Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня ___магістра__
(бакалавра, спеціаліста, магістра)

студента _____Нікуліна Максима Сергійовича_____
(ПІБ)
академічної групи _____123м-19-1_____
(шифр)
спеціальності _____123 «Комп'ютерна інженерія»_____
(код і назва спеціальності)
за освітньо-професійною програмою ___«Комп'ютерна інженерія»___
(офіційна назва)
на тему «Комп'ютерна система контролю завантаження стрічкових конвеєрів вугільної шахти з використання технологій обробки зображень»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Цвіркун Л.І.			
розділів:				
теоретичний розділ	проф. Цвіркун Л.І.			
синтез системи	доц. Ткаченко С.М.			
розроблення програмного забезпечення	ас. Бешта Л.В.			
експериментальний розділ	доц. Ткаченко С.М.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2020

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ __Гнатушенко В.В._
(підпис) (прізвище, ініціали)

«___»_____ 2020 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня _____магістр _____
(бакалавра, спеціаліста, магістра)

студента _____Нікуліна М.С.____ **академічної групи** ____123м-19-1____
(прізвище та ініціали) (шифр)

спеціальності _____123 «Комп'ютерна інженерія»_____

за освітньою-професійною програмою _123 «Комп'ютерна інженерія»__
(офіційна назва)

на тему «Комп'ютерна система контролю завантаження стрічкових конвеєрів вугільної шахти з використання технологій обробки зображень»,

затверджену наказом ректора НТУ «Дніпровська політехніка» від _____ №_____

Розділ	Зміст	Термін виконання
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел сформулювати наукове завдання, конкретизувати предмет та мету досліджень	21.09.2020
Теоретичний	Обґрунтувати теоретичну базу розв'язання наукового завдання, якому присвячено роботу	30.10.2020
Синтез системи	Розробити структурну і функціональну модель, схему алгоритму комп'ютерної системи	12.11.2020
Розроблення програмного забезпечення	Розробити програмне забезпечення на основі результатів синтезу системи	26.11.2020
Експериментальний розділ	Проведення і обробка результатів експериментів	06.12.2020

Завдання видано _____ _проф. Цвіркун Л. І._
(підпис керівника) (прізвище, ініціали)

Дата видачі _07 вересня_2020 р._

Дата подання до екзаменаційної комісії ____10.12.2020 р.____

Прийнято до виконання _____ _Нікулін М.С._
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 80 стор., 27 рис., 1 табл., 2 додатки, 7 джерел.

Об’єкт дослідження: комп’ютерна система контролю завантаження стрічки конвеєра вугільної шахти, побудована на базі технології комп’ютерного зору.

Мета: аналіз і обґрунтування доцільності використання окремих алгоритмів обробки цифрових зображень у комп’ютерній системі контролю, розробка програмного забезпечення системи.

У вступі відображена актуальність проблеми, мета, завдання та ідея дослідження, наукові положення та результати.

У розділі «Стан питання та постановка завдань дослідження» викладаються загальні відомості про галузь, комплекс шахти та шахтний транспорт, розглядаються існуючі реалізації систем контролю стрічкового конвеєра.

У теоретичному розділі досліджувана комп’ютерна система розглядається як окремий компонент у складі інформаційного комплексу шахти. Визначено принцип функціонування та внутрішню структуру досліджуваної системи. Надані відомості про алгоритми з обробки зображень, які пропонуються до застосування.

У розділі «Синтез системи контролю» згідно вимогам було обрано апаратні складові системи. Розроблено структурну та функціональну схему модулю вводу відеоінформації.

У розділі «Розробка програмного забезпечення системи» сформульовані вимоги до призначення і умов застосування. Обґрунтовано структуру, алгоритм і методи організації даних. Розроблено програму і наведено її детальний опис.

У експериментальному розділі поставлено задачу і описано методику експерименту з аналізу і оцінки результатів застосування алгоритмів обробки зображення. Проведено експеримент та проаналізовані його результати. Перевірено точність результатів програми контролю завантаження стрічки конвеєра.

