    int q = 0;
    //set image backcolor
    QImage image = QImage(1024, 770, QImage::Format_RGB32);
    image.fill(QColor("white"));
    QPainter painter(&image);
    QVector<QColor> colors = { QColor("yellow"), QColor("aqua"), QColor("magenta"),
QColor("red"), QColor("blue")};

    painter.setBrush(QColor("black"));
    int imageNumber = 0;
    int lastNumber = -1;
    int colorInd = 1;
    while (!in.atEnd())
    {
        QStringList parms = in.readLine().split(" ");
        int number = parms[0].toInt();
        int x = parms[1].toInt();
        int y = parms[2].toInt();
        int r = parms[3].toInt();


            painter.drawEllipse(x, y, 2*r, 2*r);
            q++;

        //это новое изображение
        if (number != lastNumber)
        {
            delta++;
            lastNumber = number;

            //save image with circules if exist
            if (delta == 1)
            {
                image.save(saveDir + "/" + QString::number(number) + ".png");
               image.fill(QColor("white"));
               imageNumber++;
                delta = 0;
           //     painter.setBrush(colors[colorInd++]);
                out << q << endl;
                q = 0;
            }
        }
    }
    image.save(saveDir + "/" + QString::number(lastNumber) + ".png");
    out << q << endl;
    emit finished();
}
```

2.6 Текст програми у файлі structures.h

```
#ifndef STRUCTURES_H
#define STRUCTURES_H
```

```
#include "utils/i_name_generator.h"

struct Source
{
    int from;
    int to;
    INameGenerator* nameGenerator = nullptr;

    ~Source()
    {
        if (nameGenerator != nullptr)
            delete nameGenerator;
    }
};

struct ParmsLab {
    bool isL;
    int toL;
    int fromL;

    bool isA;
    int toA;
    int fromA;

    bool isB;
    int toB;
    int fromB;
};

struct ParmsLCH {
    bool isL;
    int toL;
    int fromL;

    bool isC;
    int toC;
    int fromC;

    bool isH;
    int toH;
    int fromH;
};

#endif // STRUCTURES_H
```

2.7 Текст програми у файлі actions.h

```
#ifndef ACTIONS_H
#define ACTIONS_H

#include "i_action.h"
#include <QVector>

class Actions {
public:
    ~Actions();

    bool add(IAction* newAction);
    void remove(int index);

    void moveUp(int index);
    void moveDown(int index);

    int getSize() const;

    const IAction* get(int index) const;
    IAction* get(int index);

    static Actions CreateForPreProc();

    static const int DELETE_COLOR_CHANNEL_N = 0;
```

```
    static const int DELETE_BACKGROUND_N = 1;
    static const int DELETE_NOISES1_N = 2;
    static const int FILL_EMPTIES_N = 3;
    static const int DELETE_NOISES2_N = 4;
    static const int FIND_SPOTS_N = 5;

private:
    QVector<IAction*> actions;
};

#endif // ACTIONS_H
```

2.8 Текст програми у файлі actions.cpp

```
#include "actions.h"
#include <assert.h>

Actions::~Actions()
{
    for (int i = 0; i < actions.size(); i++)
    {
        delete actions[i];
    }
}

bool Actions::add(IAction* newAction)
{
    actions.push_back(newAction);
    return true;
}

void Actions::remove(int index)
{
    if (index < 0 || index >= actions.size())
    {
        assert(false);
        return;
    }
    actions.erase(actions.begin() + index);
}

void Actions::moveUp(int index)
{
    if (index <= 0 || index >= actions.size())
    {
        assert(false);
        return;
    }
    std::swap(actions[index], actions[index-1]);
}

void Actions::moveDown(int index)
{
    if (index < 0 || index >= actions.size() - 1)
    {
        assert(false);
        return;
    }
    std::swap(actions[index], actions[index+1]);
}

int Actions::getSize() const
{
    return actions.size();
}

const IAction* Actions::get(int index) const
{
    if (index < 0 || index >= actions.size())
    {
        assert(false);
        return nullptr;
```

```
    }
    return actions[index];
}

IAction* Actions::get(int index)
{
    if (index < 0 || index >= actions.size())
    {
        assert(false);
        return nullptr;
    }
    return actions[index];
}

Actions Actions::CreateForPreProc()
{
    Actions actions;

    actions.add(IAction::Create(ActionCode::DeleteChannel));
    actions.add(IAction::Create(ActionCode::DeleteBackGround));
    actions.add(IAction::Create(ActionCode::DeleteNoises));
    actions.add(IAction::Create(ActionCode::FillEmpties));
    actions.add(IAction::Create(ActionCode::DeleteNoises));
    actions.add(IAction::Create(ActionCode::FindSpots));

    return actions;
}
```

2.9 Текст програми у файлі delete_background.h

```
#ifndef DELETE_BACKGROUND_H
#define DELETE_BACKGROUND_H

#include "i_action.h"
#include "utils/colorconvertor.h"
#include "utils/decisive_function.h"
#include "structures.h"

class DeleteBackground : public IAction {
public:
    DeleteBackground();
    bool process(const QString &objectDir, const QString &objectName) override;
    void reset() override;

    int getLevel() const
    {
        return level;
    }

    void setLevel(int value)
    {
        level = value;
    }

    bool getIsDynamic() const
    {
        return isDynamic;
    }

    void setIsDynamic(bool value)
    {
        isDynamic = value;
    }

    const DecisiveFunction &getFunctionContour() const
    {
        return functionContour;
    }

    DecisiveFunction &getFunctionContour()
    {
        return functionContour;
```

```
    }

    const DecisiveFunction &getFunctionLightning() const
    {
        return functionLightning;
    }

    DecisiveFunction& getFunctionLightning()
    {
        return functionLightning;
    }

    const QString getName() const override
    {
        return DeleteBackground::ACTION_NAME;
    }

private:
    DecisiveFunction functionContour;
    DecisiveFunction functionLightning;
    bool isDynamic = false;
    int level = SIM_LEVEL_DEF;

    static const int SIM_LEVEL_DEF = 13;
    static const QString ACTION_NAME;
    static const int RGB_RANGE_VALUE = 256;

    QImage *backImage;

    bool areColorsEqual(const QColor &color1, const QColor &color2) const;
    bool setLight(const int i, const int j, const QImage &image, const QImage &image2) const;
};

#endif // DELETE_BACKGROUND_H
```

2.10 Текст програми у файлі delete_background.cpp

```
#include "delete_background.h"
#include <iostream>

const QString DeleteBackground::ACTION_NAME = "Удаление фона";


DeleteBackground::DeleteBackground()
    : functionContour(ParmsLab(), ParmsLCH(), DecisiveFunctionType::Quadratic),
      functionLightning(ParmsLab(), ParmsLCH(), DecisiveFunctionType::Quadratic)
{
    functionContour.setToUse(true);
    functionLightning.setToUse(true);
}

bool DeleteBackground::process(const QString &objectDir, const QString &objectName)
{
    QImage image = GetImage(objectDir, objectName);

    if (image.isNull())
        return false;

    //save image without changes
    QImage imageCopy = image;

    //create contour
    if (functionContour.getToUse() == true)
    {
        for (int i = 0; i < image.width(); i++)
        {
            for (int j = 0; j < image.height(); j++)
            {
                QPoint point(i, j);
                if (!functionContour.isObject(image.pixelColor(point)))
                    image.setPixelColor(point, getBackgroundColor());
            }
```

```
        }
        if (functionLightning.getToUse() == true)
        {
            QImage newImage = image;
            for (int x = 0; x < image.width(); x++)
            {
                for (int y = 0; y < image.height(); y++)
                {
                      newImage.setPixelColor(QPoint(x, y), getBackgroundColor());
                }
            }

            for (int x = 0; x < image.width(); x++)
            {
                for (int y = 0; y < image.height(); y++)
                {
                    QPoint point(x, y);
                    QColor color = image.pixelColor(point);
                    if(color == getBackgroundColor() && setLight(x, y, image, newImage) &&
                            functionLightning.isObject(imageCopy.pixelColor(point)))
                    {
                            newImage.setPixelColor(point, imageCopy.pixelColor(point));
                    }
                }
            }
            image = newImage;
        }
    }
    else if (functionLightning.getToUse() == true)
    {
        for (int i = 0; i < image.width(); i++)
        {
            for (int j = 0; j < image.height(); j++)
            {
                QPoint point(i, j);
                if (!functionLightning.isObject(image.pixelColor(point)))
                    image.setPixelColor(point, getBackgroundColor());
            }
        }
    }

    if (isDynamic && backImage != nullptr)
    {
        for (int x = 0; x < image.width(); x++)
        {
            for (int y = 0; y < image.height(); y++)
            {
                QPoint point(x, y);
                QColor color = image.pixelColor(point);
                if(color != getBackgroundColor() && areColorsEqual(color, backImage-
>pixelColor(point)))
                {
                    image.setPixelColor(point, getBackgroundColor());
                }
            }
        }
    }

    if (backImage != nullptr)
        delete backImage;

    backImage = new QImage(imageCopy);

    return SaveImage(image, getDirToSave(), objectName);
}

void DeleteBackground::reset()
{
    level = SIM_LEVEL_DEF;
}
```

```
bool DeleteBackground::setLight(const int i, const int j, const QImage &image1, const QImage
&image2) const
{
    int count1 = 0, count2 = 0, invalidCount = 0;
    for (int iP = i - 1; iP <= i + 1; iP++)
    {
        for (int jP = j - 1; jP <= j + 1; jP++)
        {
            QPoint point(iP, jP);
            QColor pointColor1 = image1.pixelColor(point);
            QColor pointColor2 = image2.pixelColor(point);
            if (pointColor1 != QColor::Invalid)
            {
                if (pointColor1 != getBackgroundColor())
                    count1++;
                else if (pointColor2 != getBackgroundColor())
                    count2++;
                continue;
            }
            invalidCount++;
        }
    }

    if (count1 >= 3 || (count1 + count2 >= 4) || (count1 + invalidCount >= 4))
        return true;

    return false;
}

bool DeleteBackground::areColorsEqual(const QColor &color1, const QColor &color2) const
{
    int deltaRed = abs(color1.red() - color2.red());
    int deltaBlue = abs(color1.blue() - color2.blue());
    int deltaGreen = abs(color1.green() - color2.green());

    int result = (deltaRed + deltaBlue + deltaGreen) * 100 / RGB_RANGE_VALUE;
    return (result <= level);
}
```

2.11 Текст програми у файлі delete_color_channel.h

```
#ifndef DELETE_COLOR_CHANNEL_H
#define DELETE_COLOR_CHANNEL_H

#include "i_action.h"

enum class ColorChannel
{
    Red = 0,
    Green,
    Blue,
    RedAndGreen,
    GreenAndBlue,
    RedAndBlue
};

class DeleteColorChannel : public IAction
{
public:
    bool process(const QString &objectDir, const QString &objectName) override;
    void reset() override;

    void setColorChannel(ColorChannel value)
    {
        colorChannel = value;
    }

    ColorChannel getColorChannel() const
    {
        return colorChannel;
    }
```

```
    const QString getName() const override
    {
        return ACTION_NAME;
    }
private:
    ColorChannel colorChannel = COLOR_CHANNEL_DEF;

    static const ColorChannel COLOR_CHANNEL_DEF;
    static const QString ACTION_NAME;
};

#endif // DELETE_COLOR_CHANNEL_H
```

2.12 Текст програми у файлі delete_color_channel.cpp

```
#include "delete_color_channel.h"

const ColorChannel DeleteColorChannel::COLOR_CHANNEL_DEF = ColorChannel::Blue;
const QString DeleteColorChannel::ACTION_NAME = "Удаление цветового канала";

bool DeleteColorChannel::process(const QString &objectDir, const QString &objectName)
{
    QImage image = GetImage(objectDir, objectName);

    if (image.isNull())
        return false;

    int height = image.height();
    int width = image.width();

    for (int i = 0; i < width; i++)
    {
        for (int j = 0; j < height; j++)
        {
            QPoint point(i, j);
            QColor color = image.pixelColor(point);
            switch(colorChannel)
            {
            case ColorChannel::Red:
                color.setRed(0);
                break;
            case ColorChannel::Green:
                color.setGreen(0);
                break;
            case ColorChannel::Blue:
                color.setBlue(0);
                break;
            case ColorChannel::RedAndGreen:
                color.setRed(0);
                color.setGreen(0);
                break;
            case ColorChannel::GreenAndBlue:
                color.setBlue(0);
                color.setGreen(0);
                break;
            case ColorChannel::RedAndBlue:
                color.setBlue(0);
                color.setRed(0);
                break;
            }
            image.setPixelColor(point, color);
        }
    }

    return SaveImage(image, getDirToSave(), objectName);
}

void DeleteColorChannel::reset()
{
    colorChannel = COLOR_CHANNEL_DEF;
}
```

2.13 Текст програми у файлі delete_noises.h

```
#ifndef DELETE_NOISES_H
#define DELETE_NOISES_H

#include "i_action.h"

class DeleteNoises : public IAction
{
public:
    bool process(const QString &objectDir, const QString &objectName) override;
    void reset() override;

    void setCycles(int value)
    {
        cycles = value;
    }

    int getCycles() const
    {
        return cycles;
    }

    const QString getName() const override
    {
        return ACTION_NAME;
    }
private:
    int cycles = CYCLES_DEF;

    static const int CYCLES_DEF = 5;
    static const QString ACTION_NAME;

    bool toRepaint(const int i, const int j, const QImage &image) const;
};

#endif // DELETE_NOISES_H
```

2.14 Текст програми у файлі delete_noises.cpp

```
#include "delete_noises.h"

const QString DeleteNoises::ACTION_NAME = "Устранение шумов";

bool DeleteNoises::process(const QString &objectDir, const QString &objectName)
{
    QImage image = GetImage(objectDir, objectName);

    if (image.isNull())
        return false;

    int height = image.height();
    int width = image.width();

    for (int cycle = 0; cycle < cycles; cycle++)
    {
        for (int i = 0; i < width; i++)
        {
            for (int j = 0; j < height; j++)
            {
                QPoint point(i, j);
                if (image.pixelColor(point) != getBackgroundColor() && toRepaint(i, j, image))
                {
                    image.setPixelColor(point, getBackgroundColor());
                }
            }
        }
    }

    return SaveImage(image, getDirToSave(), objectName);
}
```

```
void DeleteNoises::reset()
{
    cycles = CYCLES_DEF;
}

bool DeleteNoises::toRepaint(const int i, const int j, const QImage &image) const
{
    int count = 0, countMax = 0;

    for (int iP = i - 1; iP <= i + 1; iP++)
    {
        for (int jP = j - 1; jP <= j + 1; jP++)
        {
            QPoint point(iP, jP);
            QColor pointColor = image.pixelColor(point);
            if (pointColor != QColor::Invalid)
            {
                countMax++;
                if (pointColor == getBackgroundColor())
                    count++;
            }
        }
    }

    if (count > countMax / 2)
        return true;


    return false;
}
```

2.15 Текст програми у файлі fill_empties.h

```
#ifndef FILL_EMPTIES_H
#define FILL_EMPTIES_H

#include "i_action.h"

class FillEmpties : public IAction
{
    Q_OBJECT
public:
    bool process(const QString &objectDir, const QString &objectName) override;
    void reset() override;

    void setObjectColor(const QColor &value)
    {
        objectColor = value;
    }

    const QColor& getObjectColor() const
    {
        return objectColor;
    }

    const QString getName() const override
    {
        return ACTION_NAME;
    }
private:
    QColor objectColor = OBJECT_COLOR_DEF;

    static const QColor OBJECT_COLOR_DEF;
    static const QString ACTION_NAME;

    bool toRepaint(int i, int j, const QImage &image) const;
 //   QColor getAverageColorFullNeighboursColor(int i, int j, const QImage *image) const;
};

#endif // FILL_EMPTIES_H
```

2.16 Текст програми у файлі fill_empties.cpp

```
#include "fill_empties.h"

const QColor FillEmpties::OBJECT_COLOR_DEF = QColor("white");
const QString FillEmpties::ACTION_NAME = "Дорисовка удалённых частей";

bool FillEmpties::process(const QString &objectDir, const QString &objectName)
{
    QImage image = GetImage(objectDir, objectName);

    if (image.isNull())
        return false;

    int height = image.height();
    int width = image.width();

    for (int i = 0; i < width; i++)
    {
        for (int j = 0; j < height; j++)
        {
            QPoint point(i, j);
            if (image.pixelColor(point) == getBackgroundColor() && toRepaint(i, j, image))
            {
                image.setPixelColor(point, objectColor);
            }
        }
    }

    return SaveImage(image, getDirToSave(), objectName);
}

void FillEmpties::reset()
{
    objectColor = OBJECT_COLOR_DEF;
}

bool FillEmpties::toRepaint(int i, int j, const QImage &image) const
{
    int count = 0;
    int countMax = 0;

    for (int iP = i - 1; iP <= i + 1; iP++)
    {
        for (int jP = j - 1; jP <= j + 1; jP++)
        {
            QPoint point(iP, jP);
            QColor pointColor = image.pixelColor(point);
            if (pointColor != QColor::Invalid)
            {
                countMax++;
                if (pointColor != getBackgroundColor())
                    count++;
            }
        }
    }

    if (count >= countMax / 2)
        return true;

    return false;
}
```

2.17 Текст програми у файлі find_spots_action.h

```
#ifndef FINDSPOTSACTION_H
#define FINDSPOTSACTION_H

#include <QFile>
#include <QTextStream>
```

```
#include <QImage>

#include "i_action.h"
#include "structures.h"
#include "spots/spot.h"

inline uint qHash (const QPoint & key)
{
    return qHash (static_cast <qint64> (key.x () ) << 32 | key.y () );
}

class FindSpotsAction : public IAction
{
public:
    ~FindSpotsAction();

    bool process(const QString &objectDir, const QString &objectName) override;
    void reset() override;

    void setFileName(const QString &fileName);
    void setMinSpotSize(int value)
    {
       minSpotSize = value;
    }

    int getMinSpotSize() const
    {
        return minSpotSize;
    }

    const QString getName() const override
    {
        return ACTION_NAME;
    }
private:
    void getNeighbours(QPoint p, QSet<QPoint> &spotPoins, QSet<QPoint> &visitedPoints, const
QImage &image);

    int frameNumber = 0;
    int minSpotSize = MIN_SPOT_SIZE_DEF;
    QFile file;
    QTextStream *out = nullptr;

    static const QString ACTION_NAME;
    static const int MIN_SPOT_SIZE_DEF = 25;
};

#endif // FINDSPOTSACTION_H
```

2.18 Текст програми у файлі find_spots_action.cpp

```
#include "find_spots_action.h"

const QString FindSpotsAction::ACTION_NAME = "Поиск пятен";

FindSpotsAction::~FindSpotsAction()
{
    file.close();
    delete out;
}

bool FindSpotsAction::process(const QString &objectDir, const QString &objectName)
{
    QImage image = GetImage(objectDir, objectName);
    frameNumber++;
    QList<Spot> spots;
    QSet<QPoint> visitedPoints;

    int height = image.height();
    int width = image.width();

    for (int x = 0; x < width; x++)
```

```
    {
        for (int y = 0; y < height; y++)
        {
            QPoint point(x, y);
            //if color point and no point in set
            if (image.pixelColor(point) != getBackgroundColor() && !
visitedPoints.contains(point))
            {
                QSet<QPoint> spotPoins;
                getNeighbours(point, spotPoins, visitedPoints, image);  //find poins around
point
                spots.push_back(Spot(spotPoins)); //add to list of objects
                visitedPoints += spotPoins;
            }
        }
    }

    //add to file
    foreach (Spot s, spots)
    {
        if (s.getSize() < minSpotSize)
            continue;

        *out << frameNumber << " " << s.getCenter().x() << " " << s.getCenter().y() <<
               " " << s.getCircleParms().radius << endl;
    }

    return true;
}

void FindSpotsAction::reset()
{
    if (file.isOpen())
        file.close();

    if (out != nullptr)
        delete out;
    out = nullptr;
}

void FindSpotsAction::setFileName(const QString &fileName)
{
    if (file.isOpen())
        file.close();

    file.setFileName(fileName);
    if (!file.open(QIODevice::WriteOnly | QIODevice::Text))
        return;

    if (out != nullptr)
        out = new QTextStream(&file);
}

void FindSpotsAction::getNeighbours(QPoint p, QSet<QPoint> &spotPoins,
                                    QSet<QPoint> &visitedPoints, const QImage &image)
{
    if (p.x() >= 0 && p.x() < image.width() && p.y() >= 0 && p.y() < image.height() &&
            !visitedPoints.contains(p) && image.pixelColor(p) != getBackgroundColor())
    {
        spotPoins.insert(p);
        visitedPoints.insert(p);

        getNeighbours(QPoint(p.x() + 1, p.y() + 1), spotPoins, visitedPoints, image);
        getNeighbours(QPoint(p.x() + 1, p.y()), spotPoins, visitedPoints, image);
        getNeighbours(QPoint(p.x() + 1, p.y() - 1), spotPoins, visitedPoints, image);
        getNeighbours(QPoint(p.x(), p.y() + 1), spotPoins, visitedPoints, image);
        getNeighbours(QPoint(p.x(), p.y() - 1), spotPoins, visitedPoints, image);
        getNeighbours(QPoint(p.x() - 1, p.y() + 1), spotPoins, visitedPoints, image);
        getNeighbours(QPoint(p.x() - 1, p.y()), spotPoins, visitedPoints, image);
        getNeighbours(QPoint(p.x() - 1, p.y() - 1), spotPoins, visitedPoints, image);
    }
```

```
}
```

2.19 Текст програми у файлі i_action.h

```
#ifndef I_ACTION_H
#define I_ACTION_H

#include <QImage>
#include <QSettings>
#include "structures.h"

enum class ActionCode
{
    DeleteChannel = 0,
    DeleteBackGround = 1,
    DeleteNoises = 2,
    FillEmpties = 3,
    FindSpots = 4
};

class IAction : public QObject
{
    Q_OBJECT
public:
    virtual ~IAction() = default;
    virtual bool process(const QString &objectDir, const QString &objectName) = 0;
    virtual void reset() = 0;

    static IAction* Create(ActionCode actionCode);

    ActionCode getCode() const
    {
        return actionCode;
    }

    virtual const QString getName() const = 0;

    bool getToExecute() const
    {
        return toExecute;
    }

    void setToExecute(bool value)
    {
        toExecute = value;
    }

    void setDirToSave(const QString &value)
    {
        dirToSave = value;
    }

    void setBackgroundColor(const QColor &value)
    {
        backgroundColor = value;
    }

    const QColor& getBackgroundColor() const
    {
        return backgroundColor;
    }
private:
    ActionCode actionCode;
    bool toExecute = false;
    QString dirToSave = "";
    QColor backgroundColor = BACKGROUND_COLOR_DEF;

    static const QColor BACKGROUND_COLOR_DEF;
protected:
    const QString&  getDirToSave() const
    {
        return dirToSave;
```

```
    }

    static QImage GetImage(const QString &dir, const QString &name)
    {
        return QImage(dir+"/"+name);
    }

    static bool SaveImage(const QImage &image, const QString &dir, const QString &name)
    {
        return image.save(dir + "/" + name);
    }

   // static QColor GetColor(const QSettings &settings, const QString &name, const QColor
&color);
};

#endif // I_ACTION_H
```

2.20 Текст програми у файлі i_action.cpp

```
#include "i_action.h"
#include "delete_color_channel.h"
#include "delete_background.h"
#include "delete_noises.h"
#include "fill_empties.h"
#include "find_spots_action.h"

const QColor IAction::BACKGROUND_COLOR_DEF = QColor("black");

IAction* IAction::Create(ActionCode actionCode)
{
    IAction* action = nullptr;

    switch(actionCode)
    {
    case ActionCode::DeleteChannel:
        action = new DeleteColorChannel();
        break;
    case ActionCode::FillEmpties:
        action = new FillEmpties();
        break;
    case ActionCode::DeleteBackGround:
        action = new DeleteBackground();
        break;
    case ActionCode::DeleteNoises:
        action = new DeleteNoises();
        break;
    case ActionCode::FindSpots:
        action = new FindSpotsAction();
        break;
    }

    if (action != nullptr)
        action->actionCode = actionCode;

    return action;
}
```

2.21 Текст програми у файлі color_params_form.h

```
#ifndef COLOR_PARAMS_FORM_H
#define COLOR_PARAMS_FORM_H

#include "structures.h"

#include <QDialog>
#include <QGridLayout>
#include <QLabel>
#include <QSettings>

namespace Ui {
class ColorParamsForm;
```

```
}

class ColorParamsForm : public QDialog {
    Q_OBJECT
public:
    ColorParamsForm(const QString &companyName, const QString &softName, const QString
&groupName, QWidget *parent = nullptr);
    ~ColorParamsForm();

signals:
    void sendColorParms(const ParmsLab &, const ParmsLCH &);

private slots:
    void on_pB_ok_clicked();

    void changeLowerLValue(int lowerValue);
    void changeLowerAValue(int lowerValue);
    void changeLowerBValue(int lowerValue);

    void changeLowerL2Value(int lowerValue);
    void changeLowerCValue(int lowerValue);
    void changeLowerHValue(int lowerValue);

    void changeUpperLValue(int upperValue);
    void changeUpperAValue(int upperValue);
    void changeUpperBValue(int upperValue);

    void changeUpperL2Value(int upperValue);
    void changeUpperCValue(int upperValue);
    void changeUpperHValue(int upperValue);

    void on_hS_L_rangeChanged(int min, int max);
    void on_hS_a_rangeChanged(int min, int max);
    void on_hS_b_rangeChanged(int min, int max);

    void on_cB_L_stateChanged(int arg1);
    void on_cB_a_stateChanged(int arg1);
    void on_cB_b_stateChanged(int arg1);

    void on_hS_L_2_rangeChanged(int min, int max);
    void on_hS_C_rangeChanged(int min, int max);
    void on_hS_H_rangeChanged(int min, int max);

    void on_cB_L_2_stateChanged(int arg1);
    void on_cB_C_stateChanged(int arg1);
    void on_cB_H_stateChanged(int arg1);
    void on_pB_cansel_clicked();

private:
    Ui::ColorParamsForm *ui;
    ParmsLab parmsLab;
    ParmsLCH parmsLCH;
    const QString CompanyName;
    const QString SoftName;
    const QString GroupName;

    void updateLabelText(int l, int u, QLabel *label);
    void getValueFromSettings();
    void saveValueToSettings();
};

#endif // COLOR_PARAMS_FORM_H
```

2.22 Текст програми у файлі color_params_form.cpp

```
#include "color_params_form.h"
#include "ui_color_params_form.h"

ColorParamsForm::ColorParamsForm(const QString &companyName, const QString &softName, const
QString &groupName, QWidget *parent) :
    QDialog(parent),
    ui(new Ui::ColorParamsForm),
```

```
    CompanyName(companyName),
    SoftName(softName),
    GroupName(groupName)
{
    ui->setupUi(this);
    getValueFromSettings();

    ui->hS_L->SetRange(0, 100);
    ui->hS_L->setSingleStep(1);
    ui->hS_a->SetRange(-128, 127);
    ui->hS_a->setSingleStep(1);
    ui->hS_b->SetRange(-128, 127);
    ui->hS_b->setSingleStep(1);
    ui->l_L->setVisible(parmsLab.isL);
    ui->l_a->setVisible(parmsLab.isA);
    ui->l_b->setVisible(parmsLab.isB);
    ui->cB_L->setCheckState((parmsLab.isL) ? Qt::CheckState::Checked :
Qt::CheckState::Unchecked);
    ui->cB_a->setCheckState((parmsLab.isA) ? Qt::CheckState::Checked :
Qt::CheckState::Unchecked);
    ui->cB_b->setCheckState((parmsLab.isB) ? Qt::CheckState::Checked :
Qt::CheckState::Unchecked);
    ui->hS_L_2->SetRange(0, 100);
    ui->hS_L_2->setSingleStep(1);
    ui->hS_C->SetRange(0, 100);
    ui->hS_C->setSingleStep(1);
    ui->hS_H->SetRange(0, 359);
    ui->hS_H->setSingleStep(1);
    ui->l_L_2->setVisible(parmsLCH.isL);
    ui->l_C->setVisible(parmsLCH.isC);
    ui->l_H->setVisible(parmsLCH.isH);
    ui->cB_L_2->setCheckState((parmsLCH.isL) ? Qt::CheckState::Checked :
Qt::CheckState::Unchecked);
    ui->cB_C->setCheckState((parmsLCH.isC) ? Qt::CheckState::Checked :
Qt::CheckState::Unchecked);
    ui->cB_H->setCheckState((parmsLCH.isH) ? Qt::CheckState::Checked :
Qt::CheckState::Unchecked);

    if (parmsLab.isL)
    {
        ui->hS_L->SetLowerValue(parmsLab.fromL);
        ui->hS_L->setUpperValue(parmsLab.toL);
        ui->hS_L_2->SetLowerValue(parmsLab.fromL);
        ui->hS_L_2->setUpperValue(parmsLab.toL);
    }

    if (parmsLab.isA)
    {
        ui->hS_a->SetLowerValue(parmsLab.fromA);
        ui->hS_a->setUpperValue(parmsLab.toA);
    }

    if (parmsLab.isB)
    {
        ui->hS_b->SetLowerValue(parmsLab.fromB);
        ui->hS_b->setUpperValue(parmsLab.toB);
    }

    if (parmsLCH.isC)
    {
        ui->hS_C->SetLowerValue(parmsLCH.fromC);
        ui->hS_C->setUpperValue(parmsLCH.toC);
    }

    if (parmsLCH.isH)
    {
        ui->hS_H->SetLowerValue(parmsLCH.fromH);
        ui->hS_H->setUpperValue(parmsLCH.toH);
    }

    connect(ui->hS_L, SIGNAL(lowerValueChanged(int)), this, SLOT(changeLowerLValue(int)));
```

```
    connect(ui->hS_L, SIGNAL(upperValueChanged(int)), this, SLOT(changeUpperLValue(int)));
    connect(ui->hS_a, SIGNAL(lowerValueChanged(int)), this, SLOT(changeLowerAValue(int)));
    connect(ui->hS_a, SIGNAL(upperValueChanged(int)), this, SLOT(changeUpperAValue(int)));
    connect(ui->hS_b, SIGNAL(lowerValueChanged(int)), this, SLOT(changeLowerBValue(int)));
    connect(ui->hS_b, SIGNAL(upperValueChanged(int)), this, SLOT(changeUpperBValue(int)));
    connect(ui->hS_L_2, SIGNAL(lowerValueChanged(int)), this, SLOT(changeLowerL2Value(int)));
    connect(ui->hS_L_2, SIGNAL(upperValueChanged(int)), this, SLOT(changeUpperL2Value(int)));
    connect(ui->hS_C, SIGNAL(lowerValueChanged(int)), this, SLOT(changeLowerCValue(int)));
    connect(ui->hS_C, SIGNAL(upperValueChanged(int)), this, SLOT(changeUpperCValue(int)));
    connect(ui->hS_H, SIGNAL(lowerValueChanged(int)), this, SLOT(changeLowerHValue(int)));
    connect(ui->hS_H, SIGNAL(upperValueChanged(int)), this, SLOT(changeUpperHValue(int)));
}

ColorParamsForm::~ColorParamsForm()
{
    delete ui;
}

void ColorParamsForm::on_pB_ok_clicked()
{
    parmsLab.isL = ui->cB_L->isChecked();
    parmsLab.fromL = ui->hS_L->GetLowerValue();
    parmsLab.toL = ui->hS_L->GetUpperValue();
    parmsLab.isA = ui->cB_a->isChecked();
    parmsLab.fromA = ui->hS_a->GetLowerValue();
    parmsLab.toA = ui->hS_a->GetUpperValue();
    parmsLab.isB = ui->cB_b->isChecked();
    parmsLab.fromB = ui->hS_b->GetLowerValue();
    parmsLab.toB = ui->hS_b->GetUpperValue();

    parmsLCH.isL = ui->cB_L_2->isChecked();
    parmsLCH.fromL = ui->hS_L_2->GetLowerValue();
    parmsLCH.toL = ui->hS_L_2->GetUpperValue();
    parmsLCH.isC = ui->cB_C->isChecked();
    parmsLCH.fromC = ui->hS_C->GetLowerValue();
    parmsLCH.toC = ui->hS_C->GetUpperValue();
    parmsLCH.isH = ui->cB_H->isChecked();
    parmsLCH.fromH = ui->hS_H->GetLowerValue();
    parmsLCH.toH = ui->hS_H->GetUpperValue();

    saveValueToSettings();
    emit sendColorParms(parmsLab, parmsLCH);

    this->close();
}

void ColorParamsForm::on_hS_L_rangeChanged(int min, int max)
{
    updateLabelText(min, max, ui->l_L);
}

void ColorParamsForm::changeLowerLValue(int lowerValue)
{
    updateLabelText(lowerValue, ui->hS_L->GetUpperValue(), ui->l_L);
    if (ui->hS_L_2->GetLowerValue() != lowerValue)
         ui->hS_L_2->SetLowerValue(lowerValue);
}

void ColorParamsForm::changeLowerAValue(int lowerValue)
{
     updateLabelText(lowerValue, ui->hS_a->GetUpperValue(), ui->l_a);
}

void ColorParamsForm::changeLowerBValue(int lowerValue)
{
    updateLabelText(lowerValue, ui->hS_b->GetUpperValue(), ui->l_b);
}

void ColorParamsForm::changeLowerL2Value(int lowerValue)
{
    updateLabelText(lowerValue, ui->hS_L_2->GetUpperValue(), ui->l_L_2);
```

```
    if (ui->hS_L->GetLowerValue() != lowerValue)
         ui->hS_L->SetLowerValue(lowerValue);
}

void ColorParamsForm::changeLowerCValue(int lowerValue)
{
    updateLabelText(lowerValue, ui->hS_C->GetUpperValue(), ui->l_C);
}

void ColorParamsForm::changeLowerHValue(int lowerValue)
{
    updateLabelText(lowerValue, ui->hS_H->GetUpperValue(), ui->l_H);
}

void ColorParamsForm::changeUpperLValue(int upperValue)
{
    updateLabelText(ui->hS_L->GetLowerValue(), upperValue, ui->l_L);
    if (ui->hS_L_2->GetUpperValue() != upperValue)
         ui->hS_L_2->SetUpperValue(upperValue);
}

void ColorParamsForm::changeUpperAValue(int upperValue)
{
    updateLabelText(ui->hS_a->GetLowerValue(), upperValue, ui->l_a);
}

void ColorParamsForm::changeUpperBValue(int upperValue)
{
    updateLabelText(ui->hS_b->GetLowerValue(), upperValue, ui->l_b);
}

void ColorParamsForm::changeUpperL2Value(int upperValue)
{
    updateLabelText(ui->hS_L_2->GetLowerValue(), upperValue, ui->l_L_2);
    if (ui->hS_L->GetUpperValue() != upperValue)
         ui->hS_L->SetUpperValue(upperValue);
}

void ColorParamsForm::changeUpperCValue(int upperValue)
{
    updateLabelText(ui->hS_C->GetLowerValue(), upperValue, ui->l_C);
}

void ColorParamsForm::changeUpperHValue(int upperValue)
{
    updateLabelText(ui->hS_H->GetLowerValue(), upperValue, ui->l_H);
}

void ColorParamsForm::updateLabelText(int l, int u, QLabel *label)
{
    label->setText(QString::number(l)+" ... "+QString::number(u));
}

void ColorParamsForm::getValueFromSettings()
{
    QSettings settings(CompanyName, SoftName);

    settings.beginGroup(GroupName);
    parmsLCH.isL = parmsLab.isL = settings.value("labIsL", false).toBool();
    parmsLab.isA = settings.value("labIsA", false).toBool();
    parmsLab.isB = settings.value("labIsB", false).toBool();
    parmsLCH.fromL = parmsLab.fromL = settings.value("labFromL", 0).toInt();
    parmsLab.fromA = settings.value("labFromA", -127).toInt();
    parmsLab.fromB = settings.value("labFromB", -127).toInt();
    parmsLCH.toL = parmsLab.toL = settings.value("labToL", 100).toInt();
    parmsLab.toA = settings.value("labToA", 128).toInt();
    parmsLab.toB = settings.value("labToB", 128).toInt();
    parmsLCH.isC = settings.value("lchIsC", false).toBool();
    parmsLCH.isH = settings.value("lchIsH", false).toBool();
    parmsLCH.fromC = settings.value("lchFromC", 0).toInt();
    parmsLCH.fromH = settings.value("lchFromH", 0).toInt();
```

```
    parmsLCH.toC = settings.value("lchToC", 100).toInt();
    parmsLCH.toH = settings.value("lchToH", 359).toInt();
    settings.endGroup();
}

void ColorParamsForm::saveValueToSettings()
{
    QSettings settings(CompanyName, SoftName);
    settings.beginGroup(GroupName);
    settings.setValue("labIsL", parmsLab.isL);
    settings.setValue("labIsA", parmsLab.isA);
    settings.setValue("labIsB", parmsLab.isB);
    settings.setValue("labFromL", parmsLab.fromL);
    settings.setValue("labFromA", parmsLab.fromA);
    settings.setValue("labFromB", parmsLab.fromB);
    settings.setValue("labToL", parmsLab.toL);
    settings.setValue("labToA", parmsLab.toA);
    settings.setValue("labToB", parmsLab.toB);
    settings.setValue("lchIsC", parmsLCH.isC);
    settings.setValue("lchIsH", parmsLCH.isH);
    settings.setValue("lchFromC", parmsLCH.fromC);
    settings.setValue("lchFromH", parmsLCH.fromH);
    settings.setValue("lchToC", parmsLCH.toC);
    settings.setValue("lchToH", parmsLCH.toH);
    settings.endGroup();
}

void ColorParamsForm::on_hS_a_rangeChanged(int min, int max)
{
    updateLabelText(min, max, ui->l_a);
}

void ColorParamsForm::on_hS_b_rangeChanged(int min, int max)
{
    updateLabelText(min, max, ui->l_b);
}

void ColorParamsForm::on_cB_L_stateChanged(int arg1)
{
    ui->hS_L->setDisabled(ui->hS_L->isEnabled());
    ui->l_L->setVisible(ui->hS_L->isEnabled());
    if (ui->cB_L_2->checkState() != ui->cB_L->checkState())
    {
        ui->cB_L_2->setCheckState(
                    (arg1 == 0) ? Qt::CheckState::Unchecked : Qt::CheckState::Checked);
    }
}

void ColorParamsForm::on_cB_a_stateChanged(int )
{
    ui->hS_a->setDisabled(ui->hS_a->isEnabled());
    ui->l_a->setVisible(ui->hS_a->isEnabled());
}

void ColorParamsForm::on_cB_b_stateChanged(int )
{
    ui->hS_b->setDisabled(ui->hS_b->isEnabled());
    ui->l_b->setVisible(ui->hS_b->isEnabled());
}

void ColorParamsForm::on_hS_L_2_rangeChanged(int min, int max)
{
    updateLabelText(min, max, ui->l_L_2);
}

void ColorParamsForm::on_hS_C_rangeChanged(int min, int max)
{
    updateLabelText(min, max, ui->l_C);
}

void ColorParamsForm::on_hS_H_rangeChanged(int min, int max)
```

```
{
     updateLabelText(min, max, ui->l_H);
}

void ColorParamsForm::on_cB_L_2_stateChanged(int arg1)
{
    ui->hS_L_2->setDisabled(ui->hS_L_2->isEnabled());
    ui->l_L_2->setVisible(ui->hS_L_2->isEnabled());
    if (ui->cB_L_2->checkState() != ui->cB_L->checkState())
    {
        ui->cB_L->setCheckState(
                    (arg1 == 0) ? Qt::CheckState::Unchecked : Qt::CheckState::Checked);
    }
}

void ColorParamsForm::on_cB_C_stateChanged(int )
{
    ui->hS_C->setDisabled(ui->hS_C->isEnabled());
    ui->l_C->setVisible(ui->hS_C->isEnabled());
}

void ColorParamsForm::on_cB_H_stateChanged(int )
{
    ui->hS_H->setDisabled(ui->hS_H->isEnabled());
    ui->l_H->setVisible(ui->hS_H->isEnabled());
}

void ColorParamsForm::on_pB_cansel_clicked()
{
    this->close();
}
```

2.23 Текст програми у файлі delete_background_dialog.h

```
#ifndef DELETE_BACKGROUND_DIALOG_H
#define DELETE_BACKGROUND_DIALOG_H

#include <QDialog>
#include <QColorDialog>
#include <QSettings>

#include "actions/delete_background.h"
#include "color_params_form.h"

namespace Ui {
class DeleteBackgroundDialog;
}

class DeleteBackgroundDialog : public QDialog
{
    Q_OBJECT

public:
    DeleteBackgroundDialog(const QString &companyName, const QString &softName,
DeleteBackground *action, QWidget *parent = nullptr);
    ~DeleteBackgroundDialog();
private slots:
    void on_pushButton_ok_clicked();
    void on_pushButton_cancel_clicked();

    void on_pushButton_contourParms_clicked();
    void on_pushButton_lightningParms_clicked();

    void on_checkBox_isDynamic_stateChanged(int arg1);
    void on_groupBox_contour_clicked(bool checked);
    void on_groupBox_lightning_clicked(bool checked);
private:
    Ui::DeleteBackgroundDialog *ui;
    const QString &companyName;
    const QString &softName;
    static const QString CONTOUR_PARMS_NAME;
    static const QString LIGHTNING_PARMS_NAME;
```

```
    QSettings settings;
    DeleteBackground *action;
};

#endif // DELETE_BACKGROUND_DIALOG_H
```

2.24 Текст програми у файлі delete_background_dialog.cpp

```
#include "delete_background_dialog.h"
#include "ui_delete_background_dialog.h"

const QString DeleteBackgroundDialog::CONTOUR_PARMS_NAME = "ContourDFParms";
const QString DeleteBackgroundDialog::LIGHTNING_PARMS_NAME = "LightningDFParms";

DeleteBackgroundDialog::DeleteBackgroundDialog(const QString &companyName, const QString
&softName, DeleteBackground *action, QWidget *parent) :
    QDialog(parent),
    ui(new Ui::DeleteBackgroundDialog),
    companyName(companyName),
    softName(softName),
    settings(companyName, softName),
    action(action)
{
    ui->setupUi(this);


    ui->checkBox_isDynamic->setChecked(settings.value("isDynamic", action-
>getIsDynamic()).toBool());
    ui->spinBox_backgroundLevel->setEnabled(ui->checkBox_isDynamic->isChecked());
    ui->spinBox_backgroundLevel->setValue(settings.value("dynamicLevel", action-
>getLevel()).toInt());

    ui->groupBox_lightning->setChecked(settings.value("funcLightning", action-
>getFunctionLightning().getToUse()).toBool());
    ui->groupBox_contour->setChecked(settings.value("funcContour", action-
>getFunctionContour().getToUse()).toBool());

    ui->radioButton_linearDF->setChecked(settings.value("contourIsLinear", action-
>getFunctionContour().getType() == DecisiveFunctionType::Linear).toBool());
    ui->radioButton_quadraticDF->setChecked(!ui->radioButton_linearDF->isChecked());
    ui->radioButton_linearDFLightning->setChecked(settings.value("lightningIsLinear", action-
>getFunctionLightning().getType() == DecisiveFunctionType::Linear).toBool());
    ui->radioButton_quadraticDFLighnt->setChecked(!ui->radioButton_linearDFLightning-
>isChecked());
}

DeleteBackgroundDialog::~DeleteBackgroundDialog()
{
    delete ui;
}

void DeleteBackgroundDialog::on_pushButton_ok_clicked()
{
    action->setIsDynamic(ui->checkBox_isDynamic->isChecked());
    action->setLevel(ui->spinBox_backgroundLevel->value());
    action->getFunctionLightning().setToUse(ui->groupBox_lightning->isChecked());
    action->getFunctionContour().setToUse(ui->groupBox_contour->isChecked());
    action->getFunctionContour().setType(ui->radioButton_linearDF->isChecked()
                                         ? DecisiveFunctionType::Linear :
DecisiveFunctionType::Quadratic);
    action->getFunctionLightning().setType(ui->radioButton_linearDFLightning->isChecked()
                                         ? DecisiveFunctionType::Linear :
DecisiveFunctionType::Quadratic);

    settings.setValue("isDynamic", action->getIsDynamic());
    settings.setValue("dynamicLevel", action->getLevel());
    settings.setValue("funcLightning", action->getFunctionLightning().getToUse());
    settings.setValue("funcContour", action->getFunctionContour().getToUse());
    settings.setValue("contourIsLinear", action->getFunctionContour().getType() ==
DecisiveFunctionType::Linear);
    settings.setValue("lightningIsLinear", action->getFunctionLightning().getType() ==
DecisiveFunctionType::Linear);
```

```
    this->close();
}

void DeleteBackgroundDialog::on_pushButton_cancel_clicked()
{
    this->close();
}

void DeleteBackgroundDialog::on_pushButton_contourParms_clicked()
{
    ColorParamsForm *colorParmsForm = new ColorParamsForm(companyName, softName,
CONTOUR_PARMS_NAME);

    connect(colorParmsForm, SIGNAL(sendColorParms(const ParmsLab &, const ParmsLCH &)),
            &(action->getFunctionContour()), SLOT(updateCriterias(const ParmsLab &, const
ParmsLCH &)));
    colorParmsForm->show();
}

void DeleteBackgroundDialog::on_pushButton_lightningParms_clicked()
{
    ColorParamsForm *colorParmsForm = new ColorParamsForm(companyName, softName,
LIGHTNING_PARMS_NAME);

    connect(colorParmsForm, SIGNAL(sendColorParms(const ParmsLab &, const ParmsLCH &)),
            &(action->getFunctionLightning()), SLOT(updateCriterias(const ParmsLab &, const
ParmsLCH &)));
    colorParmsForm->show();
}

void DeleteBackgroundDialog::on_checkBox_isDynamic_stateChanged(int arg1)
{
    if (arg1 == 0 && !ui->groupBox_contour->isChecked() && !ui->groupBox_lightning-
>isChecked())
    {
        ui->checkBox_isDynamic->setCheckState(Qt::CheckState::Checked);
        return;
    }
    ui->spinBox_backgroundLevel->setEnabled(arg1 != 0);
}

void DeleteBackgroundDialog::on_groupBox_contour_clicked(bool checked)
{
    if (!checked && !ui->checkBox_isDynamic->isChecked() && !ui->groupBox_lightning-
>isChecked())
    {
        ui->groupBox_contour->setChecked(true);
        return;
    }
}

void DeleteBackgroundDialog::on_groupBox_lightning_clicked(bool checked)
{
    if (!checked && !ui->checkBox_isDynamic->isChecked() && !ui->groupBox_contour-
>isChecked())
    {
        ui->groupBox_lightning->setChecked(true);
        return;
    }
}
```

2.25 Текст програми у файлі image_sourse_form.h

```
#ifndef IMAGE_SOURSE_FORM_H
#define IMAGE_SOURSE_FORM_H

#include <QWidget>
#include <QFileDialog>
#include <QMessageBox>
#include <QVector>
#include <QSettings>
```

```
#include <utils/i_name_generator.h>
#include <utils/fixed_name_generator.h>
#include <utils/simple_name_generator.h>

#include "structures.h"

namespace Ui {
class ImageSourseForm;
}

class ImageSourseForm : public QWidget {
    Q_OBJECT
public:
    explicit ImageSourseForm(const QString &companyName, const QString &softName, QWidget
*parent = nullptr);
    ~ImageSourseForm();
signals:
    void sendSourceData(Source*);

private slots:
    void on_pBSelectInputDir_clicked();
    void on_cBFormatType_currentIndexChanged(int index);
    void on_pushButton_clicked();
    void on_pB_cansel_clicked();
private:
    Ui::ImageSourseForm *ui;
    QSettings settings;

    void receiveParameters();
    void saveParameters();

    void setFieldValue(const int groupId, const int fieldId, const QVariant& value);
    QVariant getFieldValue(const int groupId, const int fieldId);

    void hideShowFixedIndexPams(bool toShow);
    bool checkForErrors(QString &errors);

    static const int fixedName = 0;
    static const int fromImages = 0;

    static const int rangeFiedlsGroupId = 0;
    static const int nameTypeFieldsGroupId = 1;
    static const int pathFieldsGroupId = 3;

    static const int nameFormatFieldId = 0;
    static const int prefixFieldId = 1;
    static const int resolutionFieldId = 2;
    static const int digitsInNumberFieldId = 3;
    static const int valueFieldId = 4;

    static const int processFromFieldId = 0;
    static const int processToFieldId = 1;

    static const int pathFieldId = 0;

    const QString imageDirName = "ImageDir";
    const QStringList groupNames = {"range", "nametype", "path"};
    const QStringList rangeFieldNames = {"processfrom", "processto"};
    const QStringList nameTypeFieldNames = {"nameformatid", "prefix", "resolutionid",
"digitsinnumber", "value"};
    const QStringList pathFieldNames = {"pathvalue"};
    const QList<QStringList> groupFieldNames = {rangeFieldNames, nameTypeFieldNames,
pathFieldNames};

    const QList<QVariant> defaultRangeFieldValues = {0, 0};
    const QList<QVariant> defaultNameTypeFieldValues = {0, "images", 0, 5, "0"};
    const QList<QVariant> defaultPathFieldValues = {"/home"};
    const QList<QPair<QStringList, QList<QVariant>>> namesToSefaultGroupValues = {
        { rangeFieldNames, defaultRangeFieldValues },
        { nameTypeFieldNames, defaultNameTypeFieldValues},
```

```
        { pathFieldNames, defaultPathFieldValues} };
};

#endif // IMAGE_SOURSE_FORM_H
```

2.26 Текст програми у файлі image_sourse_form.cpp

```
#include "image_sourse_form.h"
#include "ui_imagesourseform.h"

#include <QSettings>

ImageSourseForm::ImageSourseForm(const QString &companyName, const QString &softName, QWidget
*parent) :
    QWidget(parent), ui(new Ui::ImageSourseForm), settings(companyName, softName)
{
    ui->setupUi(this);
    receiveParameters();
}

ImageSourseForm::~ImageSourseForm()
{
    delete ui;
}

void ImageSourseForm::on_pBSelectInputDir_clicked()
{
    QString lastDir = settings.value(imageDirName, "/home").toString();

    QString dir = QFileDialog::getExistingDirectory(this, tr("Open Directory"), lastDir,
                                                    QFileDialog::ShowDirsOnly |
QFileDialog::DontResolveSymlinks);
    if (dir.length())
    {
        ui->tBrInputPath->setText(dir);
        settings.setValue(imageDirName, dir);
    }
}

void ImageSourseForm::on_cBFormatType_currentIndexChanged(int index)
{
    hideShowFixedIndexPams(index == fixedName);
}

void ImageSourseForm::on_pushButton_clicked()
{
    QString errors = "";

    if (checkForErrors(errors))
    {
        QMessageBox msgBox;
        msgBox.setText(errors);
        msgBox.exec();
        return;
    }

    Source *source = new Source();
    source->from = getFieldValue(rangeFiedlsGroupId, processFromFieldId).toInt();
    source->to = getFieldValue(rangeFiedlsGroupId, processToFieldId).toInt();

    if (ui->cBFormatType->currentIndex() == fixedName)
        source->nameGenerator = new FixedNameGenerator(getFieldValue(nameTypeFieldsGroupId,
digitsInNumberFieldId).toInt(),
                                                       getFieldValue(nameTypeFieldsGroupId,
valueFieldId).toString().at(0),
                                                       getFieldValue(nameTypeFieldsGroupId,
prefixFieldId).toString(),
                                                       getFieldValue(nameTypeFieldsGroupId,
resolutionFieldId).toString(),
                                                       getFieldValue(pathFieldsGroupId,
pathFieldId).toString());
    else
```

```
        source->nameGenerator = new SimpleNameGenerator(getFieldValue(nameTypeFieldsGroupId,
prefixFieldId).toString(),

getFieldValue(nameTypeFieldsGroupId, resolutionFieldId).toString(),

getFieldValue(pathFieldsGroupId, pathFieldId).toString());

    saveParameters();
    emit sendSourceData(source);
    this->close();
}

void ImageSourseForm::on_pB_cansel_clicked()
{
    saveParameters();
    this->close();
}

void ImageSourseForm::receiveParameters()
{
    for (int i = 0; i < groupNames.size(); i++)
    {
         settings.beginGroup(groupNames.at(i));
         for(int j = 0; j < namesToSefaultGroupValues[i].first.size(); j++)
            setFieldValue(i, j, settings.value(namesToSefaultGroupValues[i].first[j],
namesToSefaultGroupValues[i].second[j]));

         settings.endGroup();
    }
}

void ImageSourseForm::saveParameters()
{
    for (int i = 0; i < groupNames.size(); i++)
    {
         settings.beginGroup(groupNames.at(i));
         for(int j = 0; j < groupFieldNames[i].size(); j++)
            settings.setValue(groupFieldNames[i][j], getFieldValue(i, j));

         settings.endGroup();
    }
}

void ImageSourseForm::setFieldValue(const int groupId, const int fieldId, const QVariant&
value)
{
    switch(groupId)
    {
    case rangeFiedlsGroupId:
        switch(fieldId)
        {
        case processFromFieldId:
            ui->sBFrom->setValue(value.toInt());
            break;
        case processToFieldId:
            ui->sBTo->setValue(value.toInt());
            break;
        }
        break;
    case nameTypeFieldsGroupId:
        switch(fieldId)
        {
        case nameFormatFieldId:
            ui->cBFormatType->setCurrentIndex(value.toInt());
            hideShowFixedIndexPams(value.toInt() == fixedName);
            break;
        case prefixFieldId:
            ui->pTEPrefix->setPlainText(value.toString());
            break;
        case resolutionFieldId:
            ui->cBInputResolution->setCurrentText(value.toString());
```

```
            break;
        case valueFieldId:
            ui->pTEValue->setPlainText(value.toString());
            break;
        case digitsInNumberFieldId:
            ui->sBDigits->setValue(value.toInt());
            break;
        }
        break;
    case pathFieldsGroupId:
        switch(fieldId)
        {
        case pathFieldId:
            ui->tBrInputPath->setText(value.toString());
            break;
        }
        break;
    }
}

QVariant ImageSourseForm::getFieldValue(const int groupId, const int fieldId)
{
    QVariant result;
    switch(groupId) {
    case rangeFiedlsGroupId:
        switch(fieldId)
        {
        case processFromFieldId:
            result = ui->sBFrom->value();
            break;
        case processToFieldId:
            result = ui->sBTo->value();
            break;
        }
        break;
    case nameTypeFieldsGroupId:
        switch(fieldId)
        {
        case nameFormatFieldId:
            result = ui->cBFormatType->currentIndex();
            break;
        case prefixFieldId:
            result = ui->pTEPrefix->toPlainText();
            break;
        case resolutionFieldId:
            result = ui->cBInputResolution->currentText();
            break;
        case valueFieldId:
            result = ui->pTEValue->toPlainText();
            break;
        case digitsInNumberFieldId:
            result = ui->sBDigits->value();
            break;
        }
        break;
    case pathFieldsGroupId:
        switch(fieldId)
        {
        case pathFieldId:
            result = ui->tBrInputPath->toPlainText();
            break;
        }
        break;
    }
    return result;
}

void ImageSourseForm::hideShowFixedIndexPams(bool toShow)
{
    ui->labelDidgits->setVisible(toShow);
    ui->labelValue->setVisible(toShow);
```

```
    ui->sBDigits->setVisible(toShow);
    ui->pTEValue->setVisible(toShow);
}

bool ImageSourseForm::checkForErrors(QString& errors)
{
    if (errors.isNull())
        errors = "";

    if (ui->tBrInputPath->toPlainText().length() == 0)
    {
         errors += "Укажите, пожалуйста, путь к папке с входными изображениями. \n";
    }

    if (ui->cBFormatType->currentIndex() == fixedName)
    {
        if (ui->pTEValue->toPlainText().length() == 0)
            errors += "Поле заполнение не может быть пустым. \n";

        if (ui->pTEValue->toPlainText().length() > 1)
            errors += "Длина поля заполнение не может быть больше 1 символа. \n";
    }
    if (ui->sBFrom->value() > ui->sBTo->value())
        errors += "Номер начала должен быть меньше, чем номер конца. \n";

    return errors.length() != 0;
}
```

2.27 Текст програми у файлі drawablelabel.h

```
#ifndef DRAWABLELABEL_H
#define DRAWABLELABEL_H

#include <QLabel>
#include <QPixmap>
#include <QPainter>
#include <QMouseEvent>
#include <QKeyEvent>
#include <QtMath>
#include <QStack>
#include <QRandomGenerator>

#include "sequencegenerator.h"

enum class SpetialSign {
    RotateRight = '-',
    RotateLeft = '+',
    StartBranch = '[',
    EndBranch = ']',
    ChangeAngle = '%',
    ChangeLength = '#'
};

class DrawableLabel : public QLabel
{
    Q_OBJECT
public:
    explicit DrawableLabel(QWidget *parent = nullptr);

    void draw(const QString &rule, const QHash<QChar, NTerminalDescriptor> &descriptor);

    void zoomIn();
    void zoomOut();

    void setAlphaExpected(double value);
    void setAlphaDispertion(double value);

    void setIsDrawMode(bool value);

protected:
    virtual void mousePressEvent(QMouseEvent *ev);
signals:
```

```
private:
    double alpha;
    double alphaExpected;
    double alphaDispertion;

    bool isDrawMode = false;

    void generateNewAlpha();



    QPoint startPoint;




    void recalculateStartPoint();
    void drawNewPoint();
    QPainter *painter;
    QPixmap *drawPixmap;
    int scaleK;
    QSize actualSize;

    void scale();
};

#endif // DRAWABLELABEL_H
```

2.28 Текст програми у файлі drawablelabel.cpp

```
#include "drawablelabel.h"