ВУГІЛЬНА ШАХТА, КОНВЕЄР, КОМП’ЮТЕРНИЙ ЗІР, OPENCV

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ARM (Advanced RISK Machine) – архітектура процесору з мінімальним набором команд;

BVS (Belt Volume Scanner) – стара назва для CVS;

CVS (Conveyor Volume Scanner) – сканер об’єму вантажу стрічкового конвеєра, клас комерційних продуктів;

GPIO (General-purpose input/output) – електричний інтерфейс вводу/виводу загального призначення

IP, IP-протокол (Internet Protocol) – найпоширеніший протокол мережевого рівня

IP-камера – цифрова відеокамера, що передає відеопоток в цифровому форматі комп’ютерною мережею, що використовує IP протокол;

LIDAR, лідар (Light Identification, Detection and Ranging) – технологія виявлення та визначення дальності за допомогою світла;

RPC (Remote Procedure Call) – виклик віддаленої процедури – протокол, що дозволяє комп’ютерним програмам обмінюватись даними та викликати функції одна іншої з використанням комп’ютерних мереж;

SBC (Single-Board Computer) – клас повнофункціональних портативних комп’ютерів;

SMTP (Simple Mail Transfer Protocol) – простий протокол пересилання пошти;

SQL (Structured Query Language) – мова структурованих запитів;

АСК – автоматизована система керування;

ООП – об’єктно-орієнтоване програмування, одна з парадигм програмування

ПЗ – програмне забезпечення;

СКБД – система керування базами даних;

ВСТУП

Вугільна промисловість є однією з ключових галузей української економіки. Близько 2/3 видобутого вугілля використовується в електро- та теплоенергетиці, 1/6 – у чорній металургії, та ще 1/6 перетворюється (брикетування, коксохімія). У той же час, приблизно третина всієї споживаної електроенергії в Україні виробляється на теплових електростанціях, з яких 80% використовують вугільну продукцію. [1]

Але основні виробничі фонди шахтних підприємств є застарілими та зношеними. Більшість державних вугледобувних підприємств, значну частку яких становлять дрібні малопотужні шахти зі складними гірничо-геологічними умовами, працюють неефективно та перебувають на державній дотації і утриманні. [2]

Транспортування видобутого вугілля на поверхню шахти є одним з основних етапів виробничого циклу. Найефективнішим шахтним транспортом вважається стрічковий конвеєр через простоту конструкції та низькі витрати на обслуговування. Рух конвеєра забезпечується електроприводом, потужність якого може сягати декількох мегават.

Дослідження енергоефективності шахтного конвеєрного транспорту показують, що застосування регульованого привода для зміни швидкості стрічки може скоротити витрати на електроенергію для всієї лінії. В залежності від технічних та технологічних умов, скорочення витрат може становити до 20%. [3]

Досягнення подібного економічного ефекту є неможливим без впровадження автоматизованої системи регулювання швидкості стрічки конвеєра. Керуючий вплив автоматизованої системи формується відносно вхідного сигналу, у ролі якого виступає поточна завантаженість стрічки. Джерелом такого сигналу має бути комп’ютерна система контролю завантаження стрічки.

Мета дослідження – аналіз і обґрунтування доцільності використання окремих алгоритмів обробки цифрових зображень у комп’ютерній системі контролю, розробка програмного забезпечення системи.

Завдання дослідження: дослідити існуючі алгоритми обробки цифрових зображень, які можуть бути застосовані у комп'ютерній системі контролю завантаження стрічки конвеєра; розробити програму, що реалізує досліджувані алгоритми для обчислення величини, що характеризує завантаження стрічки конвеєра.

Об'єкт дослідження – комп'ютерна система контролю завантаження стрічки конвеєра, побудована на базі технології комп'ютерного зору.

Предмет дослідження – алгоритми обробки цифрових зображень, що можуть бути застосовані у комп'ютерній системі.

Методи дослідження. Для досягнення поставленої мети використовувались методи теорії комп'ютерного зору, теорії обробки сигналів, теорії інформації.

Ідея роботи – розробка альтернативної комп'ютерної системи контролю завантаження стрічкових конвеєрів, що не залежить від специфічного або дорогого обладнання, не потребує значних монтажних робіт, та здатна працювати за існуючих умов місця розташування конвеєра.