DrawableLabel::DrawableLabel(QWidget *parent) : QLabel(parent)
{
    this->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);
    this->setBackgroundRole(QPalette::Light);
    this->resize(620, 500);
    actualSize = this->size();
    startPoint.setX(310);
    startPoint.setY(50);
    scaleK = 1;
    drawPixmap = new QPixmap(this->size());
    painter = new QPainter(drawPixmap);
    drawNewPoint();
}

void DrawableLabel::draw(const QString &rule, const QHash<QChar, NTerminalDescriptor>
&descriptors) {
    drawPixmap->fill();
    double angle = qDegreesToRadians(270.0);
    QPointF currentPoint = startPoint;
    QStack<QPair<QPointF, double>> settings;
    foreach (QChar sign, rule) {
        SpetialSign spSign = static_cast<SpetialSign>(sign.unicode());
        switch (spSign) {
        case SpetialSign::RotateRight:
            angle -= alpha;
            break;
        case SpetialSign::RotateLeft:
            angle += alpha;
            break;
        case SpetialSign::StartBranch:
            settings.push(QPair<QPointF, double>(currentPoint, angle));
            break;
        case SpetialSign::EndBranch:
            currentPoint = settings.top().first;
            angle = settings.pop().second;
            break;
        case SpetialSign::ChangeAngle:
            generateNewAlpha();
            break;
```

```
        default:
            NTerminalDescriptor descriptor = descriptors.value(sign, NTerminalDescriptor());
//            int length = QRandomGenerator::global()->bounded(descriptor.lExpected -
descriptor.lDispertion,
//                                                             descriptor.lExpected +
descriptor.lDispertion);
            int length = descriptor.lExpected;
            QPointF nextPoint(currentPoint.x() + length * qCos(angle), currentPoint.y() -
length * qSin(angle));
            if (sign >= 'A' && sign <= 'Z') {
                painter->setPen(QPen(QBrush("black"), descriptor.width));
                painter->drawLine(currentPoint, nextPoint);
            }
            currentPoint = nextPoint;
            break;
        }
    }
    scale();
}

void DrawableLabel::zoomIn() {
    if (scaleK == 5) {
        return;
    }
    scaleK++;
    scale();
}

void DrawableLabel::zoomOut() {
    if (scaleK == 1) {
        return;
    }
    scaleK--;
    scale();
}

void DrawableLabel::scale() {
    this->setPixmap(drawPixmap->scaled(drawPixmap->size()*scaleK));
    this->resize(drawPixmap->size()*scaleK);
}

void DrawableLabel::mousePressEvent(QMouseEvent *ev) {
    startPoint = ev->pos()/scaleK;
    drawPixmap->fill();
    drawNewPoint();
}

void DrawableLabel::setIsDrawMode(bool value)
{
    isDrawMode = value;
    drawPixmap->fill();
    drawNewPoint();
}

void DrawableLabel::setAlphaDispertion(double value)
{
    alphaDispertion = qDegreesToRadians(value);
}

void DrawableLabel::setAlphaExpected(double value)
{
    alphaExpected = qDegreesToRadians(value);
    alpha = alphaExpected;
}

void DrawableLabel::generateNewAlpha()
{
//    alpha = QRandomGenerator::global()->bounded(alphaExpected - alphaDispertion,
alphaExpected + alphaDispertion)
//            + QRandomGenerator::global()->generateDouble();
}
```

```
void DrawableLabel::recalculateStartPoint() {
    if (this->width() < startPoint.x()) {
        startPoint.setX(this->width() / 2);
    }
    if (this->height() < startPoint.y()) {
        startPoint.setY(this->height() / 10);
    }
    drawNewPoint();
}

void  DrawableLabel::drawNewPoint() {
    drawPixmap->fill();
    painter->setBrush(QBrush("red"));
    painter->drawEllipse(startPoint, 6, 6);
 //   this->setPixmap(*pixmap);
    scale();
}
```

2.29 Текст програми у файлі sequencegenerator.h

```
#ifndef SEQUENCEGENERATOR_H
#define SEQUENCEGENERATOR_H

#include <QString>
#include <QHash>
#include <QFile>
#include <QTextStream>
#include <QRandomGenerator>

struct NTerminalDescriptor {
    int lExpected;
    int lDispertion;
    int width;

    NTerminalDescriptor(int lExpected, int lDispertion, int width)
        : lExpected(lExpected), lDispertion(lDispertion), width(width) {

    }

    NTerminalDescriptor()
        : lExpected(1), lDispertion(0), width(1) {

    }
};

class SequenceGenerator {
public:
    bool addNTerminal(QChar name, const QString &value = "", const NTerminalDescriptor
&descriptor = NTerminalDescriptor());
    void updateValue(QChar name, const QString &rule);
    void updateDescriptor(QChar name, const NTerminalDescriptor &descriptor);

    void deleteTerminal(QChar name);
    void setAxiom(QChar axiom);

    void save(const QString &filename);
    void load(const QString &filename);

    QString getValue(QChar name) const;
    NTerminalDescriptor getDescriptor(QChar name) const;
    QString getSequence(int n) const;

    QHash<QChar, NTerminalDescriptor> &getDescriptors();
    QChar getAxiom() const;

private:
    QHash<QChar, QString> rules;
    QHash<QChar, NTerminalDescriptor> descriptors;
    QChar axiom = '\0';
};
```

```
#endif // SEQUENCEGENERATOR_H
```

2.30 Текст програми у файлі sequencegenerator.cpp

```
#include "sequencegenerator.h"

bool SequenceGenerator::addNTerminal(QChar name, const QString &rule, const
NTerminalDescriptor &descriptor) {
    if (rules.contains(name)) {
        return false;
    }
    rules.insert(name, rule.length() == 0 ? name : rule);
    descriptors.insert(name, descriptor);
    return true;
}

void SequenceGenerator::updateValue(QChar name, const QString &rule) {
   assert(rules.contains(name));
   rules[name] = rule;
}

void SequenceGenerator::updateDescriptor(QChar name, const NTerminalDescriptor &descriptor) {
    assert(descriptors.contains(name));
    descriptors[name] = descriptor;
}

void SequenceGenerator::deleteTerminal(QChar name) {
    assert(rules.contains(name));
    rules.remove(name);
    descriptors.remove(name);
}

void SequenceGenerator::setAxiom(QChar newAxiom) {
    axiom = newAxiom;
}

void SequenceGenerator::save(const QString &filename) {
    QFile file(filename);
    if (file.open(QIODevice::WriteOnly)) {
        QTextStream stream(&file);
        stream << axiom << endl;
        for (QHash<QChar, QString>::const_iterator it = rules.constBegin();
            it != rules.constEnd(); ++it) {
           NTerminalDescriptor descr = descriptors.find(it.key()).value();
           stream << it.key() << " " << it.value() << " " << descr.width << " " <<
descr.lExpected << " " << descr.lDispertion << endl;
        }
    }
}

void SequenceGenerator::load(const QString &filename) {
    QFile file(filename);
    if (file.open(QIODevice::ReadOnly | QIODevice::Text)) {
        QTextStream stream(&file);
        descriptors.clear();
        rules.clear();
        axiom = stream.readLine()[0];
        while(!stream.atEnd()) {
            QString line = stream.readLine();
            QStringList fields = line.split(" ");
            rules.insert(fields[0][0], fields[1]);
            descriptors.insert(fields[0][0], NTerminalDescriptor(fields[3].toInt(),
fields[4].toInt(), fields[2].toInt()));
        }
    }
}

QString SequenceGenerator::getValue(QChar name) const {
    assert(rules.contains(name));
    return rules.value(name);
}
```

```
NTerminalDescriptor SequenceGenerator::getDescriptor(QChar name) const {
    assert(descriptors.contains(name));
    return descriptors.value(name);
}

QString SequenceGenerator::getSequence(int n) const {
    assert(n > 0);

    QString result = axiom;

    for (int i = 1; i < n; i++) {
        QString tmp;
        for (int j = 0; j < result.length(); j++) {
            tmp += rules.value(result[j], QString(result[j]));
        }
        result = tmp;
    }

    return result;
}

QHash<QChar, NTerminalDescriptor> &SequenceGenerator::getDescriptors() {
    return descriptors;
}

QChar SequenceGenerator::getAxiom() const
{
    return axiom;
}
```

2.31 Текст програми у файлі spot.h

```
#ifndef SPOT_H
#define SPOT_H

#include <QPoint>
#include <QSet>
#include <QtMath>

struct CircleParms {
    int radius;

 //   CircleParms(int radius) : radius(radius) {}
};

struct EllipseParms {
    int bigRadius;
    int smallRadius;
    double radAngle;

//    EllipseParms(int R, int r, double radAngle) :
//        bigRadius(R), smallRadius(r), radAngle(radAngle) {}
};

class Spot {
public:
    Spot(QSet<QPoint> points);

    QPoint getCenter() const;
    EllipseParms getEllipseParms() const;
    CircleParms getCircleParms() const;
    int getSize() const;
private:
    QPoint center;

    EllipseParms ellipseParms;
    CircleParms circleParms;


    QPoint left;
    QPoint right;
    QPoint top;
```

```
    QPoint bottom;

    int size;

    void calculateEllipseParms();
    void calculateCircleParms();
};

#endif // SPOT_H
```

2.32 Текст програми у файлі spot.cpp

```
#include "spot.h"

Spot::Spot(QSet<QPoint> points)
{
    size = 0;
    unsigned long long int sumX = 0;
    unsigned long long int sumY = 0;

    foreach(QPoint p, points)
    {
        if (p.x() <= left.x()) {
            left = p;
        }
        if (p.x() >= right.x()) {
            right = p;
        }
        if (p.y() <= top.y()) {
            top = p;
        }
        if (p.y() >= bottom.y()) {
            bottom = p;
        }

        sumX += p.x();
        sumY += p.y();
        size++;
    }

    center = QPoint(sumX/size, sumY/size);

    calculateCircleParms();
}

QPoint Spot::getCenter() const
{
    return center;
}

EllipseParms Spot::getEllipseParms() const
{
    return ellipseParms;
}

CircleParms Spot::getCircleParms() const
{
    return circleParms;
}

int Spot::getSize() const
{
    return size;
}

void Spot::calculateEllipseParms()
{
    // ellipseParms = value;
}

void Spot::calculateCircleParms()
{
```

```
    circleParms.radius = static_cast<int>(sqrt(size/M_PI));
}
```

2.33 Текст програми у файлі actions_runner.h

```
#ifndef ACTIONS_RUNNER_H
#define ACTIONS_RUNNER_H

#include <QThread>
#include <QDir>
#include "actions/actions.h"
#include "structures.h"

class ActionsRunner : public QObject
{
        Q_OBJECT
public:
    ActionsRunner(Actions &actions, const Source &source, const QString &dirToSave, bool
useDefParams = false);
public slots:
    void run();
signals:
    void finished();
    void actionNameChanged(const QString &actionName);
    void processedQuantityChanged(int processed, int total);
private:
    Actions &actions;
    const Source &source;
    QString inputPath;
    QString dirToSave;
    bool useDefParams;

    void applyAction(IAction *action);
};

#endif // ACTIONS_RUNNER_H
```

2.34 Текст програми у файлі actions_runner.cpp

```
#include "actions_runner.h"

ActionsRunner::ActionsRunner(Actions &actions, const Source &source, const QString &dirToSave,
bool useDefParams)
    : actions(actions), source(source), dirToSave(dirToSave), useDefParams(useDefParams)
{
    inputPath = source.nameGenerator->getPath();
}

void ActionsRunner::run()
{
    for (int i = 0; i < actions.getSize(); i++)
    {
        emit(actionNameChanged(actions.get(i)->getName()));

        if(!actions.get(i)->getToExecute())
            continue;

        actions.get(i)->setDirToSave(dirToSave);
        applyAction(actions.get(i));
        inputPath = dirToSave;
    }
}

void ActionsRunner::applyAction(IAction *action)
{
    if (useDefParams)
        action->reset();

    int total = source.to - source.from + 1;
    for(int i = source.from; i <= source.to; i++)
    {
        action->process(inputPath, source.nameGenerator->getFileName(i));
```

```
        emit(processedQuantityChanged(i - source.from, total));
    }
    emit(processedQuantityChanged(total, total));
}
```

2.35 Текст програми у файлі colorconvertor.h

```
#ifndef COLORCONVERTOR_H
#define COLORCONVERTOR_H

#include <math.h>

struct Lab
{
    double L;
    double a;
    double b;

    Lab(double L, double a, double b) : L(L), a(a), b(b) {}
};

struct LCH
{
    int L;
    int C;
    int H;

    LCH(int L, int C, int H) : L(L), C(C), H(H) {}
};

struct XYZ
{
    double X;
    double Y;
    double Z;

    XYZ(double X, double Y, double Z) : X(X), Y(Y), Z(Z) {}
};

struct RGB
{
    int R;
    int G;
    int B;

    RGB(int R, int G, int B) : R(R), G(G), B(B) {}
};

class ColorConvertor
{
private:
    static XYZ RGBToXYZ(const RGB &rgb);
public:
    static Lab RGBToLab(const RGB &rgb);
    static LCH LabToLCH(const Lab &lab);
    static LCH RGBToLCH(const RGB &rgb);
};

#endif // COLORCONVERTOR_H
```

2.36 Текст програми у файлі colorconvertor.cpp

```
#include "colorconvertor.h"

XYZ ColorConvertor::RGBToXYZ(const RGB &rgb)
{
    double var_R = ( rgb.R / 255.0 );
    double var_G = ( rgb.G / 255.0 );
    double var_B = ( rgb.B / 255.0 );

    if ( var_R > 0.04045 )
```

```
        var_R = pow(( ( var_R + 0.055 ) / 1.055 ), 2.4);
    else
        var_R = var_R / 12.92;


    if ( var_G > 0.04045 )
        var_G = pow(( ( var_G + 0.055 ) / 1.055 ), 2.4);
    else
        var_G = var_G / 12.92;

    if ( var_B > 0.04045 )
        var_B = pow(( ( var_B + 0.055 ) / 1.055 ), 2.4);
    else
        var_B = var_B / 12.92;

    var_R = var_R * 100;
    var_G = var_G * 100;
    var_B = var_B * 100;

   return XYZ(var_R * 0.4124 + var_G * 0.3576 + var_B * 0.1805,
              var_R * 0.2126 + var_G * 0.7152 + var_B * 0.0722,
              var_R * 0.0193 + var_G * 0.1192 + var_B * 0.9505);
}

Lab ColorConvertor::RGBToLab(const RGB &rgb)
{
    XYZ xyz = RGBToXYZ(rgb);
    const double Reference_X = 95.047;
    const double Reference_Y = 100.0;
    const double Reference_Z = 108.883;
    double var_X = xyz.X / Reference_X;
    double var_Y = xyz.Y / Reference_Y;
    double var_Z = xyz.Z / Reference_Z;

    if ( var_X > 0.008856 )
        var_X = pow(var_X, ( 1 / 3.0 ));
    else
        var_X = ( 7.787 * var_X ) + (16 / 116.0);

    if ( var_Y > 0.008856 )
        var_Y = pow(var_Y, ( 1 / 3.0 ));
    else
        var_Y = ( 7.787 * var_Y ) + (16 / 116.0);

    if ( var_Z > 0.008856 )
        var_Z = pow(var_Z, ( 1 / 3.0 ));
    else
        var_Z = ( 7.787 * var_Z ) + (16 / 116.0);

    return Lab(( 116 * var_Y ) - 16, 500 * ( var_X - var_Y ), 200 * ( var_Y - var_Z ));
}

LCH ColorConvertor::LabToLCH(const Lab &lab)
{
    double var_H = (abs(lab.a) < pow(10, -9)) ? (lab.b > 0) ? 90 : -90
                                                            : atan(lab.b/lab.a);

    if ( var_H > 0 )
        var_H = ( var_H / M_PI ) * 180;
    else
        var_H = 360 - ( abs( var_H ) / M_PI ) * 180;

    return LCH(int(lab.L), int(sqrt((lab.a * lab.a) + lab.b*lab.b)), int(var_H));
}

LCH ColorConvertor::RGBToLCH(const RGB &rgb)
{
    return LabToLCH(RGBToLab(rgb));
}
```

2.37 Текст програми у файлі decisive_function.h

```
#ifndef DECISIVEFUNCTION_H
#define DECISIVEFUNCTION_H

#include <QVector>
#include <QtMath>
#include <QColor>
#include <QSettings>
#include <QObject>

#include "colorconvertor.h"
#include "structures.h"

enum class DecisiveFunctionType
{
    Linear,
    Quadratic
};

class DecisiveFunction : public QObject
{
    Q_OBJECT
public:
    DecisiveFunction(const ParmsLab &lab, const ParmsLCH &lch, const DecisiveFunctionType
&type);

    bool isObject(const QColor &color);

    DecisiveFunctionType getType() const;
    void setType(const DecisiveFunctionType &value);

    void setToUse(bool value);
    bool getToUse() const;

    ParmsLCH getParmsLCH();
    ParmsLab getParmsLab();
public slots:
    void updateCriterias(const ParmsLab &lab, const ParmsLCH &lch);
private:
    struct Criteria
    {
        bool isCriteria;
        double l;
        double r;
        Criteria(bool _isC, double _l, double _r)
        {
            l = _l;
            r = _r;
            isCriteria = _isC;
        }
    };

    QVector<double> coefficients;
    QVector<Criteria> criterias;
    DecisiveFunctionType type;
    bool toUse;
    const QString parmGroupName;

    void recalculateCoeffients();
    double fx(double x, double min, double max) const;
};

#endif // DECISIVEFUNCTION_H
```

2.38 Текст програми у файлі decisive_function.cpp

```
#include "decisive_function.h"

DecisiveFunction::DecisiveFunction(const ParmsLab &lab, const ParmsLCH &lch, const
DecisiveFunctionType &type) :
    type(type)
{
    toUse = false;
```

```
    updateCriterias(lab, lch);
}

bool DecisiveFunction::isObject(const QColor &color)
{
    Lab lab = ColorConvertor::RGBToLab(RGB(color.red(), color.green(), color.blue()));
    LCH lch = ColorConvertor::LabToLCH(lab);
    QVector<double> values = {lab.L, lab.a, lab.b, static_cast<double>(lch.C),
static_cast<double>(lch.H), 1.0};
    double dotProduct = values[5]*coefficients[5];
    for (int i = 0; i < 5; i++)
    {
        dotProduct += fx(values[i], criterias[i].l, criterias[i].r) * coefficients[i];
    }

    if(dotProduct > qPow(10.0, -9.0))
    {
        return true;
    }
    return false;
}

DecisiveFunctionType DecisiveFunction::getType() const
{
    return type;
}

void DecisiveFunction::setType(const DecisiveFunctionType &value)
{
    type = value;
    recalculateCoeffients();
}

void DecisiveFunction::setToUse(bool value)
{
    toUse = value;
}

bool DecisiveFunction::getToUse() const
{
    return toUse;
}

ParmsLCH DecisiveFunction::getParmsLCH()
{
    ParmsLCH parmsLCH;
    parmsLCH.isL = criterias[0].isCriteria;
    parmsLCH.isC = criterias[3].isCriteria;
    parmsLCH.isH = criterias[4].isCriteria;
    parmsLCH.fromL = criterias[0].l;
    parmsLCH.fromC = criterias[3].l;
    parmsLCH.fromH = criterias[4].l;
    parmsLCH.toL = criterias[0].r;
    parmsLCH.toC = criterias[3].r;
    parmsLCH.toH = criterias[4].r;
    return parmsLCH;
}

ParmsLab DecisiveFunction::getParmsLab()
{
    ParmsLab parmsLab;
    parmsLab.isL = criterias[0].isCriteria;
    parmsLab.isA = criterias[1].isCriteria;
    parmsLab.isB = criterias[2].isCriteria;
    parmsLab.fromL = criterias[0].l;
    parmsLab.fromA = criterias[1].l;
    parmsLab.fromB = criterias[2].l;
    parmsLab.toL = criterias[0].r;
    parmsLab.toA = criterias[1].r;
    parmsLab.toB = criterias[2].r;
    return parmsLab;
```

```
}

void DecisiveFunction::updateCriterias(const ParmsLab &lab, const ParmsLCH &lch)
{
    criterias.clear();

    criterias = {
        Criteria(lab.isL, lab.fromL, lab.toL),
        Criteria(lab.isA, lab.fromA, lab.toA),
        Criteria(lab.isB, lab.fromB, lab.toB),
        Criteria(lch.isC, lch.fromC, lch.toC),
        Criteria(lch.isH, lch.fromH, lch.toH),
    };

    recalculateCoeffients();
}

void DecisiveFunction::recalculateCoeffients()
{
    coefficients.clear();
    coefficients.resize(6);
    switch (type)
    {
    case DecisiveFunctionType::Linear:
        coefficients[5] = 1.0;
        for(int i = 0; i < 5; i++)
        {
            coefficients[i] = (criterias[i].isCriteria) ? 1.0 : 0.0;
            coefficients[5] -= coefficients[i];
        }
        break;
    case DecisiveFunctionType::Quadratic:
        for (int i = 0; i < 5; i++)
        {
            coefficients[i] = (criterias[i].isCriteria) ?
                        -1.0/(((criterias[i].r - criterias[i].l) / 2.0)*((criterias[i].r -
criterias[i].l) / 2.0)) : 0.0;
        }
        coefficients[5] = 1.0;
    }
}

double DecisiveFunction::fx(double x, double min, double max) const
{
    switch (type)
    {
    case DecisiveFunctionType::Linear:
        if (x >= min && x <= max)
            return 1.0;

        return 0.0;
    case DecisiveFunctionType::Quadratic:
        x -= min + (max - min) / 2.0;
        return x*x;
    }

    return 0.0;
}
```

2.39 Текст програми у файлі fixed_name_generator.h

```
#ifndef FIXED_NAME_GENERATOR_H
#define FIXED_NAME_GENERATOR_H

#include "i_name_generator.h"

class FixedNameGenerator : public INameGenerator
{
public:
    FixedNameGenerator(int digits, const QChar value, const QString &prefix, const QString
&resolution, const QString &path);
```

```
   QString getFileName(int number);
   QString getPath();
private:
    int digits;
    QChar value;
    QString prefix;
    QString resolution;
    QString path;
};

#endif // FIXED_NAME_GENERATOR_H
```

2.40 Текст програми у файлі fixed_name_generator.cpp

```
#include "fixed_name_generator.h"

FixedNameGenerator::FixedNameGenerator(int digits, const QChar value, const  QString &prefix,
                                       const QString &resolution, const QString &path)
    : digits(digits), value(value), prefix(prefix), resolution(resolution), path(path) {}

QString FixedNameGenerator::getFileName(int number)
{
    QString strNumber = QString::number(number);
    int n = digits - strNumber.length();
    QString result = prefix;
    for (int i = 0; i < n; i++)
    {
        result += value;
    }
    result += strNumber + resolution;
    return result;
}

QString FixedNameGenerator::getPath()
{
    return path;
}
```

2.41 Текст програми у файлі form_validator.h

```
#ifndef FORM_VALIDATOR_H
#define FORM_VALIDATOR_H

#include <QString>

class FormValidator
{
public:
    FormValidator();

    static void checkField(QString text, QString name, QString &errors);
    static void checkInput(QString text, QString name, QString &errors);
    static void checkIncrease(int less, int bigger, QString nameLess, QString nameBigger,
QString &errors);
    static void checkFieldMinLenght(QString text, QString name, QString &errors, unsigned
short min);
    static void checkFieldMaxLenght(QString text, QString name, QString &errors, unsigned
short max);

    static void checkForNoPresent(QString text, QString name, QString &errors, unsigned short
max);
};

#endif // FORM_VALIDATOR_H
```

2.42 Текст програми у файлі form_validator.cpp

```
#include "form_validator.h"

FormValidator::FormValidator()
{
```

```
}

void FormValidator::checkField(QString text, QString name, QString &errors)
{
    if (text.length() == 0)
    {
         errors += QStringLiteral("Поле %1 не может быть пустым. \n").arg(name);
    }
}

void FormValidator::checkInput(QString text, QString name, QString &errors)
{
    if (text.length() == 0)
    {
         errors += QStringLiteral("Укажите, пожалуйста, %1. \n").arg(name);
    }
}

void FormValidator::checkIncrease(int less, int bigger, QString nameLess, QString nameBigger,
QString &errors)
{
    if (less > bigger)
    {
        nameLess.replace(0, 1, nameLess.at(0).toUpper());
        errors += QStringLiteral("%1 должен быть меньше, чем %2. \
n").arg(nameLess).arg(nameBigger);
    }
}

void FormValidator::checkFieldMinLenght(QString text, QString name, QString &errors, unsigned
short min)
{
    if (text.length() < min)
    {
        errors += QStringLiteral("Длина поля %1 не может быть меньше %2 символов. \
n").arg(name).arg(min);
    }
}

 void FormValidator::checkFieldMaxLenght(QString text, QString name, QString &errors, unsigned
short max)
{
     if (text.length() > max){
         errors += QStringLiteral("Длина поля %1 не может быть больше %2 символов. \
n").arg(name).arg(max);
     }
}
```

2.43 Текст програми у файлі i_name_generator.h

```
#ifndef I_NAME_GENERATOR_H
#define I_NAME_GENERATOR_H

#include <QString>

class INameGenerator
{
public:
    virtual ~INameGenerator() = default;
    virtual QString getFileName(int number)=0;
    virtual QString getPath()=0;
};

#endif // I_NAME_GENERATOR_H
```

2.44 Текст програми у файлі i_name_generator.cpp

```
#include "i_name_generator.h"
```

2.45 Текст програми у файлі simple_name_generator.h

```
#ifndef SIMPLE_NAME_GENERATOR_H
```

```
#define SIMPLE_NAME_GENERATOR_H

#include "i_name_generator.h"

class SimpleNameGenerator : public INameGenerator
{
public:
    SimpleNameGenerator(const QString &prefix, const QString &resolution, const QString
&path);

   QString getFileName(int number);
   QString getPath();
private:
   QString prefix;
   QString resolution;
   QString path;
};

#endif // SIMPLE_NAME_GENERATOR_H
```

2.46 Текст програми у файлі simple_name_generator.cpp

```
#include "simple_name_generator.h"

SimpleNameGenerator::SimpleNameGenerator(const QString &prefix, const QString &resolution,
const QString &path)
    : prefix(prefix), resolution(resolution), path(path) {}

QString SimpleNameGenerator::getFileName(int number)
{
    return prefix+QString::number(number)+resolution;
}

QString SimpleNameGenerator::getPath()
{
    return path;
}
```

Constructive Modeling of Lightning Activity in Thunderstorm Front

Viktor Shynkarenko
Department of Computer Information Technology
Dnipropetrovsk National University of Railway Transport
named after academician V. Lazaryan
Dnipro, Ukraine
shinkarenko_vi@ua.fm

Kostiantyn Lytvynenko
Department of Computer Information Technology
Dnipropetrovsk National University of Railway Transport
named after academician V. Lazaryan
Dnipro, Ukraine
kosta11111973@gmail.com

Robert Chyhir
Department of Computer Information Technology
Dnipropetrovsk National University of Railway Transport
named after academician V. Lazaryan
robertchigir@ukr.net

Iryna Sansiieva
Department of Computer Information Technology
Dnipropetrovsk National University of Railway Transport
named after academician V. Lazaryan
irinasansieva@gmail.com

***Abstract*—Using the tools of constructive-synthesizing modeling, constructors of fractal time series which determine the location, magnitude and rate of damping of lighting discharges are developed. Model video images of lightning in the thunderstorm front are formed according to constructors' implementation. The adequacy of the model is verified by comparison of the model video image with the same produced by NASA satellite.**

Keywords—L-system; constrictive-synthesizing modeling; fractal; lightning activity; thunderstorm front; time series

I. INTRODUCTION

The study of patterns of spatial distribution of thunderstorms is the relevant and practically important problem for solving both the essential tasks of atmospheric electricity and lightning protection of engineering constructions and thunderstorm fire risk of forest areas. One of the sources of data on the spatial distribution of thunderstorms is WWLLN (World Wide Lightning Location Network) [1].

Lightning monitoring was performed by satellites using detectors OTD (Optical Transient Detector) and LIS (Lightning Imaging Sensor). They are recording short bursts of infrared radiation, which arise from the lightning discharge and can be seen from space even in daytime under the clouds.

The main directions of modeling and studying of lightning activity are associated with the study of spreading of currents from clouds to the ground [2], impact of lightning on electrical systems [3], isolation of zones of the lightning activity in specific geographic areas [4], and their impact on breaking-out of fires [5]. Much lesser number of works deals with the problem of modeling of lightning in the thunderstorm front, which is primarily due to its complexity. Typically, such works are limited to isolation of compact zones (clusters) of lightning formation [6].

This paper refers to modeling of lightning activity in the thunderstorm front based on the generated fractal time series which determine the time, coordinates and duration of flashes, and comparison of the model video images to video images received from the satellite.

II. CONSTRUCTIVE-SYNTHESIZING MODELING OF FRACTAL TIME SERIES

The basis of constructive-synthesizing modeling is the concept of generalized constructive-synthesizing structure [7-9], or generalized constructor (GC):

$$C = \langle M, \Sigma, \Lambda \rangle, \tag{1}$$

where M is the heterogeneous replenishable carrier, Σ is the signature of relations and relevant operations, such as linking, substitution, and inference, over attributes, Λ is the set of statements of the information support of construction (ISC) including: ontology, purpose, rules, restrictions, terms of starting and completion of construction.

Peculiarities of the constructive-synthesizing modeling are as follows [7-9]: attributiveness of elements and operations, replenishable carrier, model of performer in the form of its basic algorithms, relation of operations to the algorithms of their implementation.

Ontology of generalized constructor in its informal representation is given in [7, 8]; below we provide its part required for the subsequent presentation.

Signature Σ comprises sets of operations: Ξ – linking, Θ – substitution and inference, Φ – operations over attributes. The signature also contains the relations of substitution "$\rightarrow$".

Operations of linking of constructor elements combine the individual elements into constructions or parts thereof (intermediate forms).

Under the form ${}_{w_l}l$ with the set of attributes w_l we understand:

- ${}_{w_l}l = {}_{w_0}\otimes({}_{w_1}m_1, {}_{w_2}m_2, \ldots, {}_{w_k}m_k)$ for $\forall {}_{w_i}m_i \in M$;
- ${}_{w_l}l = {}_{w_j}m_j$, if $l = {}_{w_0}\otimes(\varepsilon, \ldots, \varepsilon, {}_{w_j}m_j, \varepsilon, \ldots, \varepsilon)$;
- ${}_{w_l}l = {}_{w_0}\otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_k}l_k)$,

where ${}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_k}l_k$ – forms, ${}_{w_0}\otimes$ – any linking operation of Ξ with attribute w_o, ε – empty element.

The substitution relation is ${}_{w_i}l_i \rightarrow {}_{w_j}l_j$.

Let it be $s = \langle {}_{w_1}l_1 \rightarrow {}_{w_2}l_2, {}_{w_3}l_3 \rightarrow {}_{w_4}l_4, \ldots, {}_{w_m}l_m \rightarrow {}_{w_{m+1}}l_{m+1} \rangle$ – sequence of substitution relations or $s = \varepsilon$, and $g = \langle \oplus_1(w_{1,1}, w_{2,1}, \ldots, w_{k_1,1}), \oplus_2(w_{1,2}, w_{2,2}, \ldots, w_{k_2,2}), \ldots, \oplus_n(w_{1,n}, w_{2,n}, \ldots, w_{k_n,n}) \rangle$ – sequence of operations over attributes. Substitution rule is $\psi : \langle s, g \rangle$. Here $\oplus$ is a any operation over attributes ($\oplus \in \Phi$).

A set of substitution rules is $\Psi = \{\psi_i : \langle s_i, g_i \rangle\}$.

Suppose the specified form ${}_{w_l}l = \otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_h}l_h, \ldots, {}_{w_k}l_k)$ and relation of substitution ${}_{w_h}l_h \rightarrow {}_{w_q}l_q$ is such that ${}_{w_h}l_h \prec {}_{w_l}l$ (relation $\prec$ – contains), then the result of ${}_{w_l}{:}l^*$ trinary operation of substitution $\Rightarrow({}_{w_h}l_h, {}_{w_q}l_q, {}_{w_l}l)$ will be the form ${}_{w_l}{:}l^* = \otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_q}l_q, \ldots, {}_{w_k}l_k)$ where $\Rightarrow \in \Theta$.

Binary operation of partial inference ${}_{w_l}{:}l^* = {}_{v_p}|\!\Rightarrow(\Psi, {}_{w_l}l)$ ($|\!\Rightarrow \in \Theta$) consists in:

- choice of one of available substitution rules $\psi_r : \langle s_r, g_r \rangle$ with the relations of substitution s_r;
- performance of substitution operations on its base;
- performance of operations over attributes g_r.

Operation of full inference ($||\!\Rightarrow \in \Theta$) consists in stepwise conversion of forms starting with the initial form and ending with the construction satisfying the condition of inference completion which involves the cyclic performance of partial inference operations. It is a binary operation ${}_{\Delta,w_l}{:}l^* = ||\!\Rightarrow(\Psi, {}_{w_l}l)$ where ${}_{w_l}l \in M$.

The resulting constructions of full inference operations belong to $\Omega(C)$.

With a view to forming the constructions, a number of clarifying transformations are carried out:

- specialization determines the subject area: semantic nature of the carrier, finite set of operations and their semantics, attributes of operations, order of their performance and limitations of substitution rules;
- interpretation binds of signature operations with their execution algorithms, thus connecting the information model of means of constructions' formation and performer model, which generate the constructive system;
- concratization of the constructor expands axiomatics with a set of substitution rules, assigning of specific sets of nonterminal and terminal characters with their attributes and, where appropriate, the attribute values;
- implementation, which essence is formation of a set of constructions using carrier elements.

Specialization of generalized constructor on the basis of constructive-synthesizing approach and L-systems [10] can be considered as

$$C = \langle M, \Sigma, \Lambda \rangle \ {}_S\!\mapsto\ C_L = \langle M_L, \Sigma_L, \Lambda_L \rangle, \qquad (2)$$

where M_L includes the character terminals, as well as intermediate forms and multi-character constructions, Σ_L comprises a single operation, i.e. concatenation of characters and character strings (as a rule, the sign of operation between operands is omitted), Λ_L – information support includes the basics of constructive-synthesizing modeling and peculiar features of L-systems $\Lambda_L = \Lambda \bigcup \Lambda_1$.

Ontology of ISC Λ_1 includes the above designations and their semantics, notions "character", "concatenation", "character string" and other well-known concepts of multi-character processing, as well as provisions given below.

Partial inference operation $|\!\Rightarrow(\Psi, {}_{W_l}l)$ is clarified: all permitted operations of substitution of Ψ applicable to terminals of the form ${}_{w_l}l$ are performed, with looking through from left to right, except the recursion.

Initial conditions are given as a character string (axiom).

A set of non-terminal is empty.

The constructive system allows to generate the certain set of constructions (possibly, one) or to perform the check of attribution of specified construction to the above set.

In some cases, it is necessary to form two or more distinct sets of constructions similarly, where the sets of constructions being formed are different, and the processes of their formation have little variability.

In such cases it is advisable to apply parametric constructors. Suppose the family of constructions is a set of constructions characterized by the limited number of provisions

of ISC. When determining the family we specify in parentheses the constructor parameters (variable of ISC elements within the family are listed).

Concretization of C_L to the level of the family of parametric multi-character constructors gives

$$C_L = \langle M_L, \Sigma_L, \Lambda_L \rangle_K \mapsto C_{MS}(\mathrm{B},\mathrm{P},\mathrm{n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle \quad (3)$$

where B – initial character string (axiom), P – set of substitution rules, n – minimum number of terminals f in output string, $M_{MS} \supset \{f, x, \mathrm{y}, p, m, d, u, +, -, /, \backslash\}$, $\Sigma_{MS} = \Xi_{MS} = \{\circ\}$, $\circ$ – concatenation operation, $\Lambda_{MS} = \Lambda_L \bigcup \Lambda_2$. Ontological component Λ_2 includes the above terms and their semantics, as well as provisions below:

- purpose of construction – formation of string of fractal structure;
- substitution rules are set by the parameter P;
- limitations – there are no operations over attributes;
- initial conditions – the axiom is specified by B;
- termination condition – number of terminals f in output string $\geq \mathrm{n}$.

As a result of interpretation, we form the constructive system as a set of two models: constructor and internal performer

$$\langle C_{MS}(\mathrm{B},\mathrm{P},\mathrm{n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle, C_A = \langle M_A, \Sigma_A, \Lambda_A \rangle \rangle_I \mapsto C_{A,MS}(\mathrm{B},\mathrm{P},\mathrm{n}) = \langle M_{A,MS}, \Sigma_{A,MS}, \Lambda_{A,MS} \rangle, \quad (4)$$

where C_A – model of the performer in the constructor form capable of executing the basic and constructed algorithms; M_A – a set of basic and constructed algorithms; $\Sigma_A = \{;,:\}$ includes operations of sequential and conditional algorithms execution; ISC Λ_A is given in [9]; $M_{A,MS} = \langle M_{MS}, M_A \rangle$, $\Sigma_{A,MS} = \langle \Sigma_{MS}, \Sigma_A \rangle$, $\Lambda_{A,MS} = \Lambda_{MS} \bigcup \Lambda_A \bigcup \{(A_1^0 \mid_{A_i, A_j}^{A_i \cdot A_j} \hookleftarrow \cdot),$

$(A_2^0 \mid_{Z_1, Z_2, A_i}^{:A_i} \hookleftarrow :), (A_3^0 \mid_{l_i, l_j}^{l_i \circ l_j} \hookleftarrow \circ); (A_4 \mid_{l_h, l_q, l_i}^{l_j} \hookleftarrow \Rightarrow); (A_5 \mid_{l_i, \Psi}^{l_j} \hookleftarrow \mid\Rightarrow);$

$(A_6 \mid_{l_i, \Psi}^{l_j} \hookleftarrow \|\Rightarrow)\}$.

The family of parametric constructors-converters from the character string to time series

$$C_{TS}(\Omega_i(\mathrm{C}_{MS}), M_x, dM_x, \mathrm{D}_x, d\mathrm{D}_x\mathrm{m}) = \langle M_{TS}, \Sigma_{TS}, \Lambda_{TS} \rangle, \quad (5)$$

where $\Omega_i(\mathrm{C}_{A,MS})$ – strings obtained as a result of implementation of the constructor $\mathrm{C}_{A,MS}$; M_x – initial value of mathematical expectation of the time series value, $\mathrm{d}M_x$ – its increment (%), D_x – initial value of dispersion of the time series, dD_x – its increment (%), m – number of time series points, M_{TS} includes a set of terminals $T = \{f, \nu, x, \mathrm{y}, p, m,), d, u, +, -, /, \backslash\}$ nonterminals $N = \{A\}$, $\Sigma_{TS} = \Xi_{TS} \bigcup \Phi_{TS}$, $\Xi_{TS} = \{\circ, f\}$, $\Phi_{TS} = \{\wedge, +, -, \times, :, /, \backslash\}$, $\Lambda_{TS} = \Lambda_L \bigcup \Lambda_3$.

Let's introduce the operations over attributes:

- addition, subtraction, multiplication and division, accordingly, $+(c,a,b)$, $-(c,a,b)$, $\times(c,a,b)$ and $:(c,a,b)$ with operands a, b and result c;
- generation of the random normally distributed number $\wedge(c,a,b)$ with the mathematical expectation a and dispersion b.

ISC Λ_3 includes the above terms, definitions and their semantics, as well as provisions below:

- ontology complemented by known concepts allowing to operate with the time series and the real numbers;
- purpose of construction is formation of the time series $\nu(t)$;
- rules of substitution:

$$\{\langle\langle A \to fA,\ A \to \nu A\rangle, \langle \wedge(\nu, \mathrm{t}M_x, tD_x), +(t,t,dt)\rangle\rangle,$$
$$\langle\langle A \to +A\rangle, \langle q_1, +(\mathrm{t}M_x, \mathrm{t}M_x, qM)\rangle\rangle,$$
$$\langle\langle A \to -A\rangle, \langle q_1, -(\mathrm{t}M_x, \mathrm{t}M_x, qM)\rangle\rangle,$$
$$\langle\langle A \to /A\rangle, \langle q_2, +(\mathrm{t}D_x, \mathrm{t}D_x, qD)\rangle\rangle,$$
$$\langle\langle A \to \backslash A\rangle, \langle q_2, -(\mathrm{t}D_x, \mathrm{t}D_x, qD)\rangle\rangle, \quad ,$$
$$\langle\langle A \to xA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to yA\rangle, \langle\varepsilon\rangle\rangle,$$
$$\langle\langle A \to pA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to mA\rangle, \langle\varepsilon\rangle\rangle,$$
$$\langle\langle A \to dA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to uA\rangle, \langle\varepsilon\rangle\rangle,$$
$$\langle\langle A \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle\},$$

  were $q_1 = \langle \times(\mathrm{qM}, M_x, dM_x), : (\mathrm{qM}, \mathrm{qM}, 100)\rangle$,
  $q_2 = \langle \times(\mathrm{qD}, D_x, dD_x), : (\mathrm{qD}, \mathrm{qD}, 100)\rangle$.

- limitation – the rule $\langle A \to \varepsilon, \langle\varepsilon\rangle\rangle$ is observed if the other ones are not applicable;
- initial conditions – string $\Omega_i(\mathrm{C}_{MS})$; initial time $t = 0$, its step $dt = 0.04$ seconds, current value $\mathrm{t}M_x = M_x$, $\mathrm{t}D_x = D_x$;
- termination condition – observance of the empty rule.

Let's define the constructive system by interpreting C_{TS}:

$$\left\langle C_{TS}(\Omega_i(\mathrm{C}_{MS}),M_x,\mathrm{d}M_x,\mathrm{D}_x,\mathrm{m})=\left\langle M_{TS},\Sigma_{TS},\Lambda_{TS}\right\rangle,\right.$$
$$C_A=\left\langle M_A,\Sigma_A,\Lambda_A\right\rangle\rangle\ {}_I\mapsto$$
$$C_{A,TS}(\Omega_i(\mathrm{C}_{A,MS}),M_x,\mathrm{d}M_x,\mathrm{D}_x,\mathrm{d}D_x,\mathrm{m})=\left\langle M_{A,TS},\Sigma_{A,TS},\Lambda_{A,TS}\right\rangle.$$

III. Modeling of Lightning Activity in Thunderstorm Front

Based on the above constructors, the program modeling the lightning activity was developed.

Simulation results in comparison with the real thunderstorm, the video of which was obtained from the satellite are presented below. Fig.1 represents: frame of satellite video image (a) [11], the same with the removal of background (relief and cloudiness, b), with the flashes brought to the regular form (c), with all discharges (d).

A similar result was obtained as a result of modeling. The model of thunderstorm front is specified by Bezier curve. The pair of constructors $C_{A,MS}(f,\{f\to f+f\},200)\ {}_R\mapsto\Omega_1(C_{A,MS})$ and $C_{A,TS}(\Omega_1(C_{A,MS}),13,50,20,10,200)\ {}_R\mapsto\Omega_1(C_{A,TS})$ forms the time series of the lightning discharge position along the curve $u_S(t)=\Omega_1(C_{A,TS})$ (fig.1, e). The other pair $C_{A,MS}(yxyxy,\{x\to+xf,y\to-yf\},200)\ {}_R\mapsto\Omega_2(C_{A,MS})$ and $C_{A,TS}(\Omega_2(C_{A,MS}),9,5,20,10,200)\ {}_R\mapsto\Omega_2(C_{A,TS})$ – distance from the curve $u_L(t)=\Omega_2(C_{A,TS})$. The third pair $C_{A,MS}(--\backslash\backslash\backslash\backslash\backslash\backslash ppmyuumxmxxmpmmxmpxmff,$ $\{y\to-yf,x\to+xf,f\to ff,p\to+p,m\to-m,d\to\backslash\backslash d,$ $u\to/u\},200)\ {}_R\mapsto\Omega_3(C_{A,MS})$ and $C_{A,TS}(\Omega_3(C_{A,MS}),6,30,30,$ $15,200)\ {}_R\mapsto\Omega_3(C_{A,TS})$ – value of the discharge $u_R(t)=\Omega_3(C_{A,TS})$. According to the constructed three time series, a video is created, all discharges of which are shown in the (fig.1, f).

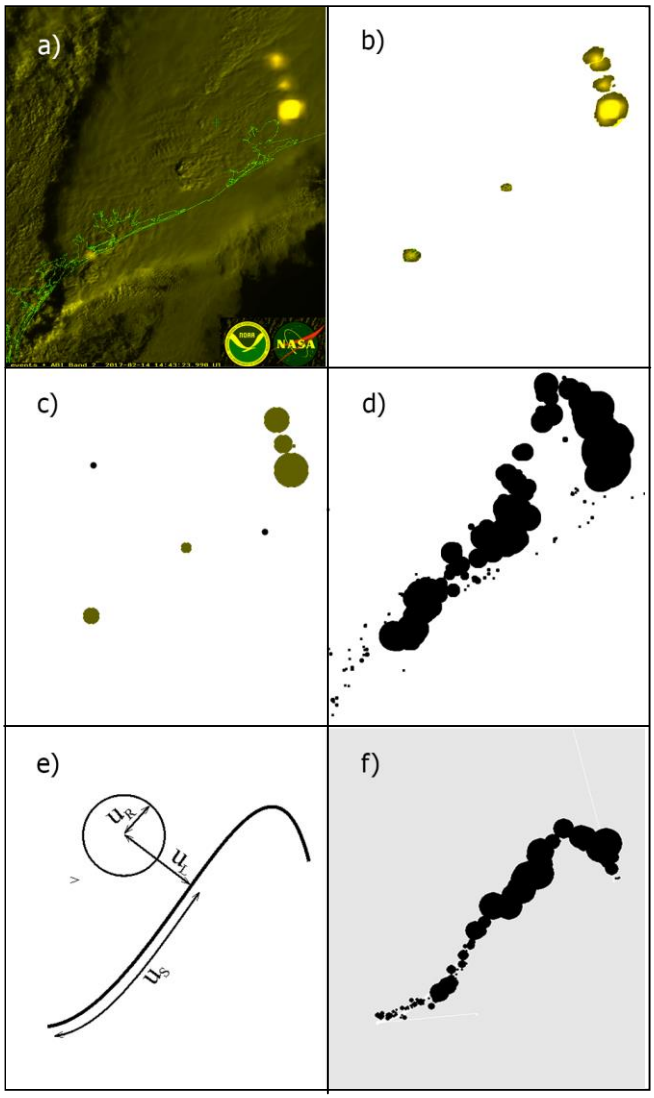

Fig. 1. Satellite and model images of thunderstorm front with lighning discharge

Conclusions

Usage of modeling in the formation of lightning discharges based on the constructive-synthesizing approach allows obtaining the realistic description of the thunderstorm front lightning activity. This approach can be the basis for solving the dynamic problem on lightning protection of engineering constructions and civil objects, and development of strategy of aircraft behavior in order to mitigate the risks of lightning strokes in the conditions of movement in the thunderstorm front.

References

[1] C.J.Rodger, S.Werner, J.B.Brundell, E.H. Lay et al. “Detection efficiency of the VLF World-Wide Lightning Location Network (WWLLN): initial case study”. Ann. Geophys, 2006, V. 24, pp. 3197-3214.

[2] T. Kraaij, Cowling R. M. and B. W. van Wilgen “Lightning and fire weather in eastern coastal fynbos shrublands: seasonality and long-term trends”. International Journal of Wildland Fire. 2013, 22, pp. 288–295.

[3] S.Pack and Piantini A. “Lightning research and lightning protection technology”. Electric Power Systems Research, vol. 113, 2014, pp. 1-2.

[4] E. Galanaki, V. Kotroni, K. Lagouvardos and A. Argiriou “A ten-year analysis of cloud-to-ground lightning activity over the Eastern Mediterranean region”. Atmospheric Research, vol. 166, 2015, pp. 213-222.

[5] Marty Ahrens “Lightning fires and lightning strikes”. National Fire Protection Association, Quincy, MA, 2013, 31p.

[6] B.R. Fuchs, E.C. Bruning, S.A. Rutledge, L.D. Carey, P.R. Krehbiel, and W. Rison “Climatological analyses of LMA data with an open‐source lightning flashclustering algorithm”. Journal of Geophysical Research: Atmospheres, vol. 121(14), 2016., pp. 8625-8648.

[7] V.I. Shynkarenko and V.M. Ilman “Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part I. Generalized Formal Constructive-Synthesizing Structure”. Cybernetics and Systems Analysis, vol. 50(5), 2014, pp. 665 – 662. doi: 10.1007/s10559-014-9655-z

[8] V.I. Shynkarenko and V.M. Ilman “Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part II. Refining Transformations”. Cybernetics and Systems Analysis, vol. 50(6), 2014, pp. 829 – 841. doi: 10.1007/s10559-014-9674-9,

[9] V.I. Shynkarenko, V.M. Ilman and V.V. Skalozub “Structural models of algorithms in problems of applied programming. I. Formal algorithmic structures”. Cybernetics and Systems Analysis, vol. 45(3), 2009, pp 329-339. doi: org/10.1007/s10559-009-9118-0

[10] A. Lindenmayer “Mathematical models for cellular interaction in development. Parts I and II.”. Journal of Theoretical Biology, vol. 18, 1968, pp. 280 - 315.

[11] First Images from GOES-16 Lightning Mapper [Online]. Available: https://www.americaspace.com/2017/03/07/goes-16-satellite-returns-first-lightning-mapping-images-like-never-seen-before/

Lightning Recognition on the Filtered Videos in the Dynamic Clouds

Viktor Shynkarenko
Department of Computer Information Technology
Dnipro National University of Railway Transport
named after academician V. Lazaryan
Dnipro, Ukraine
shinkarenko_vi@ua.fm

Iryna Nikitina
Department of Computer Information Technology
Dnipro National University of Railway Transport
named after academician V. Lazaryan
irinasansieva@gmail.com

***Abstract*— Color models' opportunities were explored for lightning flashes extraction in significant dynamic cloudiness. It was shown that the greatest efficiently in recognition is provided by combining Lab and LCH model-based features. The ranges of color channels are defined for lightning aureoles. Linear and quadratic filters were developed for aureoles detection at the series frames of the video from meteorological satellites. An analysis of their effectiveness was done. The lightning detection filter is based on the processing a current frame of the video and the filtered one containing lightning aureoles. The method and software for lightning extraction were developed based on the image filtering.**

Keywords*— *iltering of the image; color model; color image; video from satellite; lightning detection; image recognition

I. INTRODUCTION

Meteorological satellites are widely used in order to investigate and forecast the behavior of severe weather. Currently, to automate the process of the shots series analysis new algorithms are developed [1]. Much of the research related to processing data from weather satellites is devoted to detection [2, 3] or classification [4, 5] of the clouds' types. These researches are based on analysis of shape, color and cloud's motion using different methods of pattern recognition.

In [6, 7] we presented the constructive-synthesizing models of lightning activity. They are based on the video produced by NASA satellites where, firstly, the cloudiness is static and, secondly, the shape of the lightning is close to the shape of circle/ellipse. Both of these factors were taken into account in the algorithms for lightning detection and their modeling.

In this study the series of video from the satellites are investigated which demonstrate the highly dynamic cloudiness. This fact greatly complicates the lightning detection. Filters applied during the shooting allow monitoring the lightning shape in the horizontal plane with greater accuracy. We are developing lightning models by combining both horizontal and vertical directions. These series contain useful information for their modeling and validation (Fig. 1). At the figure video images [8] are shown through four frames. We can see directly the cloud's dynamic and lightning flashes' spreading.