Наукові положення:

1. Встановлено, що застосування алгоритмів комп'ютерного зору дозволить спростити структуру та зменшити витрати на розробку і обслуговування комп'ютерної системи контролю завантаження стрічкових конвеєрів.

Наукові результати:

1. Обґрунтована доцільність впровадження окремих алгоритмів комп'ютерного зору для підготовки відеозображення для подальшого розпізнавання контурів вантажопотоку на стрічці конвеєра.
2. Охарактеризовані етапи застосування алгоритмів комп'ютерного зору, обробки зображень та геометричних методів для виявлення ширини потоку транспортованого матеріалу відносно ширини стрічки конвеєра.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз об’єкту дослідження

1.1.1 Структура підземного комплексу шахти

Вугільні пласти формуються на певних етапах тектонічного розвитку континентів – при переважаючому значенні загального занурювання, на фоні яких проявляються коливальні рухи підлеглих амплітуд. Пласт – найбільш розповсюджена форма покладу осадових родовищ рудних покладів і нерудних корисних копалин. Пластом називається тіло, складене однорідними мінеральними утвореннями і обмежене двома практично паралельними поверхнями – поверхнями напластування. Будова вугільних пластів змінюється від простого (без породних прошарків) і помірно складного (з одним або кількома породними прошарками) до дуже складного (перешарування численних вугільних і породних прошарків).[4]

Шахта – це гірничопромислове підприємство, яке здійснює видобуток корисних копалин підземним способом. В поняття «шахта» входять наземні споруди та сукупність гірничих виробок, призначених для розробки родовища в межах шахтного поля.

Гірнича виробка – це порожнина, зроблена в надрах землі або на поверхні. Підземні гірничі виробки, незалежно від наявності безпосереднього виходу на поверхню, мають замкнутий контур поперечного перетину.

Залежно від призначення, розрізняють гірничі виробки розвідувальні та експлуатаційні. Перші використовують для пошуків і розвідки родовищ корисних копалин, другі – для розробки родовищ, тобто для виймання корисних копалин із надр.

Ствол – це вертикальна капітальна гірнича виробка, що має безпосередній вихід на земну поверхню і призначена для обслуговування підземних гірничих робіт. По шахтних стволах підіймають корисні копалини, породу, матеріали, устаткування, людей та рухається повітряна течія.

Залежно від основного призначення, шахтні стволи поділяють на головні й допоміжні. Головний ствол служить для підйому на поверхню корисних копалин. Допоміжні стволи відповідно до їх функцій підрозділяються на вантажно-людські – для спуску та підйому людей, матеріалів, устаткування і вентиляційні – для провітрювання і т. ін.

Лава, смуга – очисні горизонтальні або похилі гірничі виробки великої довжини, в яких безпосередньо добувається вугілля.

Штрек – горизонтальна гірнича виробка, яка не має безпосереднього виходу на земну поверхню, споруджена по простяганню пласта. За своїм призначенням штреки можуть бути транспортними, вентиляційними, корінними, головними, бутовими та ін.

Приствольним двором називається комплекс гірничих виробок, розміщених навколо стволів, з'єднуючих їх з головними виробками горизонту і призначених для обслуговування гірничих робіт на горизонті. У ньому виконуються роботи по прийому вугілля з дільниць та підйому вугілля та породи на поверхню, прийому і підйому людей, матеріалів і обладнання, водовідливу, вентиляції та ін. Тут розміщуються камери різного технологічного призначення.

По типу стволів, що розкривають шахтне поле, розрізняють приствольні двори при вертикальних стволах і приствольні двори при похилих стволах. В залежності від кількості вертикальних стволів, розміщених в приствольному дворі, розрізняють двори при одному стволі, при двох і при трьох стволах.

В залежності від типу підйомних посудин в стволах, бувають скіпові, клітьові і скіпово-клітьові приствольні двори.

В залежності від виду транспорту в приствольному дворі, останні підрозділяються на три типа: приствольні двори з локомотивним, стрічковим і гідравлічним транспортом.

1.1.2 Транспортні процеси та види шахтного транспорту

Вантажопотоки вугілля, породи, матеріалів, різного обладнання, а також перевезення людей в гірничих виробках забезпечує підземний транспорт. [4]

Розрізняють безперервний транспорт, коли вантажі переміщуються безперервно, та транспорт перериваний, коли навантаження і розвантаження можливі тільки під час повної зупинки транспортних засобів.