In order to detect the lightning flashes in the dynamically changing cloudiness the classification of the changed pixels could be applied [9], but not always pixels that are lightning part change their color (the bright flashes central part as a rule is almost the same during several frames). The other approach uses particular properties of its representation by current satellite [10, 11], where the luminous strip is detected as lightning if it has sixteen pixels in its width. Despite this method cannot be applied to detect lightning at the frames with high resolution, the approach using lightning particular properties at the frames from the current satellite can be applied.

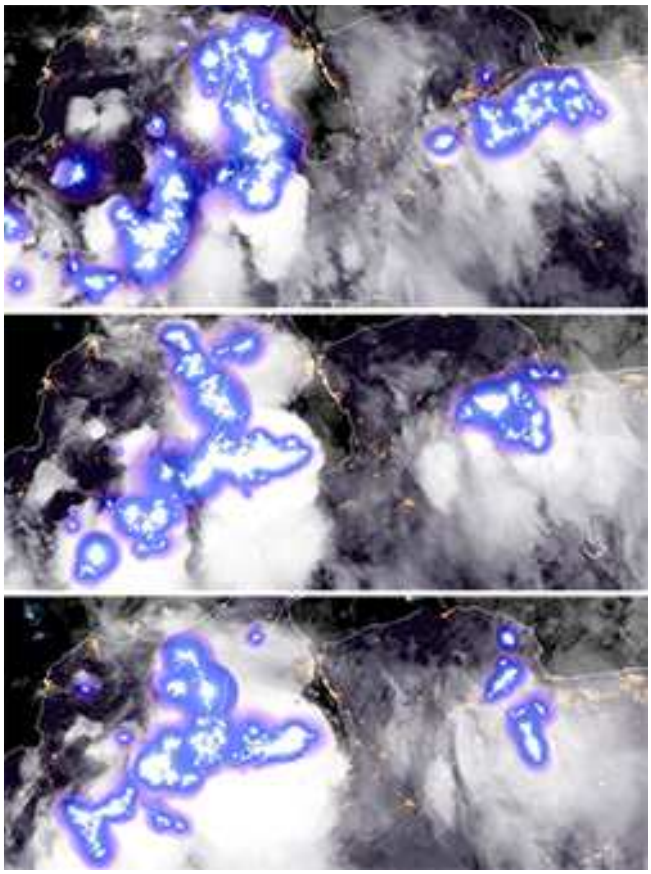

Fig. 1. Satellite images of thunderstorm front with lightning discharge

This article is devoted to searching for the particular properties at the satellite images and their future usage for lightning detection.

II. SEPARATION ABILITY OF COLOR MODELS IN LIGHTNING FLASHES EXTRACTION

At the video images filtered during the shooting from satellites [8, 12] lightning are demonstrated as bright spots with colorful contour (Fig. 1).

The analysis of lightning color from the video didn't give the positive result even after images' preprocessing such as removing color channel, edge detection using the difference of Gaussians and their varied combination was done. But it was found that the main particular property of the given videos is the clear defined aureole around the lightning. That's why the lightning contour was analyzed instead of them.

The color characteristics of the object for identification can be considered in the pattern recognition problem, while the color model matters. Studies are being conducted to search the most effective model in different fields of science, for example: skin lesion segmentation is based on the RGB and XYZ models [13], search of particular properties for damage regions in retinopathy was done in HLS, LCH and Lab models [14], Lab and HSV color models were the most informative in detecting roads segmentation for satellite aerial images [15]. We defined the most perspective color models to solve the task, they are RGB, HSV, HSL, Lab и LCH.

Lab and LCH color models were found as the most informative. In [16] to detect the color difference scheme for applying color models was proposed as at Fig. 2, we modified it (Fig. 3).

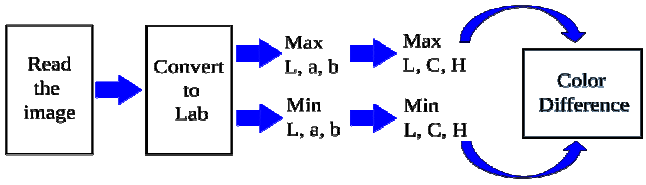

Fig. 2. The process diagram to color difference

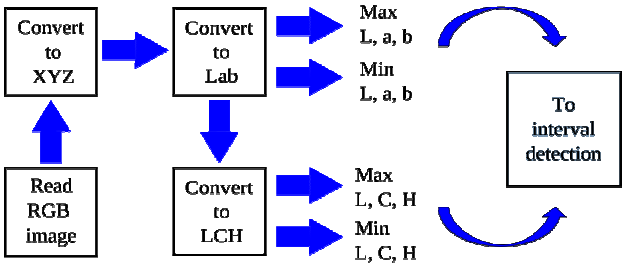

Fig. 3. The process diagram to interval detection

Five video images form [8] and [12] were selected for analysis. It was established that the lightning itself is not noticeable, while the color of the contour stands out at the image and is suitable to identify lightning. The research results showed that to identify the contour it should be considered in Lab and LCH color models as they are the best to represent the contour particular properties since low values of the exponent b are present only at the contours, and high values are present only in a small amount outside it (correlation of pixels amount to b value for contour and entire image is shown with blue and green respectively at Fig. 4), and the exponent H is a quite narrow part of the scale.

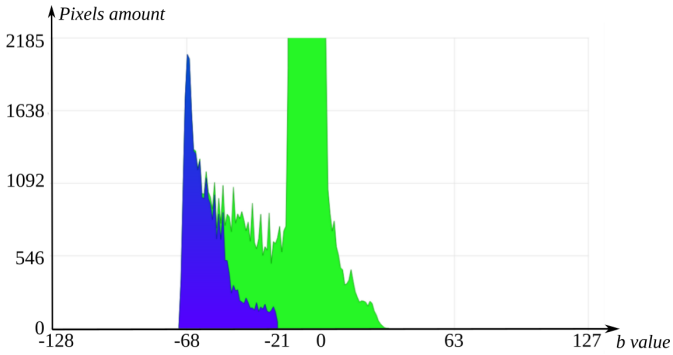

Fig. 4. Pictograph of the color frequency in the channel ***b*** (Lab) frame 1

The values range in these models are presented in Table I. L parameter has the same sense for both models.

The average value of the range was calculated for manually extracted lightning aureoles at the selected video images (row "Average" in the table):

$$c_l(k) = \sum_{i=1}^{5} c_{il}(k)/5;\ c_u(k) = \sum_{i=1}^{5} c_{iu}(k)/5, \qquad (1)$$

where $(c_{il}(k), c_{iu}(k))$ – are the low and the upper bound of the color for i-th frame in the k channel.

The range for extracting lightning aureoles is somewhat expanded (row "Interval" in the table):

$$\begin{aligned} l_k &= c_l + 0{,}005 \min_i (c_l(k) - c_{il}(k)), \\ u_k &= c_u + 0{,}005 \max_i (c_{il}(k) - c_l(k)). \end{aligned} \qquad (2)$$

where 0,005 – is expansion coefficient obtained by experimentation.

Table I also shows the part (%) of the scale-covered by the calculated range (2).

TABLE I. THE LIGHTNING AUREOLES COLOR RANGES IN LAB И LCH MODELS

	L	a	b	C	H
Frame 1 [8]	(24, 85)	(1, 42)	(-68, -21)	(19, 79)	(274, 302)
Frame 2 [8]	(22, 89)	(-2, 40)	(-69, -17)	(14, 80)	(270, 299)
Frame 3 [8]	(24, 90)	(0, 41)	(-67, -14)	(18, 78)	(272, 301)
Frame 4 [12]	(21, 92)	(0, 43)	(-61, -13)	(10, 79)	(270, 302)
Frame 5 [12]	(25, 87)	(2, 39)	(-63, -15)	(8, 82)	(273, 301)
Average	(23, 89)	(0, 41)	(-66, -16)	(14, 80)	(272, 301)
Interval	**(22, 91)**	**(-3, 44)**	**(-70, -12)**	**(11, 81)**	**(268, 303)**
% scale	**69**	**18**	**23**	**70**	**10**

Image filtering will be performed using both models, since many researches show that the accuracy of object recognition increases when combining two [17], three [18] and even seven [19] models.

III. LIGHTNING FLASHES EXTRACTION

The next filters were developed in order to detect the lightning aureoles.

The linear one:

$$y_{ij} = \begin{cases} x_{ij} \ if \ \ d_{ij} > 0 \\ 0 \ \text{otherwise}, \end{cases} \tag{3}$$

where x_{ij} – is a pixel's value with coordinates $(i,\ j)$ at the frame before processing, y_{ij} – is the respective value at the formed video image containing only lightning contours (color model for both frames is RGB), and

$$d_{ij} = \sum_{k=0}^{4} f(x_{ijk}, l_k, u_k) - 4, \tag{4}$$

$$f(x_{ijk}, l_k, u_k) = \begin{cases} 1 \ if \ \ l_k < x_{ijk} < u_k \\ 0 \ \text{otherwise}, \end{cases} \tag{5}$$

where x_{ijk} – is a pixel's value with coordinates $(i,\ j)$ using

- $k = 0$ for L color channel in Lab and LCH models;
- $k = 1$ for a color channel in Lab model,
- $k = 2$ for b color channel in Lab model;
- $k = 3$ for C color channel in LCH model;
- $k = 4$ for H color channel in LCH model.

Quadratic filter function: (3), (6):

$$d_{ij} = 1 - \sum_{k=0}^{4} \frac{(x_{ijk} - (r_k - u_k)/2)^2}{(r_k - u_k + 1)^2}. \tag{6}$$

The values presented in Table I were substituted into the filter functions and all video images [8, 12] were filtered using each of them. Fig. 5 shows the results of contour extraction for the middle frame at Fig. 1 using linear filter (3), (4) at Fig. 5 a) and quadratic one (3), (6) at Fig. 5 b).

To find out the most effective filter, contours were extracted manually from the source image and after that they were compared with automatically detected ones. The result is at Fig. 5 a) and Fig. 5 b) for linear and quadratic filters respectively. The pixels are marked with green if they represent aureole at both manually created and filtered image. The gray pixels show the same case for the image background. Lightning aureole at the manually created frame and background at the filtered one is marked with red, and the opposite case is shown with blue.

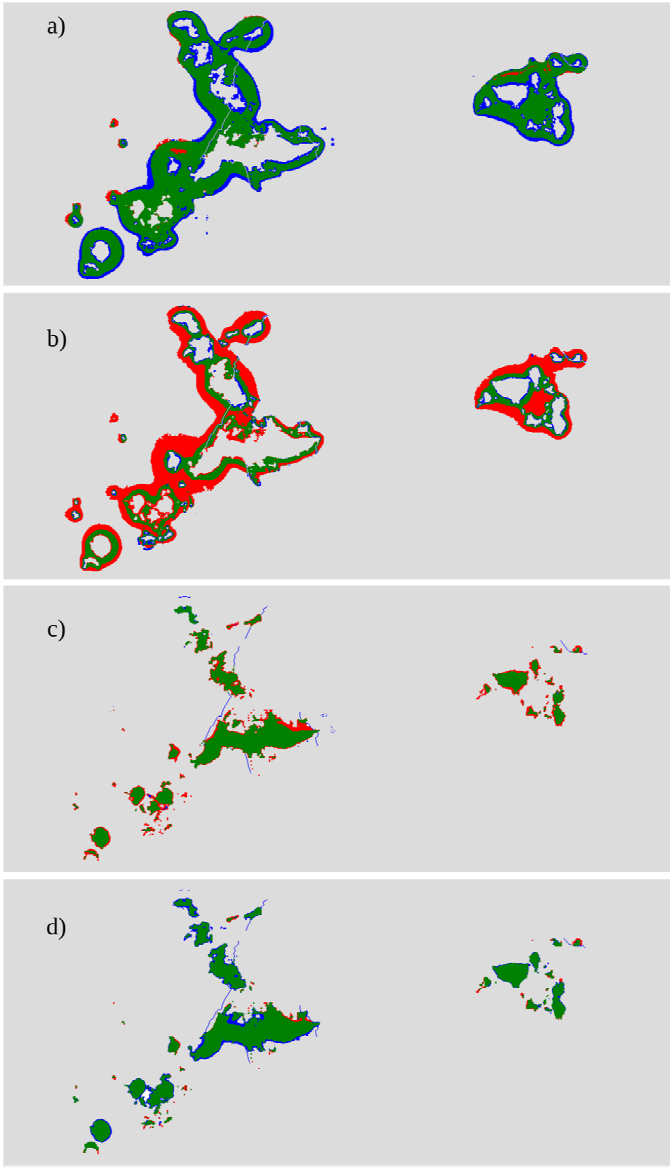

Fig. 5. Extracted lightning aureoles and the lightning themselves

As we can see the quadratic function cuts out more than the middle, which is good for small dim lightning, because the linear function does not leave a cavity in the contour, which leads to the fact that the lightning will be lost. Also, the linear filter leaves a background that cannot be attributed to the contour, which will lead to the appearance of "imaginary lightning".

Then it is required to extract the lightning. To do this the next algorithm was developed: pixels are analyzed starting from the upper left corner in left-to-right scanline order. It is considered that pixel with background color belongs to lightning if:

$$z_{ij} = \begin{cases} x_{ij} \ if \ \sum_{k=j-1}^{j=1} \ \sum_{n=i-1, n\neq i \& k\neq j}^{i=1} \delta(y_{n,k}) \geq 3 \\ x_{ij} \ if \ \sum_{k=j-1}^{j=1} \ \sum_{n=i-1, n\neq i \& k\neq j}^{i=1} (\delta(y_{n,k}) + \delta(z_{n,k})) \geq 4 \\ 0 \text{ otherwise,} \end{cases}$$

where x_{ij} is a pixel's value with coordinates (i, j) at the source frame, y_{ij} is the respective value at the filtered frame with extracted lightning aureoles, z_{ij} is the pixel's value at the frame with extracted lightning flashes, and

$$\delta(a) = \begin{cases} 1 \ if \ \ a \neq 0 \\ 0 \text{ otherwise.} \end{cases}$$

Such approach allows to process even non-closed contours as it is in the case with small lightning which loss is critical for the second video [12].

However, besides of lightning, parts from the contours outside also appear at the resulting image. To solve the problem the color properties were analyzed as for lightning, so as for noises. In this way, a pixel is marked for repaint only if it meets the criteria:

$$z_{ij} = \begin{cases} z_{ij} \ \ if \ \ x_{ij}(C) < 50 \,\&\, x_{ij}(L) < 50 \,\&\, x_{ij}(a) < 10 \\ 0 \text{ otherwise.} \end{cases}$$

New pixel's color corresponds to the pixel's color at the source image. The images that contain extracted lightning from the source frames are the result of the proceedings. Fig. 5 c) and Fig. 5 d) show the results of lightning extraction for the middle frame at Fig. 1 after the contours were detected using linear and quadratic filters respectively. The comparison with manually prepared lightning from the source image was done. The colors at Fig. 5 c) and Fig. 5 d) have the same sense as for contours comparison.

IV. CONCLUSIONS

The method and software for lightning aureoles detection and then lightning at the video frames from weather satellites with significantly dynamic cloud cover were developed. Video features are its filtering when shooting in such a way that lightning flashes are surrounded by a color contour.

The identification of lightning contours and lightning themselves are based on the images filtering using Lab and LCH color models.

REFERENCES

[1] V. Sainte, L. Landrieu, S. Giordano and N. Chehata, "Satellite Image Time Series Classification with Pixel-Set Encoders and Temporal Self-Attention". 2019. arXiv preprint arXiv:1911.07757.

[2] H. Fu, Y. Shen, J. Liu, J. Qian and J. Li, "Cloud Detection for FY Meteorology Satellite Basedon Ensemble Thresholds and Random Forests Approach". In: Remote Sensing, 11(1), 44, 2019, pp. 1-28.

[3] M. Peterson, S. Rudlosky and D. Zhang, "Thunderstorm Cloud-Type Classification from Space-Based Lightning Images". In: Monthly Weather Review, 2020, pp. 1891-1898/ doi: 10.1175/MWR-D-19-0365.1.

[4] W. Jin, F. Gong, B. Tang and S. Wang, "Cloud Types Identification for Meteorological Satellite Image Using Multiple Sparse Representation Classifiers via Decision Fusion". In: IEEE Access, vol. 7, 2019, pp. 8675-8688/ doi: 10.1109/ACCESS.2018.2890295.

[5] X. Zheng, J. Ye, Y. Chen, S. Wistar, J. Li, J. Fernandez, M. Steinberg and J. Wang, "Detecting Comma-Shaped Clouds for Severe Weather Forecasting Using Shape and Motion". In: IEEE Transactions on Geoscience and Remote Sensing, 57(6), 2019, pp. 1-14.

[6] V. Shynkarenko, K. Lytvynenko, R. Chyhir and I. Nikitina, "Modeling of Lightning Flashes in Thunderstorm Front by Constructive Production of Fractal Time Series". In: Shakhovska N., Medykovskyy M. (eds) Advances in Intelligent Systems and Computing IV, vol. 1080, Springer, 2020, pp. 173-185/ doi: 10.1007/978-3-030-33695-0_13.

[7] V. Shynkarenko, K. Lytvynenko, R. Chyhir and I. Sansiieva, "Constructive Modeling of Lightning Activity in Thunderstorm Front". In: IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), 2019, pp. 92-95/ doi:10.1109/STC-CSIT.2019.8929754.

[8] Nocturnal thunderstorms near Lake Maracaibo are known for their "Catatumbo Lightning", 2018-08-15 [Online]. Available: http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes16/20180815000000&number_of_images_to_display=100&loop_speed_ms=15.

[9] K. Radhika and S. Varadarajan, "A Neural Network Based Classification of Satellite Images for Change Detection Applications". In: Cogent Engineering, 5, 2018, pp. 1-9/ doi: 10.1080/23311916.2018.1484587.

[10] G. Chen, Y. Liu, Y. Tian and H. Tian, "Use of VIIRS DNB Satellite Images to Detect Nighttime Fishing Vessel Lights in Yellow Sea". In: Proceedings of the 3rd International Conference on Computer Science and Application Engineering, 2019, pp. 1-5/ doi: 10.1145/3331453.3361661.

[11] D. Elvidge, M. Zhizhin, K. Baugh and F. Hsu, "Automatic Boat Identification System for VIIRS Low Light Imaging Data". In: Remote Sensing, 7(3), 2015, pp. 3020–3036/ doi: 10.3390/rs70303020.

[12] The next storm to hit the U.S. West Coast is already producing lightning, 2019-01-16 [Online]. Available: http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes-16/20190116000000&number_of_images_to_display=120&loop_speed_ms=100.

[13] D. Thanh, U. Erkan, V. Prasath, V. Kumar and N. Hien, "A Skin Lesion Segmentation Method for Dermoscopic Images Based on Adaptive Thresholding with Normalization of Color Models". In: 2019 6th International Conference on Electrical and Electronics Engineering (ICEEE), 2019, pp. 116-120/ doi: 10.1109/ICEEE2019.2019.00030.

[14] J. Patil and S. Chaudhari, "Screening of Damage Regions in Retinopathy Using Segmentation-Color Space Selection". In: International Journal Multimedia and Image Processing (IJMIP), 7(1), 2017, pp. 362-365/ doi:10.20533/ijmip.2042.4647.2017.0044.

[15] L. Thanh and D. Thanh, "An Adaptive Local Thresholding Roads Segmentation Method for Satellite Aerial Images with Normalized HSV and Lab Color Models". In: V. Solanki V., Hoang M., Lu Z., Pattnaik P. (eds) Intelligent Computing in Engineering. Advances in Intelligent Systems and Computing, vol. 1125, Springer, 2020. pp. 865-872.

[16] M. Sayed, Sammani and M. Albashier, "An accurate method to calculate the color difference in a single image". In: International Conference on Robotics, Automation and Sciences (ICORAS), 2017, pp. 1-3.

[17] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat and J. Jatakia (2017), "Human skin detection using RGB, HSV and YCbCr color models". 2017. arXiv preprint arXiv:1708.02694.

[18] I. Mporas, I. Perikos and M. Paraskevas, "Color Models for Skin Lesion Classification from Dermatoscopic Images". In: Advances in Integrations of Intelligent Methods. Smart Innovation, Systems and Technologies, vol 170, Springer, pp. 85-98/ doi: 10.1007/978-981-15-1918-5_5.

[19] C. Chuang, J. Chen, H. Lin, S. Huang, and J. Wen, J. C, "Soil Moisture Estimation Using Multiple Color Spaces in Digital Image Analysis". In Geophysical Research Abstracts, vol. 21, 2019, EGU2019-11542.

Advances in Intelligent Systems and Computing 1080

Natalya Shakhovska
Mykola O. Medykovskyy *Editors*

Advances in Intelligent Systems and Computing IV

Selected Papers from the International Conference on Computer Science and Information Technologies, CSIT 2019, September 17–20, 2019, Lviv, Ukraine

Springer

Advances in Intelligent Systems and Computing

Volume 1080

Modeling of Lightning Flashes in Thunderstorm Front by Constructive Production of Fractal Time Series

Viktor Shynkarenko(✉), Kostiantyn Lytvynenko(✉), Robert Chyhir(✉), and Iryna Nikitina(✉)

Dnipro National University of Railway Transport named after academician V. Lazaryan, Dnipro, Ukraine
shinkarenko_vi@ua.fm, kostalllll973@gmail.com, robertchigir@ukr.net, irinasansieva@gmail.com

Abstract. Using the tools of structural-synthesizing modeling, a set of constructors was developed. Implementing parametric multi-character constructors allows to form fractal sequences of characters. Constructor-converter from the character string to time series creates fractal time series, which determine the location, magnitude and decay rate of lightning discharges. Model video images of lightnings in the thunderstorm front are formed in accordance with the implementation of the constructor-assembler. All constructors are developed on the basis of the generalized constructor that was previously presented and repeatedly tested. The model adequacy of the model is confirmed by comparing the video image of the model with the image, what was obtained by NASA satellite. This approach can be the basis for solving the dynamic problems on lightning protection of engineering constructions and civil objects, and development of strategy of aircraft behavior in order to mitigate the risks of lightning strokes in the conditions of movement in the thunderstorm front.

Keywords: L-system · Constrictive-synthesizing modeling · Fractal · Lightning activity · Lightning flash · Thunderstorm front · Time series

1 Introduction

Standard monitoring and forecasting of hazardous thunderstorm phenomena are carried out in all countries on the basis of a unified program and regulatory documents. Such monitoring includes:

- regular monitoring of qualitative and quantitative indicators of the atmospheric thunderstorm state;
- collection, processing and storage of observations of thunderstorm phenomena;
- creation and maintenance of observational databases.

It becomes possible to conduct modeling experiments in order to develop adequate quantitative predictive models of lightning activity of thunderstorm fronts.

The study of patterns of spatial distribution of thunderstorms is the relevant and practically important problem for solving both the essential tasks of atmospheric

N. Shakhovska and M. O. Medykovskyy (Eds.): CCSIT 2019, AISC 1080, pp. 173–185, 2020.
https://doi.org/10.1007/978-3-030-33695-0_13

electricity and lightning protection of engineering constructions and thunderstorm fire risk of forest areas. One of the sources of data on the spatial distribution of thunderstorms is WWLLN (World Wide Lightning Location Network) [1].

Lightning monitoring was performed by satellites using detectors OTD (Optical Transient Detector) and LIS (Lightning Imaging Sensor). They are recording short bursts of infrared radiation, which arise from the lightning discharge and can be seen from space even in daytime under the clouds.

The main directions of modeling and studying of lightning activity are associated with the study of spreading of currents from clouds to the ground [2], impact of lightning on electrical systems [3].

2 State of Problem

Research on regional and global lightning activity and the global electrical circuit is summarized in scientific research [4]. This area of activity has greatly expanded through observations of lightning by satellite and through increased using of the natural resonances of the Earth–ionosphere cavity.

The complex relationships between lightning and rain yields in convective storms have been studied extensively, with different relationships derived in varying geographical locations and atmospheric conditions [5].

Regional behavior of lightning activity is studied for a long time [6].

The effect of solar variability parameters and meteorological parameters on total lightning flashes and convective rain in two selected regions is studied in the article [7], isolation of zones of the lightning activity in specific geographic areas [8], and their impact on breaking-out of fires [9]. Much lesser number of works deals with the problem of modeling of lightning in the thunderstorm front, which is primarily due to its complexity. Typically, such works are limited to isolation of compact zones (clusters) of lightning formation [10].

In this time the modeling of lightning activity on the base of satellite monitoring of flash rate from convective cell can help severe weather forecasts, improve tornado forecasters and their impending threat to the public.

This paper refers to modeling of lightning activity in the thunderstorm front based on the generated fractal time series which determine the time, coordinates and duration of flashes, and comparison of the model video images to video images received from the satellite.

In the constructive approach to the formation of objects of different nature there is an opportunity to understand the constructive notion of the real technical or nature object.

3 The Main Material

3.1 Constructive Modeling Tools

Generalized Constructor. The basis of constructive-synthesizing modeling is the concept of generalized constructive-synthesizing structure [11–13], or generalized constructor (GC):

$$C = \langle M, \Sigma, \Lambda \rangle, \tag{1}$$

where M – heterogeneous replenishable carrier, Σ – signature of relations and relevant operations, such as linking, substitution, and inference, over attributes, Λ – set of statements of the information support of construction (ISC) including: ontology, purpose, rules, restrictions, terms of starting and completion of construction.

Peculiarities of the constructive-synthesizing modeling are as follows [11–13]: attributiveness of elements and operations, replenishable carrier, model of performer in the form of its basic algorithms, relation of operations to the algorithms of their implementation. Main provisions ontological support of constructive-synthesizing modeling developed in [14].

Ontology of generalized constructor in its informal representation is given in [11, 12]; below we provide its part required for the subsequent presentation.

Signature Σ comprises sets of operations: Ξ – linking, Θ – substitution and inference, Φ – operations over attributes. The signature also contains the relations of substitution "$\rightarrow$".

Operations of linking of constructor elements combine the individual elements into constructions or parts thereof (intermediate forms).

Under the form ${}_{w_l}l$ with the set of attributes w_l we understand:

- ${}_{w_l}l = {}_{w_0} \otimes ({}_{w_1}m_1, {}_{w_2}m_2, \ldots, {}_{w_k}m_k)$ for $\forall_{w_i} m_i \in M$;
- ${}_{w_l}l = {}_{w_j}m_j$, if $l = {}_{w_0} \otimes (\varepsilon, \ldots, \varepsilon, {}_{w_j}m_j, \varepsilon, \ldots, \varepsilon)$;
- ${}_{w_l}l = {}_{w_0} \otimes ({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_k}l_k)$,

where ${}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_k}l_k$ – forms, ${}_{w_0}\otimes$ – any linking operation of Ξ with attribute w_o, ε – empty element.

The substitution relation is ${}_{w_i}l_i \rightarrow {}_{w_j}l_j$.

Let it be $s = \langle {}_{w_1}l_1 \rightarrow {}_{w_2}l_2, {}_{w_3}l_3 \rightarrow {}_{w_4}l_4, \ldots, {}_{w_m}l_m \rightarrow {}_{w_{m+1}}l_{m+1} \rangle$ – sequence of substitution relations or $s = \varepsilon$, and $g = \langle \oplus_1(w_{1,1}, w_{2,1}, \ldots, w_{k_1,1}), \oplus_2(w_{1,2}, w_{2,2}, \ldots, w_{k_2,2},), \ldots, \oplus_n(w_{1,n}, w_{2,n}, \ldots, w_{k_n,n}) \rangle$ – sequence of operations over attributes. Substitution rule is $\psi : \langle s, g \rangle$. Here $\oplus$ is a any operation over attributes ($\oplus \in \Phi$).

A set of substitution rules is $\Psi = \{\psi_i : \langle s_i, g_i \rangle\}$.

Suppose the specified form ${}_{w_l}l = \otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_h}l_h, \ldots, {}_{w_k}l_k)$ and relation of substitution ${}_{w_h}l_h \rightarrow {}_{w_q}l_q$ is such that ${}_{w_h}l_h \prec {}_{w_l}l$ (relation $\prec$ – contains), then the result of ${}_{w_l^*}l^*$ trinary operation of substitution $\Rightarrow ({}_{w_h}l_h, {}_{w_q}l_q, {}_{w_l}l)$ will be the form ${}_{w_l^*}l^* = \otimes({}_{w_1}l_1, {}_{w_2}l_2, \ldots, {}_{w_q}l_q, \ldots, {}_{w_k}l_k)$ where $\Rightarrow \in \Theta$.