До першої ланки відносять конвеєрний транспорт, до другої рейковий та підйом по вертикальних стволах в скіпах.

Стрічкові конвеєри застосовують в горизонтальних та похилих виробках з похилом до 18° для транспортування вантажів, а в деяких випадках і для перевезення людей. Конвеєри для похилих виробок обладнуються гальмами і вловлювачами стрічки.

Приклад схеми конвеєрного транспорту показаний на рисунку Рисунок 1.1.

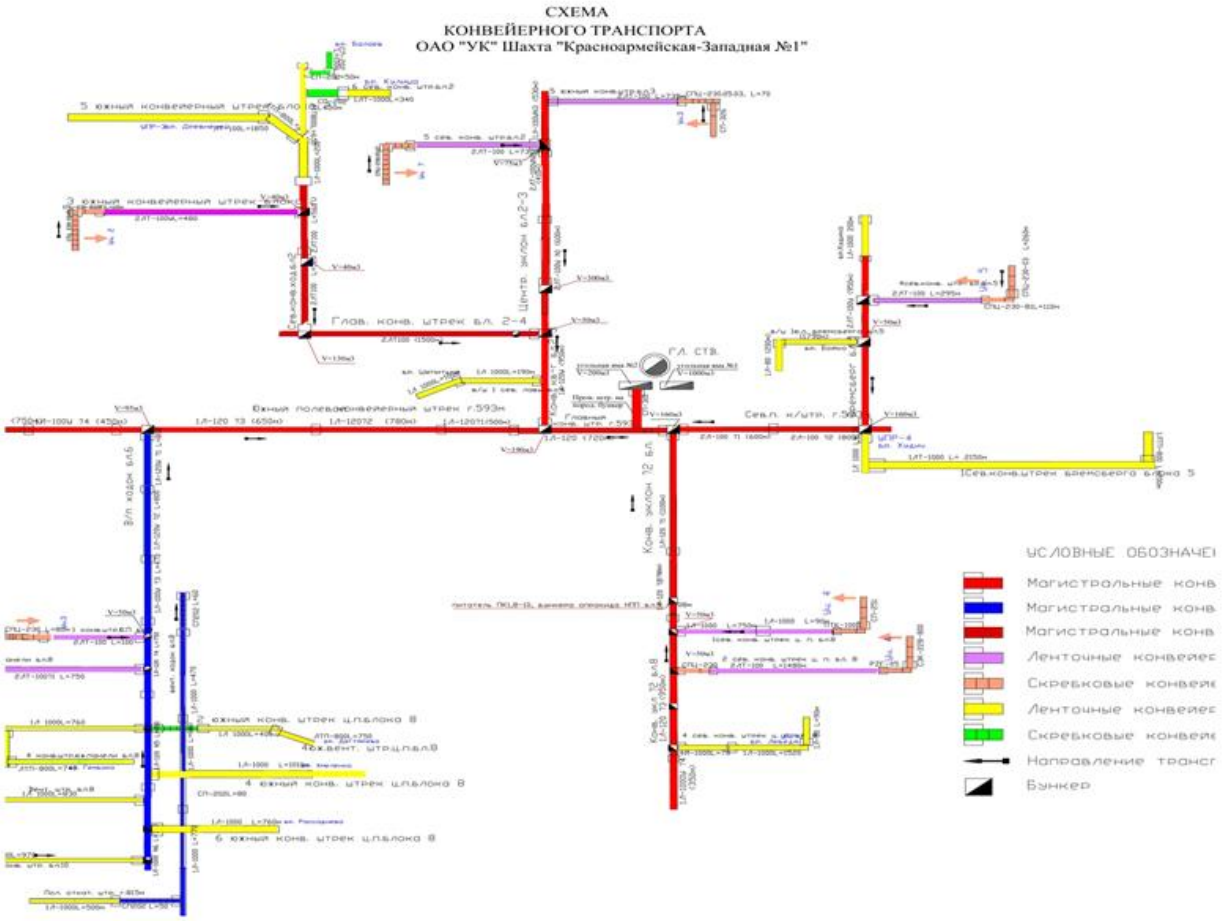

Рисунок 1.1 – Схема конвеєрного транспорту

Шахтний підйом виконує вантажно-транспортний зв'язок підземного гірничого хазяйства з поверхнею шахти через вертикальні або похилі стволи.