Binary operation of partial inference ${}_{w_l^*}l^* = {}_{v_p}|\Rightarrow(\Psi, {}_{w_l}l)$ ($|\Rightarrow \in \Theta$) consists in:

- choice of one of available substitution rules $\psi_r : \langle s_r, g_r \rangle$ with the relations of substitution s_r;
- performance of substitution operations on its base;
- performance of operations over attributes g_r.

Operation of full inference ($||\Rightarrow \in \Theta$) consists in stepwise conversion of forms starting with the initial form and ending with the construction satisfying the condition of inference completion which involves the cyclic performance of partial inference operations. It is a binary operation ${}_{\Delta,w_l^*}l^* = ||\Rightarrow(\Psi, {}_{w_l}l)$ where ${}_{w_l}l \in M$.

The resulting constructions of full inference operations belong to $\Omega(C)$.

With a view to forming the constructions, a number of clarifying transformations are carried out:

– specialization determines the subject area: semantic nature of the carrier, finite set of operations and their semantics, attributes of operations, order of their performance and limitations of substitution rules;
– interpretation binds of signature operations with their execution algorithms, thus connecting the information model of means of constructions' formation and performer model, which generate the constructive system;
– concretization of the constructor expands axiomatics with a set of substitution rules, assigning of specific sets of nonterminal and terminal characters with their attributes and, where appropriate, the attribute values;
– realization, which essence is formation of a set of constructions using carrier elements.

Specialization of generalized constructor on the basis of constructive-synthesizing approach and L-systems [15] can be considered as

$$C = \langle M, \Sigma, \Lambda \rangle_S \mapsto C_L = \langle M_L, \Sigma_L, \Lambda_L \rangle, \tag{2}$$

where M_L includes the character terminals, as well as intermediate forms and multi-character constructions, Σ_L comprises a single operation, i.e. concatenation of characters and character strings (as a rule, the sign of operation between operands is omitted), Λ_L – information support includes the basics of constructive-synthesizing modeling and peculiar features of L-systems $\Lambda_L = \Lambda \cup \Lambda_1$.

Ontology of ISC Λ_1 includes the above designations and their semantics, notions "character", "concatenation", "character string" and other well-known concepts of multi-character processing, as well as provisions given below.

Partial inference operation $|\Rightarrow(\Psi, {}_{W_l}l)$ is clarified: all permitted operations of substitution of Ψ applicable to terminals of the form ${}_{w_l}l$ are performed, with looking through from left to right, except the recursion.

Initial conditions are given as a character string (axiom).

A set of non-terminal is empty.

The constructive system allows to generate the certain set of constructions (possibly, one) or to perform the check of attribution of specified construction to the above set.

In some cases, it is necessary to form two or more distinct sets of constructions similarly, where the sets of constructions being formed are different, and the processes of their formation have little variability.

In such cases it is advisable to apply parametric constructors. Suppose the family of constructions is a set of constructions characterized by the limited number of provisions of ISC. When determining the family we specify in parentheses the constructor parameters (variable of ISC elements within the family are listed).

Parametric Multi-character Creator Constructors. Concretization of C_L to the level of the family of parametric multi-character constructors gives

$$C_L = \langle M_L, \Sigma_L, \Lambda_L \rangle_K \mapsto C_{MS}(\mathrm{B}, \mathrm{P}, \mathrm{n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle \tag{3}$$

where B – initial character string (axiom), P – set of substitution rules, n – minimum number of terminals f in output string, $M_{MS} \supset \{f, x, y, p, m, d, u, +, -, /, \backslash\}$, $\Sigma_{MS} = \Xi_{MS} = = \{\circ\}$, $\circ$ – concatenation operation, $\Lambda_{MS} = \Lambda_L \cup \Lambda_2$. Ontological component Λ_2 includes the above terms and their semantics, as well as provisions below:

- purpose of construction – formation of string of fractal structure;
- substitution rules are set by the parameter P;
- limitations – there are no operations over attributes;
- initial conditions – the axiom is specified by B;
- termination condition – number of terminals f in output string $\geq$ n.

As a result of interpretation, we form the constructive system as a set of two models: constructor and internal performer

$$\begin{aligned} &\langle C_{MS}(\mathrm{B}, \mathrm{P}, \mathrm{n}) = \langle M_{MS}, \Sigma_{MS}, \Lambda_{MS} \rangle, C_A = \langle M_A, \Sigma_A, \Lambda_A \rangle \rangle_I \mapsto \\ &C_{A,MS}(\mathrm{B}, \mathrm{P}, \mathrm{n}) = \left\langle M_{A,MS}, \Sigma_{A,MS}, \Lambda_{A,MS} \right\rangle, \end{aligned} \tag{4}$$

where C_A – model of the performer in the constructor form capable of executing the basic and constructed algorithms; M_A – a set of basic and constructed algorithms; $\Sigma_A = \{\cdot, :\}$ includes operations of sequential and conditional algorithms execution; ISC Λ_A is given in [13]; $M_{A,MS} = \langle M_{MS}, M_A \rangle$, $\Sigma_{A,MS} = \langle \Sigma_{MS}, \Sigma_A \rangle$, $\Lambda_{A,MS} = \Lambda_{MS} \cup \Lambda_A \cup$

$$\{(A_1^0 \,|_{A_i,A_j}^{A_i \cdot A_j} \hookleftarrow \cdot), (A_2^0 \,|_{Z_1,Z_2,A_i}^{:A_i} \hookleftarrow :), (A_3^0 \,|_{I_i,I_j}^{I_i \circ I_j} \hookleftarrow \circ); (A_4 \,|_{I_h,I_q,I_l}^{I_j} \hookleftarrow \Rightarrow); (A_5 \,|_{I_i,\Psi}^{I_j} \hookleftarrow |\Rightarrow);$$

$(A_6 \,|_{I_i,\Psi}^{I_j} \hookleftarrow ||\Rightarrow)\}$..

Algorithms M_A:

– performing an algorithm composition compilation $A_1^0 \Big|_{A_i,A_j}^{A_i \cdot A_j} (A|_X^Y$ – an algorithm over data from an input set X with result values from a set Y, A^0 – generating an algorithm), $A_i, A_j \in \Omega(C_{A,MS})$, $A_i \cdot A_j$ – sequential execution of the algorithm A_j after the algorithm A_i;

- conditional execution $A_2^0|_{Z_1,Z_2,A_i}^{:A_i}$, which consists in executing the algorithm A_i under condition $Z_1 \supseteq Z_2$;
- concatenations of chain of symbols $A_3^0|_{l_i,l_j}^{l_i \circ l_j}$, $l_i, l_j \in M_{MS}$;
- executing the substitution operation $\{A_4|_{l_h,l_q,l_i}^{l_j}$, $, l_i, l_j, l_h, l_q \in M_{MS}$, l_i, l_j – the current form in which the substitution operation is performed before and after it is executed, l_h, l_q – the chains in the left and right part of the substitution relation, according to which is executed;
- performing partial and complete output operations $A_5|_{l_i,\Psi}^{l_j}$, $A_6|_{l_i,\Psi}^{l_j}$, $\Psi \subset \Lambda_{MS}$ – a set of rules of substitution.

Constructor-Converter from the Character String to Time Series. The family of such constructors

$$C_{TS}(\Omega_i(C_{MS}), M_x, dM_x, \mathrm{D}_x, d\mathrm{D}_x m) = \langle M_{TS}, \Sigma_{TS}, \Lambda_{TS} \rangle \tag{5}$$

where $\Omega_i(C_{A,MS})$ – strings obtained as a result of implementation of the constructor $\mathrm{C}_{A,MS}$; M_x – initial value of mathematical expectation of the time series value, $\mathrm{d}M_x$ – its increment (%), D_x – initial value of dispersion of the time series, dD_x – its increment (%), m – number of time series points, M_{TS} includes a set of terminals $T = \{f, v, x, y, p, m,\}$, $d, u, +, -, /, \backslash\}$ nonterminals $N = \{A\}$, $\Sigma_{TS} = \Xi_{TS} \cup \Phi_{TS}$, $\Xi_{TS} = \{\circ, f\}$, $\Phi_{TS} = \{\wedge, +, -, \times, :, /, \backslash\}$, $\Lambda_{TS} = \Lambda_L \cup \Lambda_3$.

Let's introduce the operations over attributes:

- addition, subtraction, multiplication and division, accordingly, $+(c, a, b)$, $-(c, a, b)$, $\times(c, a, b)$ and $:(c, a, b)$ with operands a, b and result c;
- generation of the random normally distributed number $\wedge(c, a, b)$ with the mathematical expectation a and dispersion b.

ISC Λ_3 includes the above terms, definitions and their semantics, as well as provisions below:

- ontology complemented by known concepts allowing to operate with the time series and the real numbers;
- $''\circ''$ – relation between adjacent array elements;
- purpose of construction is formation of the time series $v(t)$;
- rules of substitution:

$\{\langle\langle A \to fA,\ B \to z_i \circ B\rangle, \langle\wedge(v, tM_x, tD_x),\ +(t, t, dt),\ +(i, i, 1)\rangle\rangle,$
$\langle\langle A \to\ +A\rangle, \langle\times(\mathrm{qM}, M_x, dM_x), :(\mathrm{qM}, \mathrm{qM}, 100),\ +(\mathrm{t}M_x, \mathrm{t}M_x, qM)\rangle\rangle,$
$\langle\langle A \to -A\rangle, \langle\times(\mathrm{qM}, M_x, dM_x), :(\mathrm{qM}, \mathrm{qM}, 100), -(\mathrm{t}M_x, \mathrm{t}M_x, qM)\rangle\rangle,$
$\langle\langle A \to /A\rangle, \langle\times(\mathrm{q}D, D_x, dD_x), :(\mathrm{q}D, \mathrm{q}D, 100),\ +(\mathrm{t}D_x, \mathrm{t}D_x, qD)\rangle\rangle,$
$\langle\langle A \to \backslash A\rangle, \langle\times(\mathrm{q}D, D_x, dD_x), :(\mathrm{q}D, \mathrm{q}D, 100), -(\mathrm{t}D_x, \mathrm{t}D_x, qD)\rangle\rangle,$
$\langle\langle A \to xA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to yA\rangle, \langle\varepsilon\rangle\rangle,$
$\langle\langle A \to pA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to mA\rangle, \langle\varepsilon\rangle\rangle,$
$\langle\langle A \to dA\rangle, \langle\varepsilon\rangle\rangle, \langle\langle A \to uA\rangle, \langle\varepsilon\rangle\rangle,$
$\langle\langle A \to \varepsilon\rangle, \langle\varepsilon\rangle\rangle\};$

- operations over attributes:
  - addition, subtraction, multiplication and division, accordingly, $+(c,a,b)$, $-(c,a,b)$, $\times(c,a,b)$ and $:(c,a,b)$ with operands a, b and result c;
  - generation of the random normally distributed number $\wedge(c,a,b)$ with the mathematical expectation a and dispersion b.
- limitation – the rule $\langle A \rightarrow \varepsilon, \langle\varepsilon\rangle\rangle$ is observed if the other ones are not applicable;
- initial conditions – initial terminals A (for multi-character construction), B (for creating time series construction), multi-character construction $\Omega_i(C_{MS})$; initial time $t = 0$, its step $dt = 0.04$ s, current value $\mathrm{t}M_x = M_x$, $\mathrm{t}D_x = D_x$, $i = 1$;
- termination condition – observance of the empty rule.

Let's define the constructive system by interpreting C_{TS}:

$$\langle C_{TS}(\Omega_i(\mathrm{C}_{MS}), M_x, \mathrm{d}M_x, D_x, \mathrm{m}) = \langle M_{TS}, \Sigma_{TS}, \Lambda_{TS}\rangle, C_B = \langle M_B, \Sigma_B, \Lambda_B\rangle\rangle_I \mapsto$$
$$C_{B,TS}(\Omega_i(\mathrm{C}_{A,MS}), M_x, \mathrm{d}M_x, \mathrm{D}_x, \mathrm{d}D_x, \mathrm{m}) = \left\langle M_{B,TS}, \Sigma_{B,TS}, \Lambda_{B,TS}\right\rangle$$

where $M_B \supset M_A$ and supplemented by algorithms that perform operations on attributes, $\Sigma_B = \Sigma_A$, $\Lambda_B = \Lambda_A$.

Constructor-Assembler. Allows to create a constructive process in the form of a video of the formation of the flashes of lightning based on several time series. Constructor

$$C_{VL}(\mathrm{m}, n, Z_1, Z_2, \ldots, Z_{3*n}) = \langle M_{VL}, \Sigma_{VL}, \Lambda_{VL}\rangle \tag{6}$$

where m – number of time series points, M_{VL} includes a set of terminals $T = \{z_{ij}\}$, $(Z_i = [z_{i1}, z_{i2}, \ldots, z_{im}])$ and computer windows with a picture on it, nonterminals $N = \{A, B\}$, $\Sigma_{VL} = \Xi_{VL} \cup \Phi_{VL}$, $\Xi_{VL} = \{\circ, \bullet\}$, $\Phi_{VL} = \{+, >, \vee\}$, $\Lambda_{VL} = \Lambda_L \cup \Lambda_4$.

ISC Λ_4 includes the above terms, definitions and their semantics, as well as provisions below:

- ontology complemented by known concepts allowing to operate with the time series, the real numbers and computer windows;
- $''\circ''$ – relation between adjacent array elements;
- purpose of construction is video of lightning flashes on computer window;
- rules of substitution:
  $\{\langle\langle A \rightarrow z_{i,j} \circ z_{i+1,j} \circ z_{i+2,j} \circ A,\ B \rightarrow \bullet(z_{i,j} \circ z_{i+1,j} \circ z_{i+2,j})B\rangle,$
  $\langle +(i,i,3),\ >(q,i,n),\ \vee(q,\ +(i,0,1)),\ \vee(q,\ +(j,j,1)),\rangle\rangle,$
  $\langle\langle A \rightarrow \varepsilon\rangle, \langle\varepsilon\rangle\rangle\}$;
- operations over attributes:
  - addition $+(c,a,b)$, with operands a, b and result c;
  - comparison $>(c,a,b)$, if $a > b$ result $c = true$, else – $c = false$;
  - $\vee(c,a)$ – run operations a if $c = true$;
- operation $\bullet(a,b,c)$ – visualization flash of lightning on window in such position: distance along given curve a, distance from given curve b, with imitation flash power c;

- limitation – the rule $\langle A \to \varepsilon, \langle \varepsilon \rangle \rangle$ is observed if the other ones are not applicable;
- initial conditions – initial terminals A (for time series construction), B (for creating video), initial time $t = 0$, its step $dt = 0.04$ s, $i = 1,\ j = 1$; given a curve (curves) of thunderstorm front;
- termination condition – observance of the empty rule when $i = 1, j > m$.

Let's define the constructive system by interpreting C_{VL}:

$$\langle C_{VL}(\mathrm{m}, n, Z_1, Z_2, \ldots, Z_{3*n}) = \langle M_{VL}, \Sigma_{VL}, \Lambda_{VL} \rangle, C_D = \langle M_D, \Sigma_D, \Lambda_D \rangle \rangle_I \mapsto$$
$$C_{D,VL}(\mathrm{m}, n, Z_1, Z_2, \ldots, Z_{3*n}) = \langle M_{D,VL}, \Sigma_{D,VL}, \Lambda_{D,VL} \rangle,$$

where $M_D \supset M_A$ and supplemented by algorithms that perform operations on attributes and operation $\bullet(a, b, c)$, $\Sigma_D = \Sigma_A,\ \Lambda_D = \Lambda_A$.

3.2 Modeling of Lightning Activity in Thunderstorm Front

NASA and National Oceanic and Atmospheric Administration (NOAA) regularly release images from the Geostationary Lightning Mapper (GLM) instrument onboard the GOES-16 satellite. This is the new step in weather-watching capability, because it might see lightning activity above the clouds [16].

Based on the above constructors, the program modeling the lightning activity was developed.

Testing of the developed methods for modeling lightning flashes is performed by comparing with real data obtained from the satellite. Figure 1 represents: frames of satellite video image [16] (first column), the same with the removal of background (relief and cloudiness, second column), with the flashes brought to the regular form (third column).

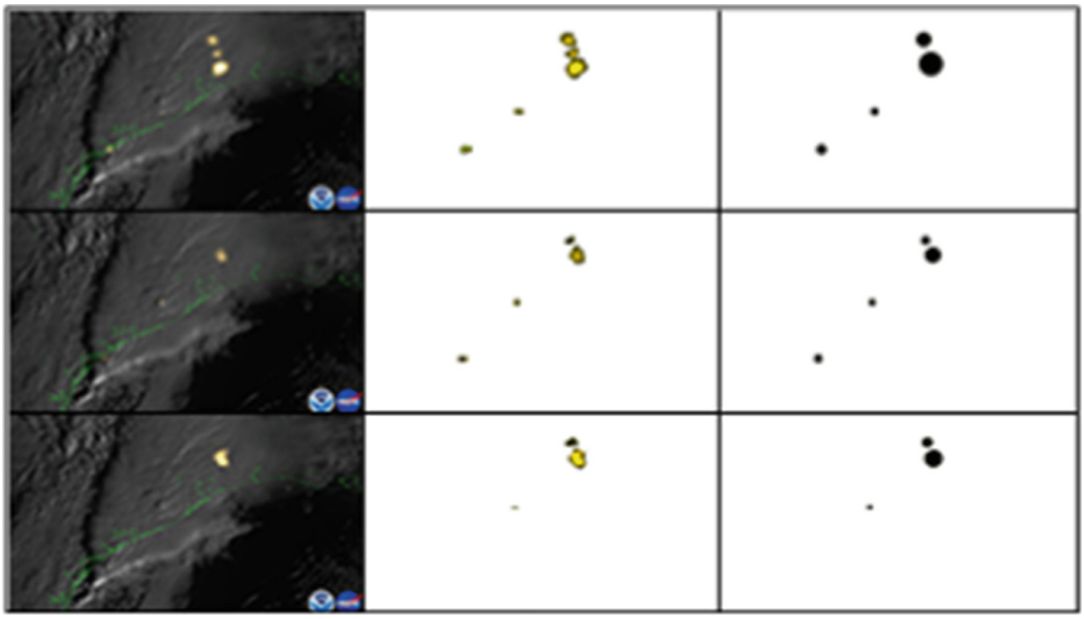

Fig. 1. Separate frames of the original video and flashes of lightning highlighted on them

For this purpose, the original video was converted as follows:

- video of the thunderstorm was divided into frames in the format RRGGBB;
- the blue channel of the images was deleted;
- the image background was removed (only flashes of lightning remain). Each pixel of the i-th frame is compared with pixel at the same position of $(i-1)$-th frame. If similarity condition $\frac{\left|C_{green,i}-C_{green,i-1}\right|}{C_{\max}}+\frac{\left|C_{red,i}-C_{red,i-1}\right|}{C_{\max}}\leq 0,1$ is doesn't true, pixel's color changes into white (here $C_{green,i}$, $C_{red,i}$ – values of green and red pixel's channels on appropriate frames, and $C_{\max}$ – the maximum possible value for each color channel);
- recursive pixel bypassing for extract highlight lightning flashes;
- smoothing out flashes, noise removal and contour and color conversion of flashes are performed;
- as a result, there is two videos of only flashes of lightning produced: a natural form and a model one having form of a circle.

A similar result was obtained as a result of modeling. The model of thunderstorm front is specified by Bezier curve. The pair of constructors $C_{A,MS}(f,\{f\rightarrow f+f\},200)_R\mapsto\Omega_1(C_{A,MS})$ and $C_{A,TS}(\Omega_1(C_{A,MS}),13,50,20,10,200)_R\mapsto\Omega_1(C_{A,TS})$ forms the time series of the lightning discharge position along the curve $u_S(t)=\Omega_1(C_{A,TS})$ (Fig. 2). The second pair $C_{A,MS}(yxyxy,\{x\rightarrow+xf,y\rightarrow-yf\},200)_R\mapsto\Omega_2(C_{A,MS})$ and $C_{A,TS}(\Omega_2(C_{A,MS}),9,5,20,10,200)_R\mapsto\Omega_2(C_{A,TS})$ – distance from the curve $u_L(t)=\Omega_2(C_{A,TS})$. The third pair $C_{A,MS}(--\backslash\backslash\backslash\backslash\backslash\backslash\backslash ppmyuumxmxxmpmmxmpxmff,\{y\rightarrow-yf,x\rightarrow+xf,f\rightarrow ff,p\rightarrow+p,m\rightarrow-m,d\rightarrow\backslash\backslash d,u\rightarrow/u\},200)_R\mapsto\Omega_3(C_{A,MS})$ and $C_{A,TS}(\Omega_3(C_{A,MS}),6,30,30,15,200)_R\mapsto\Omega_3(C_{A,TS})$ – value of the discharge $u_R(t)=\Omega_3(C_{A,TS})$. Constructed three time series (Fig. 3).

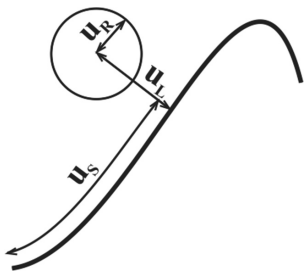

Fig. 2. Determination of the position and strength (radius of the circle) of the lightning flash

According to the constructed three time series constructor-assembler C_{VL} produced model video $C_{D,VL}(200,1,u_S(t),u_L(t),u_R(t))$. For comparison, Fig. 4 shows the lightning flashes of all frames of satellite (Fig. 4a) and model (Fig. 4b) lightning flashes.

Analysis of other satellite videos [17, 18] shows that the form of a thunderstorm cannot always be given by a single Bezier curve. Therefore, we in the constructor-assembler C_{VL} provide the possibility of assigning several corresponding series of time series for them.

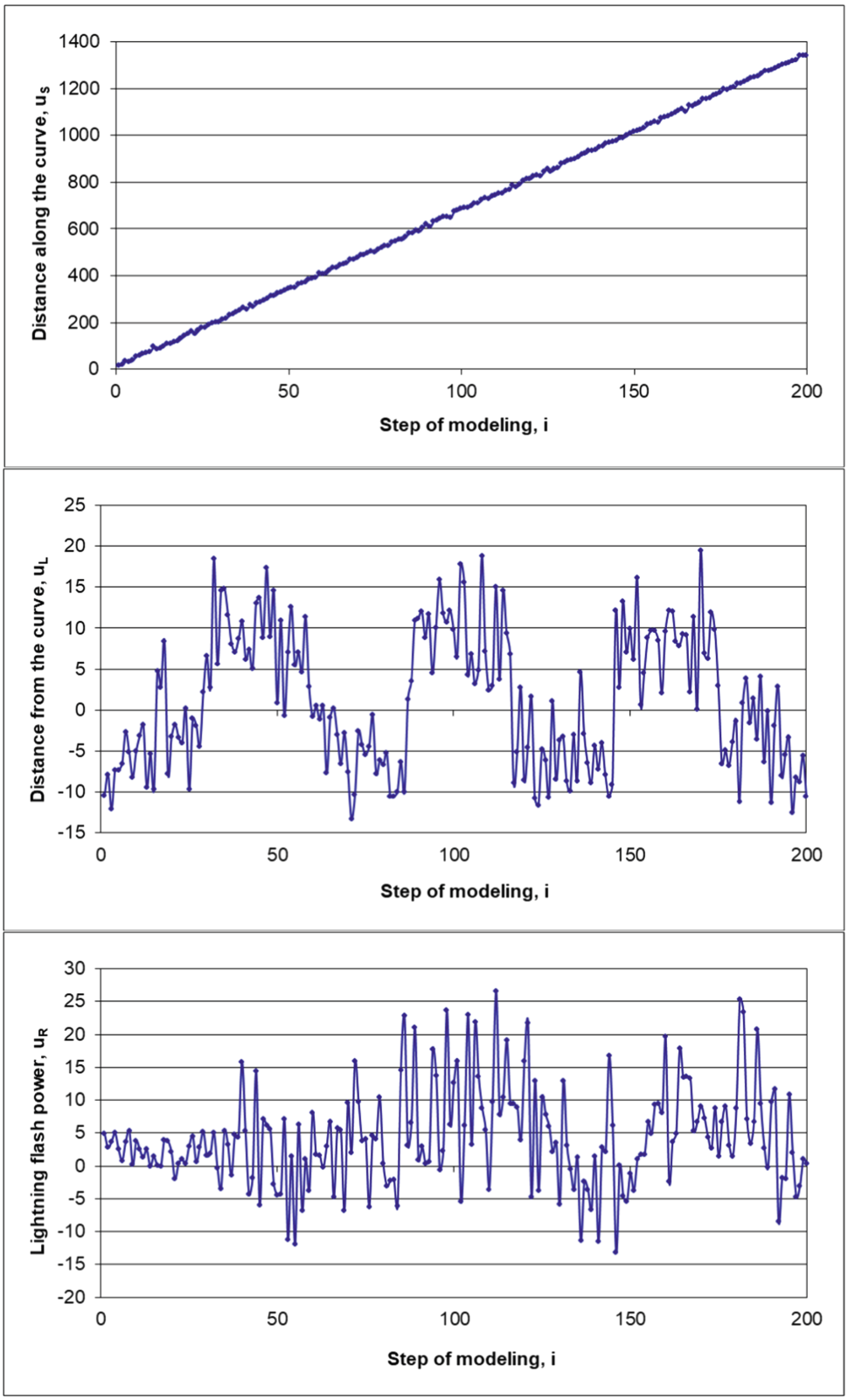

Fig. 3. Time series for distance along and from curve and lightning flash power

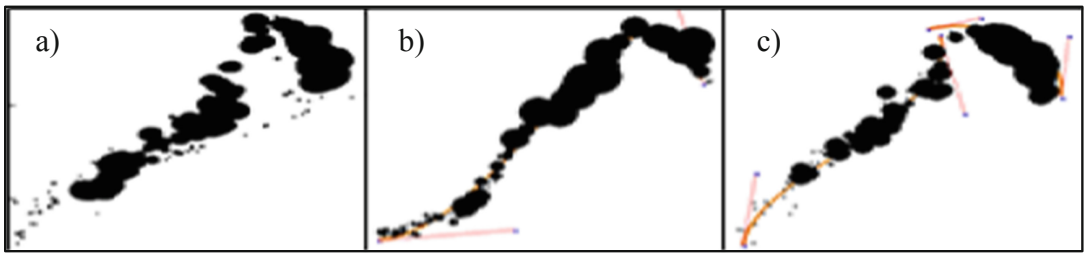

Fig. 4. Frames with all discharges of satellite and model lightning flashes

The model presented above was clarified as follows. We have two Bezier curves: long and short as an Fig. 5. Along the long curve we model two series of flashes of lightning (big and small):

- big flashes:
  - $C_{A,MS}(f, \{f \to f+f-\}, 50)_R \mapsto \Omega_4(C_{A,MS})$;
  - $C_{B,TS}(\Omega_4(C_{A,MS}), 0, 12, 150, 10, 50)_R \mapsto \Omega_4(C_{B,TS}) = u_{S,1}(t)$;
  - $C_{A,MS}(f, \{f \to f+f-f\}, 50)_R \mapsto \Omega_5(C_{A,MS})$;
  - $C_{B,TS}(\Omega_5(C_{A,MS}), 8, 12, 150, 10, 50)_R \mapsto \Omega_5(C_{B,TS}) == u_{L,1}(t)$;
  - $C_{A,MS}(zzxzzzy, \{z \to zf, y \to yf-----f, x \to xf+++++f\}, 50)_R \mapsto \Omega_6(C_{A,MS})$;
  - $C_{B,TS}(\Omega_6(C_{A,MS}), 0, 12, 150, 10, 50)_R \mapsto \Omega_6(C_{B,TS}) = u_{R,1}(t)$;
- small flashes:
  - $C_{A,MS}(f, \{f \to f+f-\}, 50)_R \mapsto \Omega_7(C_{A,MS})$;
  - $C_{B,TS}(\Omega_7(C_{A,MS}), 0, 12, 150, 10, 50)_R \mapsto \Omega_7(C_{B,TS}) = u_{S,2}(t)$;
  - $C_{A,MS}(f, \{f \to f+f-f\}, 50)_R \mapsto \Omega_8(C_{A,MS})$;
  - $C_{B,TS}(\Omega_8(C_{A,MS}), 5, 12, 8, 10, 50)_R \mapsto \Omega_8(C_{B,TS}) = u_{L,2}(t)$;
  - $C_{A,MS}(f, \{f \to f+f-\}, 50)_R \mapsto \Omega_9(C_{A,MS})$;
  - $C_{B,TS}(\Omega_9(C_{A,MS}), 1, 12, 0.1, 10, 50)_R \mapsto \Omega_9(C_{B,TS}) = u_{R,2}(t)$.

Along the shot curve we model one series of flashes of lightning:

- $C_{A,MS}(f, \{f \to f+f-\}, 50)_R \mapsto \Omega_{10}(C_{A,MS})$;
- $C_{B,TS}(\Omega_{10}(C_{A,MS}), 15, 12, 30, 10, 50)_R \mapsto \Omega_{10}(C_{B,TS}) = u_{S,3}(t)$;
- $C_{A,MS}(f, \{f \to f+f-\}, 50)_R \mapsto \Omega_{11}(C_{A,MS})$;
- $C_{B,TS}(\Omega_{11}(C_{A,MS}), 15, 12, 30, 10, 50)_R \mapsto \Omega_{11}(C_{B,TS}) = u_{L,3}(t)$;
- $C_{A,MS}(f, \{f \to f+f-\}, 50)_R \mapsto \Omega_{12}(C_{A,MS})$;
- $C_{B,TS}(\Omega_{12}(C_{A,MS}), 15, 12, 30, 10, 50)_R \mapsto \Omega_{12}(C_{B,TS}) = u_{R,3}(t)$.

According to the constructed time series constructor-assembler C_{VL} produced model video $C_{D,VL}(50, 3, u_{S,1}(t), u_{L,1}(t), u_{R,1}(t), u_{S,2}(t), u_{L,2}(t), u_{R,2}(t), u_{S,3}(t), u_{L,3}(t), u_{R,3}(t))$. On Fig. 5 shows the video frames created by the constructor C_{VL}.

These frames to some extent correspond to the frames from the satellite video (Fig. 1, column 3). Exact coincidence is not expected since the real and model processes are stochastic. However, all flashes of lightning from satellite video (Fig. 4a) and from video of improved model (Fig. 4c) correlate quite well.

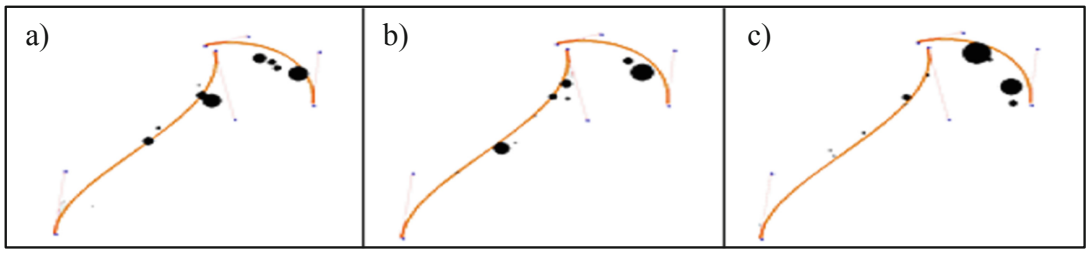

Fig. 5. Video frames of lightning flashes by improved model

4 Conclusions

Usage of modeling in the formation of lightning discharges based on the constructive-synthesizing approach allows obtaining the realistic description of the thunderstorm front lightning activity. This approach can be the basis for solving the dynamic problem on lightning protection of engineering constructions and civil objects, and development of strategy of aircraft behavior in order to mitigate the risks of lightning strokes in the conditions of movement in the thunderstorm front.

References

1. Rodger, C.J., Werner, S., Brundell, J.B., Lay, E.H.: Detection efficiency of the VLF World-Wide Lightning Location Network (WWLLN): initial case study. Ann. Geophys. **24**, 3197–3214 (2006)
2. Kraaij, T., Cowling, R.M., van Wilgen, B.W.: Lightning and fire weather in eastern coastal fynbos shrublands: seasonality and long-term trends. Int. J. Wildland Fire **22**, 288–295 (2013). https://doi.org/10.1071/WF11167
3. Pack, S., Piantini, A.: Lightning research and lightning protection technology. Electr. Power Syst. Res. **113**, 1–2 (2014)
4. Williams, E.R.: Lightning and climate: a review. Atmos. Res. **76**(1–4), 272–287 (2005). https://doi.org/10.1016/j.atmosres.2004.11.014
5. Yair, Y., Lynn, B., Price, C., Kotroni, V., Lagouvardos, K., Morin, E., Mugnai, A., Llasat, M.d.C.: Predicting the potential for lightning activity in Mediterranean storms based on the Weather Research and Forecasting (WRF) model dynamic and microphysical fields. J. Geophys. Res. **115** (2010). D04205. https://doi.org/10.1029/2008JD010868
6. Sátoria, G., Williams, E., Lempergeraams, I.: Variability of global lightning activity on the ENSO time scale. Atmos. Res. **91**(2–4), 500–507 (2009). https://doi.org/10.1016/j.atmosres.2008.06.014
7. Devendraa, S., RameshKumara, P., Kulkarnia, M.N., Singhb, R.P., Singhc, A.K.: Lightning, convective rain and solar activity — over the South/Southeast Asia. Atmos. Res. **120–121**, 99–111 (2013). https://doi.org/10.1016/j.atmosres.2012.07.026
8. Galanaki, E., Kotroni, V., Lagouvardos, K., Argiriou, A.: A ten-year analysis of cloud-to-ground lightning activity over the Eastern Mediterranean region. Atmos. Res. **166**, 213–222 (2015)
9. Ahrens, M.: Lightning fires and lightning strikes. In: National Fire Protection Association, Quincy, 31 p. (2013)

10. Fuchs, B.R., Bruning, E.C., Rutledge, S.A., Carey, L.D., Krehbiel, P.R., Rison, W.: Climatological analyses of LMA data with an open-source lightning flash-clustering algorithm. J. Geophys. Res. Atmos. **121**(14), 8625–8648 (2016)
11. Shynkarenko, V.I., Ilman, V.M.: Constructive-synthesizing structures and their grammatical interpretations. Part I. Generalized formal constructive-synthesizing structure. Cybern. Syst. Anal. **50**(5), 655–662 (2014). https://doi.org/10.1007/s10559-014-9655-z
12. Shynkarenko, V.I., Ilman, V.M.: Constructive-synthesizing structures and their grammatical interpretations. Part II. Refining transformations. Cybern. Syst. Anal. **50**(6), 829–841 (2014). https://doi.org/10.1007/s10559-014-9674-9
13. Shynkarenko, V.I., Ilman, V.M., Skalozub, V.V.: Structural models of algorithms in problems of applied programming. I. Formal algorithmic structures. Cybern. Syst. Anal. **45** (3), 329–339 (2009). https://doi.org/10.1007/s10559-009-9118-0
14. Skalozub, V., Illman, V., Shynkarenko, V.: Development of ontological support of constructive-synthesizing modeling of information systems. Eastern-Eur. J. Enterp. Technol. **6/4**(90), 58–69 (2017). https://doi.org/10.15587/1729-4061.2017.119497
15. Lindenmayer, A.: Mathematical models for cellular interaction in development. J. Theor. Biol. **18**, 280–315 (1968)
16. First Images from GOES-16 Lightning Mapper. https://www.americaspace.com/2017/03/07/goes-16-satellite-returns-first-lightning-mapping-images-like-never-seen-before/. Accessed 11 July 2018
17. Regional and Mesoscale Meteorology Branch. http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes-16/20190116000000&number_of_images_to_display=120&loop_speed_ms=100. Accessed 11 July 2018
18. Regional and Mesoscale Meteorology Branch. http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes-16/20180815000000&number_of_images_to_display=100&loop_speed_ms=150. Accessed 11 July 2018

Constructive-Synthesizing Modeling of Lightning Flashes in the Dynamic Thunderstorm Front

Viktor Shynkarenko(✉), Iryna Nikitina, and Robert Chyhir

Dnipro National University of Railway Transport named after Academician V. Lazaryan, Dnipro 49010, Ukraine
shinkarenko_vi@ua.fm, irinasansieva@gmail.com, robertchigir@ukr.net

Abstract. At first, deep analysis of modeling processes of the lightning flashes dynamic behavior in static background and moving thunderstorm fronts with very mobile clouds at NASA satellite videos was done. Color models' opportunities were explored for lightning flashes extraction in significant dynamic thunderstorm fronts and cloudiness. It was shown that the greatest efficiently in recognition is provided by combining Lab and LCH model-based features. The ranges of color channels were defined for lightning aureoles. This data was used for their detection at the series frames from meteorological satellites. The lightning detection filtering is based on the processing a current frame of the video with linear and quadratic filters were developed to solve these tasks. An analysis of their effectiveness was done. Modeling of the lightning flashes was implemented using constructive-synthesizing approach. A set of constructors was developed. Implementing parametric multi-character constructors allows forming fractal sequences of characters. Constructor-converter from the character string to time series creates fractal time series, which determine the location, magnitude and decay rate of lightning discharges. Model video images of lightning in the thunderstorm front are formed in accordance with the implementation of the constructor-assembler. The methods and software for lightning extraction from NASA satellites video, and also for realization constructive-synthesizing models were developed.

AQ1

Keywords: Constrictive-synthesizing modeling · Fractal · Lightning flash · Thunderstorm front · Time series · Color model · Image recognition

1 Introduction

Meteorological satellites are widely used in order to investigate and forecast the behavior of severe weather. Currently, to automate the process of the shots series analysis new algorithms are developed [1]. Much of the research related to processing data from weather satellites is devoted to detection [2, 3] or classification [4, 5] of the clouds' types. These studies are based on analysis of shape, color and cloud's motion using different methods of pattern recognition.

N. Shakhovska and M. O. Medykovskyy (Eds.): CSIT 2020, AISC 1293, pp. 1–18, 2021.
https://doi.org/10.1007/978-3-030-63270-0_76

In previous papers [6, 7] we presented the constructive-synthesizing modeling of lightning activity. Models are based on the video produced by NASA satellites where the cloudiness and thunderstorm front are static. This factor was taken into account in the algorithms for lightning detection and their modeling.

These studies are a follow up on the previous ones. The features of the movement of thunderstorm fronts with very mobile clouds were studied. This significantly complicated the processing of video from satellites and required clarification of constructive-synthesizing models.

2 State of Problem

Regional behavior of lightning activity has been studied for a long time [8].

The effect of solar variability parameters and meteorological parameters on total lightning flashes and convective rain in the two selected regions has been studied in the article [9], isolation of zones of the lightning activity in specific geographic areas [10], and their impact on breaking-out of fires [11]. Much smaller number of works deals with the problem of modeling of lightning in the thunderstorm front, which is primarily due to its complexity. Typically, such works are limited to isolation of compact zones (clusters) of lightning formation [12].

The problems of modeling cloudiness without the possibility of lightning activity are considered separately. So, in [13] cloud-resolving model allows performing numerical simulations of convective clouds, such as shallow cumulus and stratocumulus, or storms and squall-lines with a resolution on the order of a few tens of metres to a few kilometres.

Research of lightning density is widely used by modelers in an attempt to forecast electrical activity in thunderstorms. In particular [14], parameterization is the subject for the estimation of lightning density of convective clouds.

This paper refers to modeling of lightning activity in moving thunderstorm fronts with very mobile clouds based on the generated fractal time series which determine the time, coordinates and duration of flashes, and comparison of the model video images to video images received from the satellite.

There is an opportunity to understand the constructive notion of the real technical or nature object in the constructive approach to the formation of objects of different nature.

3 Study of Lightning Fronts from Satellites Video

The research was initialized with deep analysis of modeling processes of the lightning flashes dynamic behavior in static background and moving thunderstorm fronts with very mobile clouds at satellite videos.

3.1 Lightning Recognition at Frames with a Static Background

Some part of the video demonstrates lighting thunderstorm with no cloudiness and has a static background [15]. Consider one of these cases.

The first row at Fig. 1 demonstrates the video images through 150 frames. There are only changes in flashes, while the background is the same.

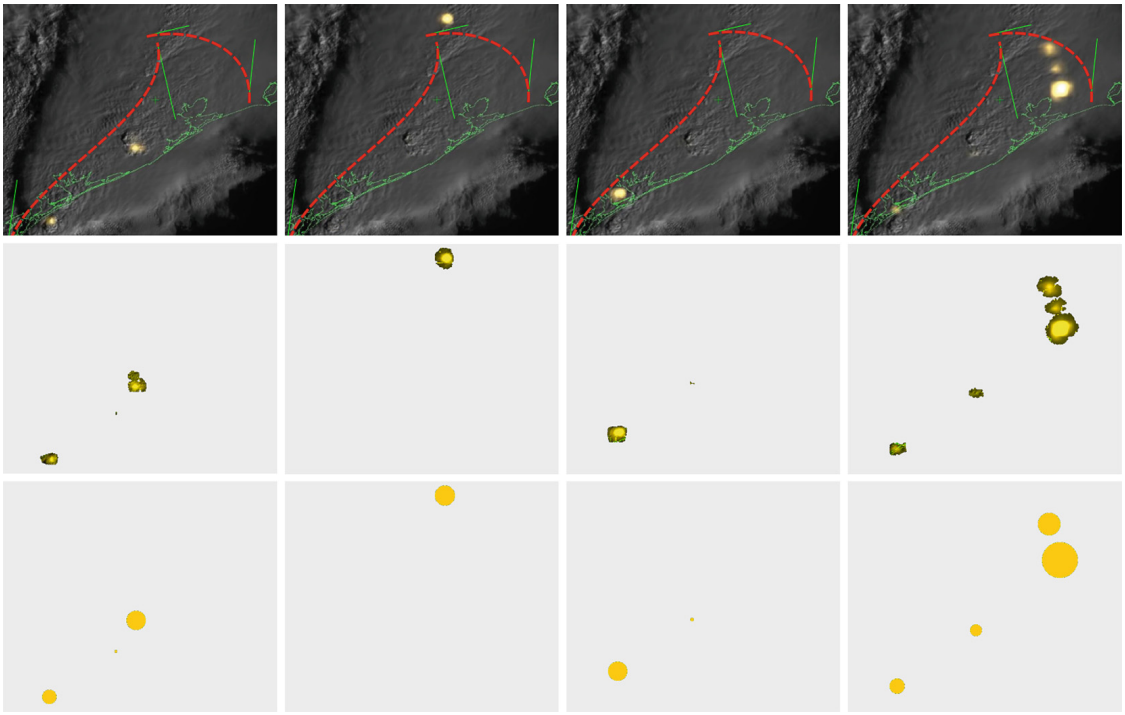

Fig. 1. Frames of NASA satellite video of lighting thunderstorm with a static background

So finding dynamic objects means lightning detection. The algorithm for solving this task uses the previous frame when processing the current one. To highlight lightning flashes in video frames, image filtering is performed as follows (color model is RGB):

$$y_{ij} = \begin{cases} x_{ij} \ if \ d_{ij} > 0 \\ 0 \text{ otherwise,} \end{cases} \tag{1}$$

where x_{ij} – is a pixel's value with coordinates $(i,\ j)$ at the frame before processing, y_{ij} – is the respective value at the formed video image containing only lightning contours, and

$$d_{ij} = l - \sum_{k=1}^{K} \frac{|x_{ijk} - z_{ijk}|}{c_{\max}}, \tag{2}$$

where x_{ijk}, z_{ijk} – is a pixel's value with coordinates $(i,\ j)$ in k channel of RBG color model at the current and previous frame before processing, l – is tolerance, K – limits the channel' quantity that is used for current video processing and $c_{\max}$ – is the maximum possible value for the color channel.

To process the examined video the next values of constant variables were used:

- $l = 0,\ 1$ – was obtained by experimentation;
- $K = 2$ – means that only red (k = 1) and green (k = 2) channels are considered (as video images are represented in yellow tones, so blue channel is useless);
- $c_{\max} = 256$ as it was enough 8 bits per color to differ.

The described filter is quite simple and all constant variables values were obtained after a few experiments number. It is worth noting, that separation ability of the RGB color model (one of the commonly used) is enough to represent the difference between pixels correctly.

However such an approach, due to some comparison inaccuracy, leads to few type I and type II errors. Type I errors are represented by holes inside the lightning where it was bright and strong flash making pixels color in the central part almost the same through several consecutive frames. To address the problem it is required to process each obtained image one more time. During the image traversing 8 neighbor pixels are checked for each one with background color: if half or more of them are not painted as background, the pixel color is changed to any other.

Type II errors are in contrast to type I and could be considered as noises. Colored pixels not constituting lightning indeed belong to them. In order to remove them the similar algorism could be used: if more than half of 8-pixel neighbors are painted as background, its value is also changed in order to be equal them. Finally, the stage of errors removal consists of the following steps:

1. the noise elimination algorithm is applied several times for all frames, which, in addition to removing noise, smoothies the shape of the lightning;
2. once for each frame rebuilding deleted lightning parts algorithm is performed;
3. step 1 is repeated in order to smooth the edges of lightning's after step 2 was executes.

The images after removal of background (relief and cloudiness) using the above approach are shown at the second column, Fig. 1 for respective frames above. As we can see there are no errors and the lightning contours are smooth enough. Then all flashes were brought to the regular form (third row, Fig. 1).

Unfortunately, the described approach works correctly only for video with a static background and cannot be used to process shots that demonstrate dynamic cloudiness.

3.2 Lightning Recognition at Frames of Satellites Video in Dynamic Clouds Front

Separation Ability of Color Models in Lightning Flashes Extraction. In order to detect the lightning flashes in the dynamically changing cloudiness the classification of the changed pixels could be applied [16], but not always pixels that are lightning part change their color (the bright flashes central part as a rule is almost the same during several frames). The other approach uses particular properties of its representation by current satellite [17, 18], where the luminous strip is detected as lightning if it has sixteen pixels in its width. Despite this method cannot be applied to detect lightning at the frames with high resolution, the approach using lightning particular properties at the frames from the current satellite can be applied.

At the video images filtered during the shooting from satellites [19, 20] lightning are demonstrated as bright spots with colorful contour.

The analysis of lightning color from the video didn't give the positive result even after images' preprocessing such as removing color channel, edge detection using the difference of Gaussians and their varied combination was done. But it was found that

the main particular property of the given videos is the clear defined aureole around the lightning. That's why the lightning contour was analyzed instead of them.

The color characteristics of the object for identification can be considered in the pattern recognition problem, while the color model matters. Studies are being conducted to search the most effective model in different fields of science, for example: skin lesion segmentation is based on the RGB and XYZ models [20], the search of particular properties for damage regions in retinopathy was done in HLS, LCH and Lab models [22], Lab and HSV color models were the most informative in detecting roads segmentation for satellite aerial images [23]. We defined the most perspective color models to solve the task, they are RGB, HSV, HSL, Lab и LCH.

Lab and LCH color models were found as the most informative. In [24] to detect the color difference scheme for applying color models was proposed as at Fig. 2, we modified it (Fig. 3).

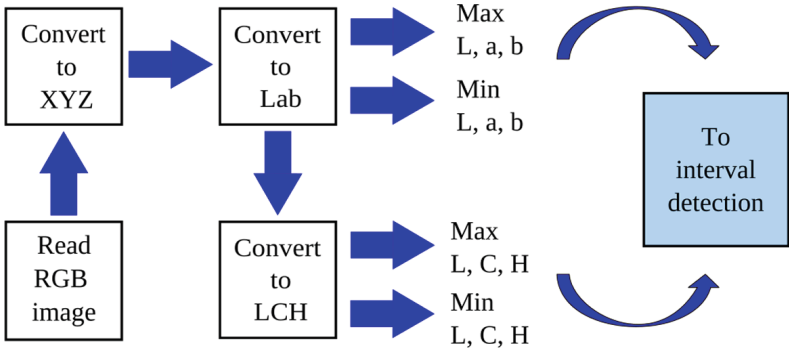

Fig. 2. The process diagram to color difference

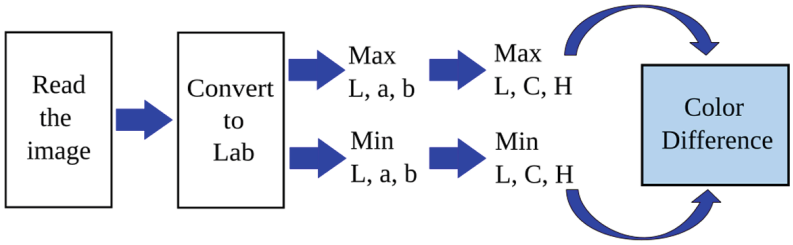

Fig. 3. The process diagram to interval detection

Five video images from [19] and [20] were selected for analysis. It was established that the lightning itself is not noticeable, while the color of the contour stands out at the image and is suitable to identify lightning. The research results showed that to identify the contour it should be considered in Lab and LCH color models as they are the best to represent the contour of the particular properties since low values of the exponent b are present only at the contours, and high values are present only in a small amount outside it (correlation of pixels amount to b value for contour and the entire image is shown with blue and green respectively at Fig. 4), and the exponent H is a quite narrow part of the scale.

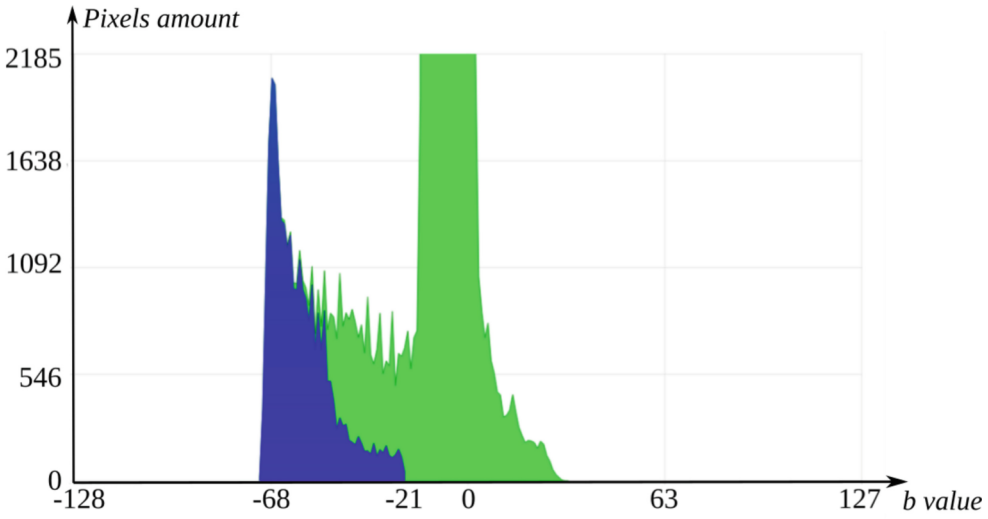

Fig. 4. Pictograph of the color frequency in the channel ***b*** (Lab) frame 1

The values range in these models are presented in Table 1. L parameter has the same sense for both models.

Table 1. The lightning aureoles color ranges in Lab и LCH models

	L	a	b	C	H
Frame 1 [18]	(24, 85)	(1, 42)	(−68, −21)	(19, 79)	(274, 302)
Frame 2 [18]	(22, 89)	(−2, 40)	(−69, −17)	(14, 80)	(270, 299)
Frame 3 [18]	(24, 90)	(0, 41)	(−67, −14)	(18, 78)	(272, 301)
Frame 4 [19]	(21, 92)	(0, 43)	(−61, −13)	(10, 79)	(270, 302)
Frame 5 [19]	(25, 87)	(2, 39)	(−63, −15)	(8, 82)	(273, 301)
Average	(23, 89)	(0, 41)	(−66, −16)	(14, 80)	(272, 301)
Interval	**(22, 91)**	**(−3, 44)**	**(−70, −12)**	**(11, 81)**	**(268, 303)**
%scale	**69**	**18**	**23**	**70**	**10**

The average value of the range was calculated for manually extracted lightning aureoles at the selected video images (row “Average” in Table 1):

$$c_{lk} = \sum_{i=1}^{5} c_{ilk}/5; \;\; c_{uk} = \sum_{i=1}^{5} c_{iuk}/5, \tag{3}$$

where (c_{ilk}, c_{iuk}) – are the low and the upper bound of the color for i-th frame in the k channel.

The range for extracting lightning aureoles is somewhat expanded (row "Interval" in Table 1):

$$l_k = c_{lk} + 0,\ 005 \cdot R_k \cdot \min_i(c_{ilk} - c_{lk}),$$
$$u_k = c_{uk} + 0,\ 005 \cdot R_k \cdot \max_i(c_{iuk} - c_{uk}), \tag{4}$$

where 0,005 – is the expansion coefficient obtained by experimentation, R_k – number possible values in color channel k.

Image filtering will be performed using both models, since many papers show that the accuracy of object recognition increases when combining two [25], three [26] and even seven [27] models..

Table 1 also shows the part (%) of the scale-covered by the calculated range (4).

Lightning Flashes Extraction. The next filters were developed in order to detect lightning aureoles.

The linear filter function (1) where:

$$d_{ij} = \sum_{k=1}^{K} f(x_{ijk},\ l_k,\ u_k) - k + 1, \tag{5}$$

$$f(x_{ijk},\ l_k,\ u_k) = \begin{cases} 1\ if\ \ l_k < x_{ijk} < u_k \\ 0\ \text{otherwise}, \end{cases} \tag{6}$$

where $K = 5$, x_{ijk} – is a pixel's value with coordinates (i, j) using

- $k = 1$ for the L color channel in Lab and LCH models;
- $k = 2$ for the a color channel in Lab model,
- $k = 3$ for the b color channel in Lab model;
- $k = 4$ for the C color channel in LCH model;
- $k = 5$ for the H color channel in LCH model.

Quadratic filter function: (1), (7):

$$d_{ij} = 1 - \sum_{k=1}^{K} \frac{(x_{ijk} - (l_k + u_k)/2)^2}{(u_k - l_k)^2/4}. \tag{7}$$

The values presented in Table 1 were substituted into the filter functions and all video images [19, 20] were filtered using each of them. Figure 5 shows the results of contour extraction for the middle frame at Fig. 1 using linear filter (1), (5) at Fig. 5 a) and quadratic one (1), (7) at Fig. 5 b).

To find out the most effective filter, contours were extracted manually from the source image and after that they were compared with automatically detected ones. The result is at Fig. 5 a) and b) for linear and quadratic filters respectively. The pixels are marked with

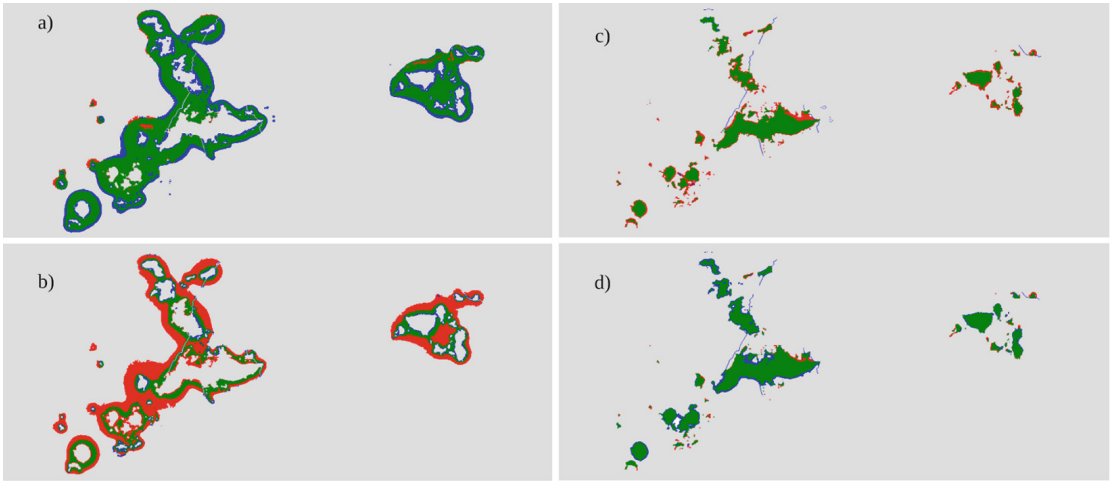

Fig. 5. Extracted lightning aureoles and the lightning themselves

green if they represent aureole at both manually created and filtered image. The gray pixels show the same case for the image background. Lightning aureole at the manually created frame and background at the filtered one is marked with red, and the opposite case is shown with blue.

As we can see the quadratic function cuts out more than the middle, which is good for small dim lightning, because the linear function does not leave a cavity in the contour, which leads to the fact that the lightning will be lost. Also, the linear filter leaves a background that cannot be attributed to the contour, which will lead to the appearance of "imaginary lightning".

Then it is required to extract the lightning. To do this the following algorithm was developed: pixels are analyzed starting from the upper left corner in left-to-right scanline order. It is considered that pixel with background color belongs to lightning if:

$$z_{ij} = \begin{cases} x_{ij} \ if \ \sum\limits_{k=j-1}^{j=1} \ \sum\limits_{n=i-1,\ n\neq i \& k\neq j}^{i=1} \delta(y_{n,k}) \geq 3 \\ x_{ij} \ if \ \sum\limits_{k=j-1}^{j=1} \ \sum\limits_{n=i-1,\ n\neq i \& k\neq j}^{i=1} (\delta(y_{n,k}) + \delta(z_{n,k})) \geq 4 \\ 0 \text{ otherwise,} \end{cases} \tag{8}$$

where x_{ij} is a pixel's value with coordinates $(i,\ j)$ at the source frame, y_{ij} is the respective value at the filtered frame with extracted lightning aureoles, z_{ij} is the pixel's value at the frame with extracted lightning flashes, and

$$\delta(a) = \begin{cases} 1 \ if \ a \neq 0 \\ 0 \text{ otherwise.} \end{cases}$$

Such approach allows processing even non-closed contours as it is in the case with small lightning which loss is critical for the second video [20].

However, besides of lightning, parts from the contours outside also appear at the resulting image. To solve the problem the color properties were analyzed for lightning

as it had been done for contour using the same five images. The obtained value ranges and the resulting interval are presented in Table 2. The last is calculated in the same way as for Table 1. Table 2 doesn't have the H-value column as its values don't form one interval.

Table 2. The lightning color ranges in Lab и LCH models

	L	a	b	C
Frame 1 [18]	(52, 100)	(−5, 20)	(−48, 1)	(0, 51)
Frame 2 [18]	(42, 100)	(−5, 19)	(−50, 1)	(0, 54)
Frame 3 [18]	(56, 100)	(−3, 12)	(−43, 2)	(0, 46)
Frame 4 [19]	(48, 99)	(−1, 10)	(−27, 3)	(0, 28)
Frame 5 [19]	(41, 100)	(−1, 17)	(−40, 2)	(0, 43)
Average	(48, 100)	(−3, 16)	(−42, 2)	(0, 44)
Interval	**(45, 100)**	**(−6, 21)**	**(−52, 3)**	**(0, 49)**

So in order to detect lightning correctly (without noises) (1) can be used with filters (5) or (7) using K equal to five in (5) and (7) and free variable equal to three in (5) as we have four color channels instead of five now. New coefficients for filters are recalculated by developed software automatically after changing the channels' quantity that influences the result.

However the video images have lines showing the regional boundaries which have the same colors characteristic as lightning. To solve this problem approach developed for detecting lightning with the static background was used as these lines don't move from frame to frame. In this way several approaches are combined with the aim of lightning detecting. The resulting algorithm consists of the next steps:

1. Forming lightning aureoles using (1) with (5) or (7) substituting values presented in Table 1.
2. Selecting pixels suspicious for lightning ones using (8).
3. Check if pixel belongs to lightning indeed: firstly using (1) with (5) or (7) substituting values from Table 2; and then, if the result is positive, check whether pixel doesn't belong to region border line with (1) and (2).

Lightning pixel's color corresponds to the pixel's color at the source image. The images that contain extracted lightning from the source frames are the result of the proceedings. The series of experiments show that both filters (5) or (7) with parameters represented in Table 2 work equally good for lightning detection, so it was decided to use filter (5) as it is simpler due to its linearity. Figure 5 c) and d) show the results of lightning extraction for the middle frame at Fig. 1 after the contours were detected using linear and quadratic filters respectively. The comparison with manually prepared lightning from the source image was done. The colors at Fig. 5 c) and d) have the same sense as for contours comparison.

In addition, in a consequence there were found new video representing lights in the same way as the reviewed two videos above [27–29]. The developed filters with the same parameters' values were used to detect lightning at the video frames.

3.3 Constructive-Synthesizing Modeling of Lightning Flashes

These studies are a follow up on the previous [6, 7]. Here constructive-synthesizing models were clarified. Based on generalized constructor [31–34] a set of constructors was developed. Implementing parametric multi-character constructors allows forming fractal sequences of characters. Constructor-converter from the character string to time series creates fractal time series, which determine the location, magnitude and decay rate of lightning discharges. Model video images of lightning in the thunderstorm front are formed in accordance with the implementation of the constructor-assembler.

Parametric Multi-character Creator Constructors. Concretization of C_L to the level of the family of parametric multi-character constructors gives

$$C_L = \langle M_L,\ \Sigma_L,\ \Lambda_L \rangle_K \mapsto C_{MS}(\mathrm{B,P,n}) = \langle M_{MS},\ \Sigma_{MS},\ \Lambda_{MS} \rangle \tag{9}$$