Кожна шахта, як мінімум, має дві підйомні установки: головний підйом – для видачі корисних копалин і породи, допоміжний підйом – для всіх останніх цілей, в тому числі підйому і спуску людей.

1.1.3 Регулювання швидкості стрічкових конвеєрів

Для вирішення проблеми зниження витрат електроенергії в останні кілька років на вугільних шахтах України застосовують частотні перетворювачі енергії для регулювання швидкості руху стрічок магістрального конвеєрного транспорту, що забезпечує більш повне їх заповнення в умовах змінного вантажопотоку і зниження питомих витрат електроенергії на транспортування вугілля. [5]

Швидкість руху конвеєрної стрічки встановлюється системою регулювання пропорційно вантажопотоку, проте в періоди відсутності вугілля конвеєрна стрічка рухається з мінімальною швидкістю без зупинки, що обумовлено складностями пуску з вантажем і вимогами постійної готовності до прийняття вантажу.

За узагальненими даними потенціал зниження витрат електроенергії на конвеєрному транспорті становить 40-50%. Розрахунки, виконані у роботі [6], показують, що впровадження перетворювачів частоти і систем регулювання швидкості руху стрічки дозволяє знизити витрату електроенергії на 28–35 %. Умови режимів роботи конвеєрного транспорту, описані у зазначеній роботі, дозволяють встановити, що регулювання швидкості руху стрічки окремого конвеєра змінює статистичні характеристики вантажопотоку після нього і призводить до зміни швидкості руху стрічки наступних конвеєрів, а також до зміни величини їх електроспоживання.

1.2 Огляд існуючих рішень

1.2.1 Сканери об’єму транспортованого матеріалу на основі лазера

Сучасний ринок пропонує цілу низку рішень для вимірювання об’єму транспортованого матеріалу, що мають загальну назву CVS (Conveyor Volume Scanner), або BVS (Belt Volume Scanner) – сканер об’єму вантажу стрічкового конвеєра. У США та країнах Європи підприємства у якості продукту надають повний комплект обладнання в закритому корпусі, як зображено на рисунку Рисунок 1.2, разом із пропрієтарним (закритим) програмним забезпеченням для ПК (АРМ оператора).

Рисунок 1.2 – Встановлений сканер об’єму вантажу конвеєрної стрічки

Вони базуються на технології лідару, що використовує явище відбиття та розсіювання світла для визначення дистанції до неї.

Сканери об’єму зазвичай використовують один або декілька лінійних лазерних випромінювачів в парі зі світловими сенсорами, що розташовані перпендикулярно до стрічки конвеєра в декількох десятках сантиметрів від неї, як зображено на рисунку Рисунок 1.3.