where $C_L = \langle M_L,\ \Sigma_L,\ \Lambda_L \rangle$ – constructor based on the constructive-synthesizing approach and L-systems [6], B – initial character string (axiom), P – set of substitution rules, n – the minimum number of terminals f in the output string, $M_{MS} \supset \{f,\ x,\ y,\ z,\ +,\ -\}$, $\Sigma_{MS} = \Xi_{MS} = = \{\circ\}$, $\circ$ – concatenation operation, $\Lambda_{MS} = \Lambda_L \cup \Lambda_1$. The ontological component Λ_1 includes the above terms and their semantics, as well as provisions below:

- purpose of construction – formation of a string of fractal structure;
- substitution rules are set by the parameter P;
- limitations – there are no operations over attributes;
- initial conditions – the axiom is specified by B;
- termination condition – number of terminals f in the output string $\geq$ n.

As a result of interpretation, we form the constructive system as a set of two models: constructor and internal performer as in [6].

Constructor-Converter from the Character String to time Series. The family of such constructors

$$C_{TS}(\Omega_i(\mathrm{C}_{MS}),\ M_x,\ dM_x,\ D_x,\ d\mathrm{D}_x m) = \langle M_{TS},\ \Sigma_{TS},\ \Lambda_{TS} \rangle \tag{10}$$

where $\Omega_i(\mathrm{C}_{A,MS})$ – strings obtained as a result of the implementation of the constructor $\mathrm{C}_{A,MS}$; M_x – initial value of mathematical expectation of the time series value, $\mathrm{d}M_x$ – its increment (%), D_x – initial value of the dispersion of the time series, $\mathrm{d}D_x$ – its increment (%), m – number of time series points, M_{TS} includes a set of terminals $T = \{f,\ x,\ y,\ z,\ +,\ -\}$), nonterminals $N = \{A\}$, $\Sigma_{TS} = \Xi_{TS} \cup \Phi_{TS}$, $\Xi_{TS} = \{\circ, f\}$, $\Phi_{TS} = \{+,\ -\}$, $\Lambda_{TS} = \Lambda_L \cup \Lambda_2$.

Let's introduce the operations over attributes:

- addition, subtraction, multiplication and division, accordingly, $+(c,\ a,\ b)$ and $-(c,\ a,\ b)$ with operands a, b and result c;

- generation of the random normally distributed number $\wedge(c,\ a,\ b)$ with the mathematical expectation a and dispersion b.

Information support of construction (ISC) Λ_2 includes the above terms, definitions and their semantics, as well as provisions below:

- ontology complemented by known concepts allowing to operate with the time series and the real numbers;
- "$\circ$" – the relation between adjacent array elements;
- purpose of construction is the formation of the time series $v(t)$;
- rules of substitution:

$$\{\langle\langle A \to fA,\quad B \to z_i \circ B\rangle,\ \langle\wedge(v,\ \mathrm{t}M_x,\ tD_x),\ +(t,\ t,\ dt),\ +(i,\ i,\ 1)\rangle\rangle,$$
$$\langle\langle A \to +A\rangle,\ \langle\times(\mathrm{q}M,\ M_x,\ dM_x),\ :(\mathrm{q}M,\ \mathrm{q}M,\ 100),\ +(\mathrm{t}M_x,\ \mathrm{t}M_x,\ qM)\rangle\rangle,$$
$$\langle\langle A \to -A\rangle,\ \langle\times(\mathrm{q}M,\ M_x,\ dM_x),\ :(\mathrm{q}M,\ \mathrm{q}M,\ 100),\ -(\mathrm{t}M_x,\ \mathrm{t}M_x,\ qM)\rangle\rangle,$$
$$\langle\langle A \to xA\rangle,\ \langle\varepsilon\rangle\rangle,\ \langle\langle A \to yA\rangle,\ \langle\varepsilon\rangle\rangle,\ \langle\langle A \to zA\rangle,\ \langle\varepsilon\rangle\rangle,\ \langle\langle A \to \varepsilon\rangle,\ \langle\varepsilon\rangle\rangle\};$$

- operations over attributes:

  – addition, subtraction, multiplication and division, accordingly, $+(c,\ a,\ b)$ and $-(c,\ a,\ b)$ with operands $a,\ \ b$ and result c;
  – generation of the random normally distributed number $\wedge(c,\ a,\ b)$ with the mathematical expectation a and dispersion b.

- limitation – the rule $\langle A \to \varepsilon,\ \langle\varepsilon\rangle\rangle$ is observed if the other ones are not applicable;
- initial conditions – initial terminals A (for multi-character construction), B (for creating time series construction), multi-character construction $\Omega_i(\mathrm{C}_{MS})$; initial time $t = 0$, its step $dt = 0.04$ seconds, current value $\mathrm{t}M_x = M_x$, $\mathrm{t}D_x = D_x$, $i = 1$;
- termination condition – observance of the empty rule.

Define the constructive system by interpreting C_{TS} as in [6].

Constructor-Assembler. Allows creating a constructive process in the form of a video of the formation of the flashes of lightning based on several time series. Constructor

$$C_{VL}(\mathrm{m},\ n,\ Z_1,\ Z_2,\ \ \ldots,\ Z_{3*n}) = \langle M_{VL},\ \Sigma_{VL},\ \Lambda_{VL}\rangle \tag{11}$$

where m – the number of time series points, n – the number of moving thunderstorm fronts, M_{VL} includes a set of terminals $T = \{z_{ij}\}$), $(Z_i = [z_{i\,1},\ \ z_{i\,2},\ \ \ldots,\ \ z_{i\,m}])$ and computer windows with a picture on it, nonterminals $N = \{A,\ B\}$, $\Sigma_{VL} = \Xi_{VL} \cup \Phi_{VL}$, $\Xi_{VL} = \{\circ,\ \bullet\}$, $\Phi_{VL} = \{+,\ >,\ \vee\}$, $\Lambda_{VL} = \Lambda_L \cup \Lambda_3$.

ISC Λ_3 includes the above terms, definitions and their semantics, as well as provisions below:

- ontology complemented by known concepts allowing to operate with the time series, the real numbers and computer windows;

- "$\circ$" – the relation between adjacent array elements;
- purpose of construction is the video of lightning flashes on the computer window;
- rules of substitution:

$$\{\langle\langle A \to z_{i,j} \circ z_{i+1,j} \circ z_{i+2,j} \circ A,\ \ B \to \bullet(z_{i,j} \circ z_{i+1,j} \circ z_{i+2,j})B\rangle,$$
$$\langle +(i,\ i,\ 3),\ > (q,\ i,\ n),\ \vee(q,\ +(i,\ 0,\ 1)),\ \vee(q,\ +(j,\ j,\ 1)),\rangle,$$
$$\langle\langle A \to \varepsilon\rangle,\ \langle\varepsilon\rangle\rangle\};$$

- operations over attributes:

  – addition $+(c,\ a,\ b)$, with operands a, b and result c;
  – comparison $> (c,\ a,\ b)$, if $a > b$ result $c = true$, else – $c = false$;
  – $\vee(c,\ a)$ – run operations a if $c = true$;

- operation $\bullet(a,\ b,\ c)$ – visualization flash of lightning on the window in such position: distance along given curve a, distance from given curve b, with imitation flash power c;
- limitation – the rule $\langle A \to \varepsilon,\ \langle\varepsilon\rangle\rangle$ is observed if the other ones are not applicable;
- initial conditions – initial terminals A (for time series construction), B (for creating video), initial time $t = 0$, its step $dt = 0.04$ s, $i = 1$, $j = 1$; given a curve (curves) of thunderstorm front;
- termination condition – observance of the empty rule when $i = 1$, $j > m$.

Let's define the constructive system by interpreting C_{VL}:

$$\langle C_{VL}(\mathrm{m},\ n,\ Z_1,\ Z_2,\ \ \ldots,\ Z_{3*n}) = \langle M_{VL},\ \Sigma_{VL},\ \Lambda_{VL}\rangle,\ C_D = \langle M_D,\ \Sigma_D,\ \Lambda_D\rangle\rangle_I \mapsto$$
$$C_{D,VL}(\mathrm{m},\ n,\ Z_1,\ Z_2,\ \ \ldots,\ Z_{3*n}) = \langle M_{D,VL},\ \Sigma_{D,VL},\ \Lambda_{D,VL}\rangle,$$

where $M_D \supset M_A$ and supplemented by algorithms that perform operations on attributes and operation $\bullet(a,\ b,\ c)$, $\Sigma_D = \Sigma_A$, $\Lambda_D = \Lambda_A$.

4 Modeling of Lightning Activity in Thunderstorm Front

NASA and National Oceanic and Atmospheric Administration (NOAA) regularly publish images from the Geostationary Lightning Mapper (GLM) instrument onboard the GOES-16 satellite. This is the new step in weather-watching capability, because it might see lightning activity above the clouds [25].

Based on the above constructors, the program modeling lightning activity was developed.

Testing of the developed methods for modeling lightning flashes is performed by comparing with real data obtained from the satellite. Figure 6 represents: frames of satellite video image [26, 35] (first row), the same with the removal of background (relief and cloudiness, second row), with the flashes brought to the regular form (third row).

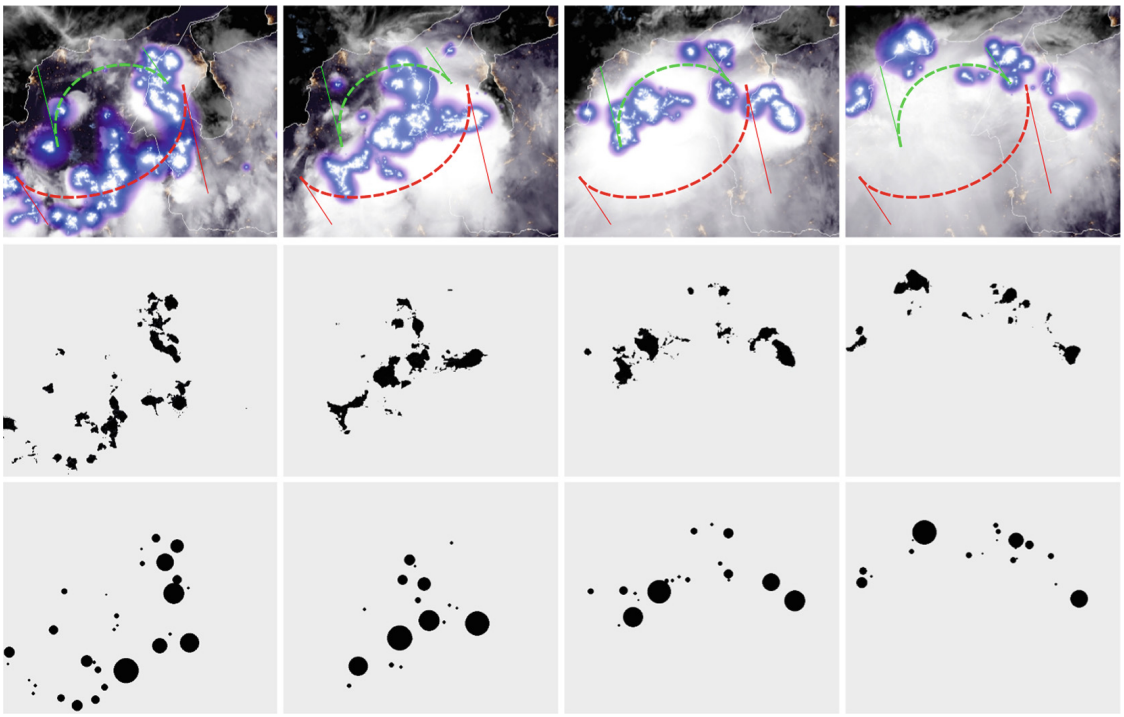

Fig. 6. Separate frames of the original video and flashes of lightning highlighted on them

A similar result was obtained as a result of modeling. The model of thunderstorm front is specified by the Bezier curve. For the largest part of the thunderstorm front (left) and a small part of the thunderstorm front (right) constructors were formed. Bezier curves are set for each of the parts that determine the initial and final position of the moving thunderstorm front.

The pair of constructors $C_{A,\,MS}$(f {f $\rightarrow$ ff, f $\rightarrow$ f+f-} 210)$_R \mapsto \Omega_1(C_{A,\,MS})$ and $C_{B,\,TS}(\Omega_1(C_{A,\,MS}),\ 10,\ 50,\ 10,\ 0,\ 210)_R \mapsto \Omega_1(C_{B,\,TS}) = u_{S,\,1}(t)$ forms the time series of the lightning discharge position along the curve (Fig. 7).

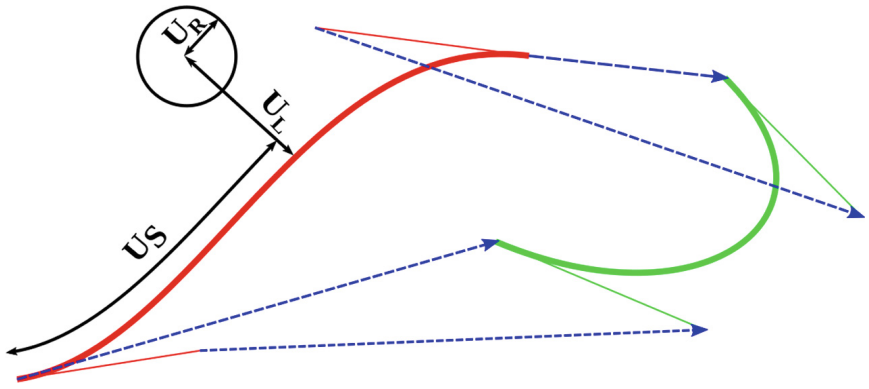

Fig. 7. Determination of the position and strength (radius of the circle) of the lightning flash

The second pair $C_{A,\,MS}$(f, {f $\rightarrow$ f-f+}, 210)$_R \mapsto \Omega_2(C_{A,\,MS})$ and $C_{B,\,TS}(\Omega_2(C_{A,\,MS}),\ 20,\ 50,\ 100,\ 0,\ 210)_R \mapsto \Omega_2(C_{B,\,TS}) == u_{L,\,1}(t)$ – distance from the curve. The third pair $C_{A,\,MS}$(f, {f $\rightarrow$ ff}, 210)$_R \mapsto \Omega_3(C_{A,\,MS})$ and

$C_{B,TS}(\Omega_3(C_{A,MS}), 6, 25, 10, 0, 210))_R \mapsto \Omega_3(C_{B,TS}) = u_{R,1}(t)$ – the value of the discharge $u_R(t) = \Omega_3(C_{A,TS})$. That's for the big (left) thunderstorm front. Constructed three time series (Fig. 8).

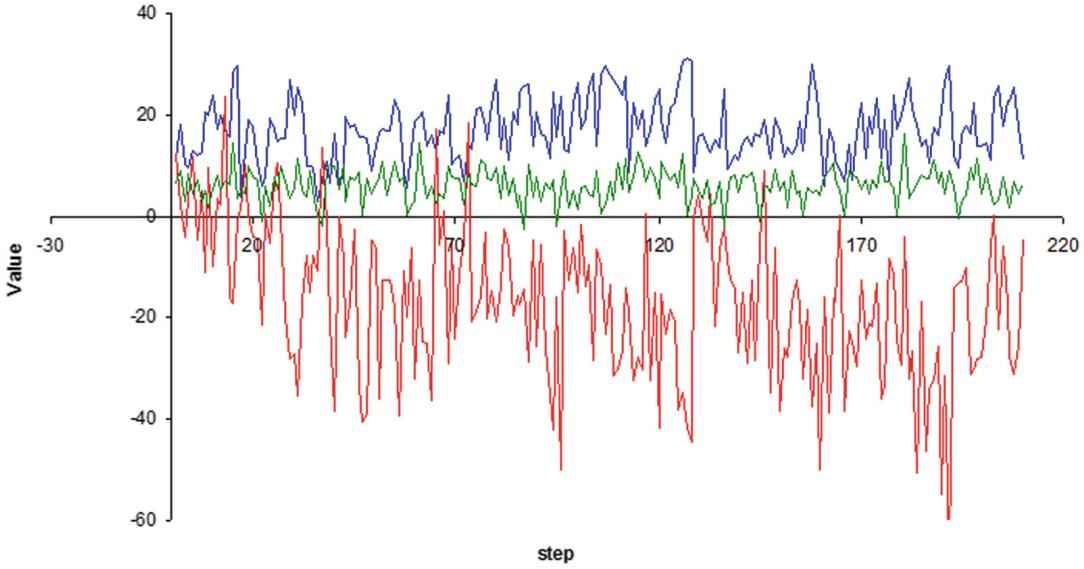

Fig. 8. Time series for the distance along (blue) and from (red) curve and lightning flash power (green) to model with the big curve

For the small (right) front the pair of constructors $C_{A,MS}$(fyz, {f → f+f-, y → -fffy, z → ---fffff}, 60)$_R \mapsto \Omega_4(C_{A,MS})$ and $C_{B,TS}(\Omega_4(C_{A,MS})$, 10, 25, 1, 0, 60))$_R \mapsto \Omega_4(C_{B,TS}) = u_{S,2}(t)$ forms the time series of the lightning discharge position along the curve. The second pair $C_{A,MS}$(f, {f → f+xf-, x → f-xf+}, 60)$_R \mapsto \Omega_5(C_{A,MS})$ and $C_{B,TS}(\Omega_5(C_{A,MS})$, 0, 12, 50, 0, 60))$_R \mapsto \Omega_5(C_{B,TS}) = u_{L,2}(t)$ – distance from the curve. The third pair $C_{A,MS}$(fz, {f → f+f-, z → --fffff}, 60)$_R \mapsto \Omega_6(C_{A,MS})$ and $C_{B,TS}(\Omega_6(C_{A,MS})$, 5, 25, 0.1, 0, 60))$_R \mapsto \Omega_6(C_{B,TS}) = u_{R,2}(t)$ – the value of the discharge $u_R(t) = \Omega_3(C_{A,TS})$.

The model presented above was clarified as follows. We have two moving Bezier curves: left and right as at Fig. 6 (only left).

According to the constructed time series constructor-assembler C_{VL} produced model video $C_{D,VL}(210, 2, u_{S,1}(t), u_{L,1}(t), u_{R,1}(t), u_{S,2}(t), u_{L,2}(t), u_{R,2}(t))$. On Fig. 9 shows all the flashes of lighting created by the constructor C_{VL}. All flashes sliced on five parts, demonstrating moving process from beginner to end by such sequence colors – yellow, aqua, magenta, red and blue.

Analysis of other satellite videos [26, 27] shows that the form of a thunderstorm cannot always be given by a single Bezier curve. Therefore, we provide the possibility of assigning several corresponding series of time series for them in the constructor-assembler C_{VL}.

These frames to some extent correspond to the frames from the satellite video (Fig. 1, column 3). Exact coincidence is not expected since the real and model processes are stochastic. However, all flashes of lightning from satellite video (Fig. 9, a) and video of the improved model (Fig. 9, b) correlation quite well.

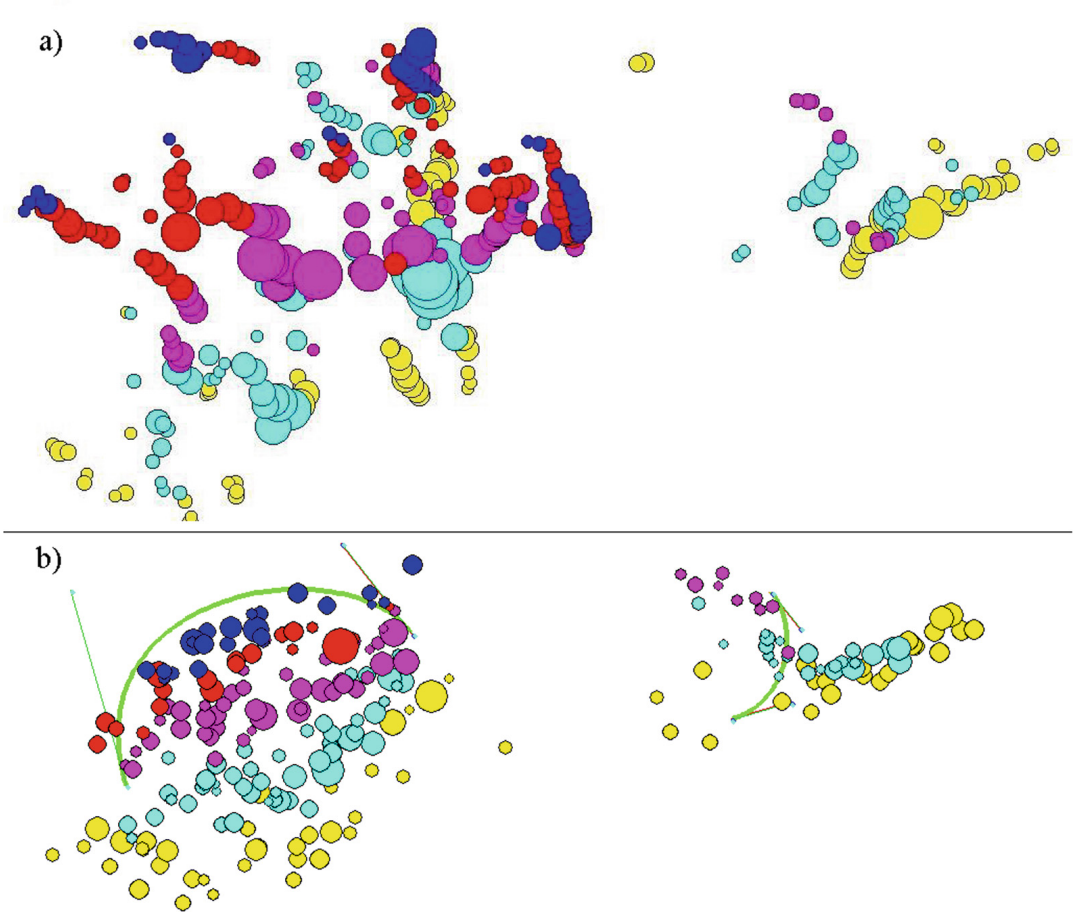

Fig. 9. Frames with all discharges of satellite and model lightning flashes

5 Conclusions

Investigation of modeling processes of the lightning flashes dynamic behavior in static background and moving thunderstorm fronts with very mobile clouds at satellite videos was performed. As a result, the method and software for lightning aureoles detection and then lightning at the weather satellites video images demonstrating significantly dynamic cloud cover were developed.

Video feature is its filtering when shooting in such a way that lightning flashes surrounded by a color contour. The contour's particular qualities of representation in Lab and LCH color models were taken into account while developing filters for identification of lightning contours and lightning themselves. As a result, two filters were proposed in order to solve the task. The series of experiments showed that quadratic one is the most effective. Despite this, the linear one has less computational complexity and so could be used in the cases when it is required to identify lightning in a short time.

A set of constructors were designed. Implementing parametric multi-character constructors allows forming fractal sequences of characters. Constructor-converter from the character string to time series creates fractal time series, which determine the location, magnitude and decay rate of lightning discharges. Modeling video images of lightning'

in the thunderstorm front are formed in accordance with the implementation of the constructor-assembler.

Software for lightning extraction from NASA satellites video, and also for realization constructive-synthesizing models was developed.

References

1. Sainte, V., Landrieu, L., Giordano, S., Chehata, N.: Satellite image time series classification with pixel-set encoders and temporal self-attention. arXiv preprint arXiv:1911.07757 (2019)
2. Fu, H., Shen, Y., Liu, J., Qian, J., Li, J.: Cloud detection for FY Meteorology satellite based on ensemble thresholds and random forests approach. Remote Sens. **11**(1), 44, 1–28 (2019). https://doi.org/10.3390/rs11010044
3. Peterson, M., Rudlosky, S., Zhang, D.: Thunderstorm cloud-type classification from space-based lightning images. Mon. Weather Rev. **148**(5), 1891–1898 (2020). https://doi.org/10.1175/MWR-D-19-0365.1
4. Jin, W., Gong, F., Tang, B., Wang, S.: Cloud types identification for meteorological satellite image using multiple sparse representation classifiers via decision fusion. IEEE Access **7**, 8675–8688 (2019). https://doi.org/10.1109/ACCESS.2018.2890295
5. Zheng, X., et al.: Detecting comma-shaped clouds for severe weather forecasting using shape and motion. IEEE Trans. Geosci. Remote Sens. **57**(6), 1–14 (2019). https://doi.org/10.1109/TGRS.2018.2887206
6. Shynkarenko, V., Lytvynenko, K., Chyhir, R., Nikitina, I.: Modeling of lightning flashes in thunderstorm front by constructive production of fractal time series. In: Shakhovska, N., Medykovskyy, M.O. (eds.) CSIT 2019. AISC, vol. 1080, pp. 173–185. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-33695-0_13
7. Shynkarenko, V., Lytvynenko, K., Chyhir, R., Sansiieva, I.: Constructive modeling of lightning activity in thunderstorm front. In: IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), pp. 92–95 (2019). https://doi.org/10.1109/STC-CSIT.2019.8929754
8. Sátoria, G., Williams, E., Lempergeraams, I.: Variability of global lightning activity on the ENSO time scale. Atmos. Res. **91**(2–4), 500–507 (2009). https://doi.org/10.1016/j.atmosres.2008.06.014
9. Devendraa, S., RameshKumara, P., Kulkarnia, M.N., Singhb, R.P., Singhc, A.K.: Lightning, convective rain and solar activity—over the South/Southeast Asia. Atmos. Res. **120–121**, 99–111 (2013). https://doi.org/10.1016/j.atmosres.2012.07.026
10. Galanaki, E., Kotroni, V., Lagouvardos, K., Argiriou, A.: A ten-year analysis of cloud-to-ground lightning activity over the Eastern Mediterranean region. Atmos. Res. **166**, 213–222 (2015). https://doi.org/10.1016/j.atmosres.2015.07.008
11. Ahrens, M.: Lightning fires and lightning strikes. In: National Fire Protection Association, Quincy (2013)
12. Fuchs, B.R., Bruning, E.C., Rutledge, S.A., Carey, L.D., Krehbiel, P.R., Rison, W.: Climatological analyses of LMA data with an open-source lightning flashclustering algorithm. J. Geophys. Res.: Atmos. **121**(14), 8625–8648 (2016). https://doi.org/10.1002/2015JD024663
13. Guichard, F., Couvreux, F.: A short review of numerical cloud-resolving models. Tellus A: Dyn. Meteorol. Oceanogr. **69**(1), 1–36 (2017). https://doi.org/10.1080/16000870.2017.1373578
14. Karagiannidis, A., Lagouvardos, K., Lykoudis, S., Kotroni, V., Giannaros, T., Betz, H.D.: Modeling lightning density using cloud top parameters. Atmos. Res. **222**, 163–171 (2019). https://doi.org/10.1016/j.atmosres.2019.02.013

15. First Images from GOES-16 Lightning Mapper. https://www.americaspace.com/2017/03/07/goes-16-satellite-returns-first-lightning-mapping-images-like-never-seen-before. Accessed 15 July 2020
16. Radhika, K., Varadarajan, S.: A neural network based classification of satellite images for change detection applications. Cogent. Eng. **5**, 1–9 (2018). https://doi.org/10.1080/23311916.2018.1484587
17. Chen, G., Liu, Y., Tian, Y. Tian, H.: Use of VIIRS DNB satellite images to detect nighttime fishing vessel lights in yellow sea. In: Proceedings of the 3rd International Conference on Computer Science and Application Engineering 2019, pp. 1–5 (2019). https://doi.org/10.1145/3331453.3361661
18. Elvidge, D., Zhizhin, M., Baugh, K., Hsu, F.: Automatic boat identification system for VIIRS low light imaging data. Remote Sens. **7**(3), 3020–3036 (2015). https://doi.org/10.3390/rs70303020
19. Nocturnal thunderstorms near Lake Maracaibo are known for their "Catatumbo Lightning". http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes16/20180815000000&number_of_images_to_display=100&loop_speed_ms=15. Accessed 15 July 2020
20. The next storm to hit the U.S. West Coast is already producing lightning. http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes-16/201901/16000000&number_of_images_to_display=120&loop_speed_ms=100. Accessed 15 July 2020
21. Thanh, D., Erkan, U., Prasath, V., Kumar, V., Hien, N.: A skin lesion segmentation method for dermoscopic images based on adaptive thresholding with normalization of color models. In: 2019 6th International Conference on Electrical and Electronics Engineering (ICEEE) 2019, pp. 116–120 (2019). https://doi.org/10.1109/ICEEE2019.2019.00030 AQ2
22. Patil, J., Chaudhari, S.: Screening of damage regions in retinopathy using segmentation-color space selection. Int. J. Multimedia Image Process. (IJMIP) **7**(1), 362–365 (2017). https://doi.org/10.20533/ijmip.2042.4647.2017.0044
23. Thanh, L.T., Thanh, D.N.H.: An adaptive local thresholding roads segmentation method for satellite aerial images with normalized HSV and lab color models. In: Solanki, V.K., Hoang, M.K., Lu, Z.J., Pattnaik, P.K. (eds.) Intelligent Computing in Engineering. AISC, vol. 1125, pp. 865–872. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-2780-7_92
24. Sayed, M., Sammani, F., Albashier, M.: An accurate method to calculate the color difference in a single image. In: International Conference on Robotics, Automation and Sciences (ICORAS) 2017, pp. 1–3 (2017). https://doi.org/10.1109/ICORAS.2017.8308079
25. Kolkur, S., Kalbande, D., Shimpi, P., Bapat, C., Jatakia, J.: Human skin detection using RGB, HSV and YCbCr color models. arXiv preprint arXiv:1708.02694 (2017). https://doi.org/10.2991/iccasp-16.2017.51
26. Mporas, I., Perikos, I., Paraskevas, M.: Color models for skin lesion classification from dermatoscopic images. In: Hatzilygeroudis, I., Perikos, I., Grivokostopoulou, F. (eds.) Advances in Integrations of Intelligent Methods. SIST, vol. 170, pp. 85–98. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-1918-5_5
27. Chuang, C., Chen, J., Lin, H., Huang, S., Wen, J.: Soil moisture estimation using multiple color spaces in digital image analysis. In: Geophysical Research Abstracts, vol. 21, EGU2019-11542 (2019)
28. Lightning, surface outflow boundaries and transverse bands in the anvil cirrus. http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes-16/20180614000000&number_of_images_to_display=100&loop_speed_ms=100. Accessed 15 July 2020
29. Fires, dust storms, severe weather, blizzard conditions and thundersnow: this storm has it all! http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_

day/goes-16/20180413000000&number_of_images_to_display=100&loop_speed_ms=100. Accessed 15 July 2020
30. Lightning packed squall line developing over Texas. http://rammb.cira.colostate.edu/ramsdis/online/loop.asp?data_folder=loop_of_the_day/goes-16/20200526000000&number_of_images_to_display=260&loop_speed_ms=80. Accessed 15 July 2020
31. Shynkarenko, V.I., Ilman, V.M.: Constructive-synthesizing structures and their grammatical interpretations. I. Generalized formal constructive-synthesizing structure. Cybern. Syst. Anal. **50**(5), 655–662 (2014). https://doi.org/10.1007/s10559-014-9655-z
32. Shynkarenko, V.I., Ilman, V.M.: Constructive-synthesizing structures and their grammatical interpretations. II. Refining transformations. Cybern. Syst. Anal. **50**(6), 829–841 (2014). https://doi.org/10.1007/s10559-014-9674-9
33. Shynkarenko, V.I., Ilman, V.M., Skalozub, V.V.: Structural models of algorithms in problems of applied programming. I. Formal algorithmic structures. Cybern. Syst. Anal. **45**(3), 329–339 (2009). https://doi.org/10.1007/s10559-009-9118-0
34. Skalozub, V., Ilman, V., Shynkarenko, V.: Development of ontological support of constructive-synthesizing modeling of information systems. Eastern-Eur. J. Enterp. Technol. **6**(4–90), 58–69 (2017). https://doi.org/10.15587/1729-4061.2017.119497
35. Shynkarenko, V., Nikitina, I.: Lightning recognition on the filtered videos in the dynamic clouds. In: IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies (2020)

УДК 510.25+004.9+519.8
№ держреєстрації 0118U004215

Міністерство освіти і науки України
Дніпровський національний університет
залізничного транспорту імені академіка В. Лазаряна
49010, м. Дніпро, вул. Лазаряна, 2;
тел. +38(056) – 776 – 59 – 47, факс . +38(056) – 562 – 47 – 18 – 66

ЗАТВЕРДЖУЮ
Ректор університету
д.т.н, проф. Пшінько О.М.

ЗВІТ З НАУКОВО-ДОСЛІДНОЇ РОБОТИ «КОНСТРУКТИВНО-ПРОДУКЦІЙНЕ МОДЕЛЮВАННЯ ФРАКТАЛІВ»

Керівник: доцент кафедри «Комп'ютерні інформаційні технології», к. т. н. К.В. Литвиненко

Керівник науково-дослідної роботи
«Конструктивно-продукційне моделювання фракталів»
доцент кафедри «Комп'ютерні інформаційні технології» (КІТ),
к.т.н., доц. каф. КІТ Литвиненко К. В.

Відповідальний виконавець
д.т.н., проф., зав. каф. КІТ Шинкаренко В.І.
Виконавці: ас. каф. КІТ Васецька Т. М., студенти Чигирь Р.Р., Жадан А.А.,
Сансієва І.М.
Виконання розділів:

№	Розділ	Виконавці
1	Конструктивні фрактали, та методи їх моделювання.	к.т.н. Литвиненко К.В. д.т.н. Шинкаренко В.І. ас. каф. Васецька Т. М.
2	Моделювання геометричних фракталів на основі L-систем.	д.т.н. Шинкаренко В.І. к.т.н. Литвиненко К.В. ст. Чигирь Р.Р., ст. Жадан А.А. ст. Сансієва І.М.
3	Моделювання часових рядів на основі L-систем. Множинна інтерпретація для конструктивно-продукційного моделювання фракталів. Застосування конструктивно-продукційного підходу для моделювання активності блискавок грозового фронту.	д.т.н. Шинкаренко В.І. к.т.н. Литвиненко К.В. ас. каф. Васецька Т. М. ст. Чигирь Р.Р., ст. Жадан А.А. ст. Сансієва І.М.