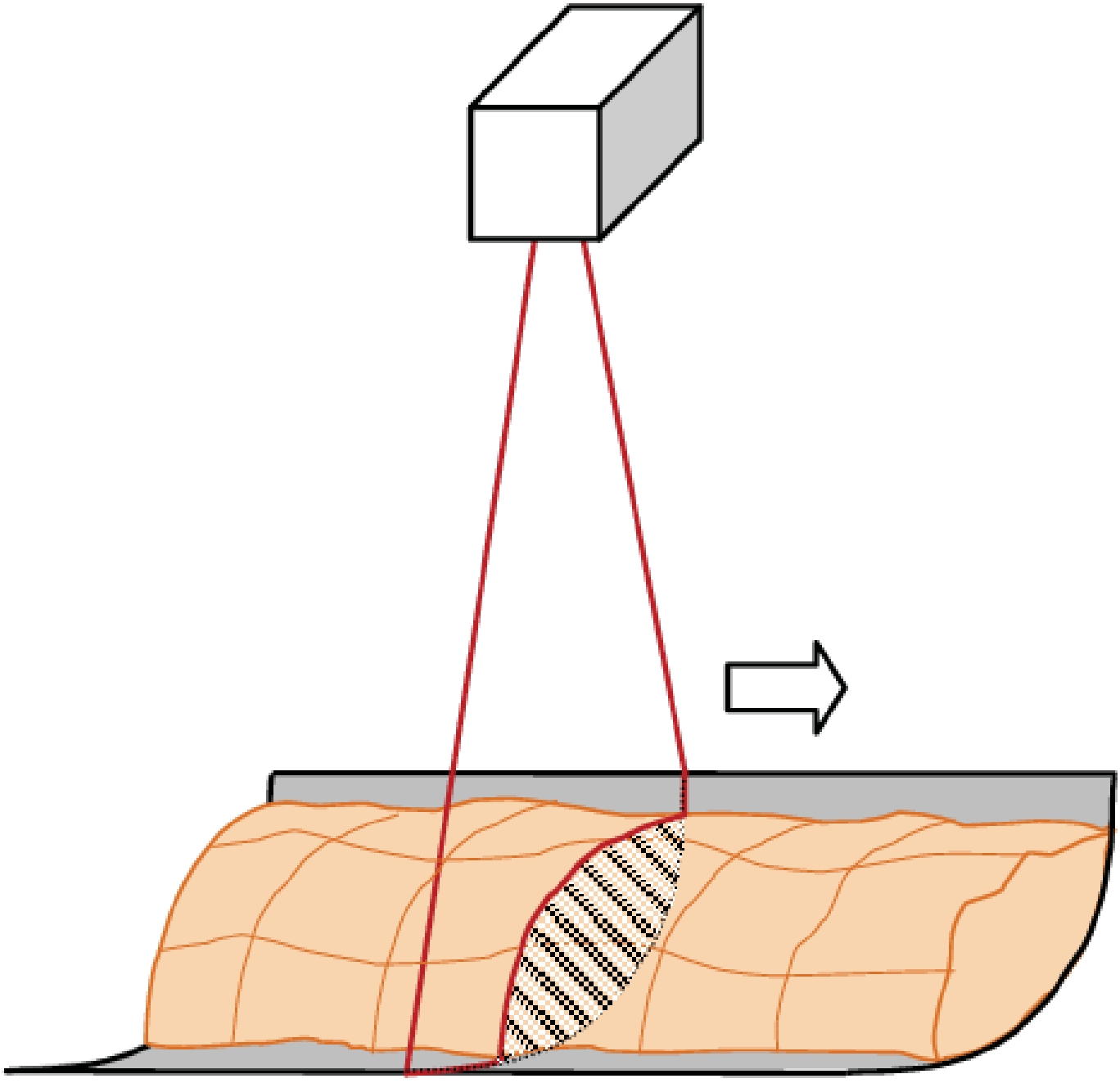

Рисунок 1.3 - Модель, що демонструє роботу сканера об’єму вантажу на стрічці

До переваг такого типу сканерів відносяться простота калібрування, налаштування програми, технічна підтримка з боку виробника.

Недоліками подібних систем можна вважати необхідність встановлення металевої рамки для закріплення, складність залучення до автоматизованої системи керування, якщо подібна функція не передбачена в ПЗ сканера. До того ж, відсутня можливість самостійного ремонту або заміни модулів, що вийшли з ладу, модифікації ПЗ у разі потреби.

Виробник не вказує ціну на веб-сторінці продукту, але виходячи з собівартості компонент системи, вартості розробки комплексного програмного забезпечення та додаткових послуг, вона орієнтовно може становити від декількох сотень до декількох тисяч доларів.

1.2.2 Комп’ютерні системи для конвеєра, що використовують технології комп’ютерного зору

У роботі [7] представлена система оцінки розподілу розмірів та об’ємів серед матеріалу, що транспортується стрічковим конвеєром із застосуванням алгоритмів машинного зору.

Як і комерційні варіанти, описані в попередньому розділі, вона використовує лінійне джерело світла, але, на відміну від лідару, вона не замірює час відбиття світла, а використовує лазерну тріангуляцію: джерело світла розташоване під кутом, а камера – перпендикулярно до плоскості стрічки. Об’єкти на стрічці будуть змінювати рівень видимої лінії, як зображено на рисунку Рисунок 1.4.

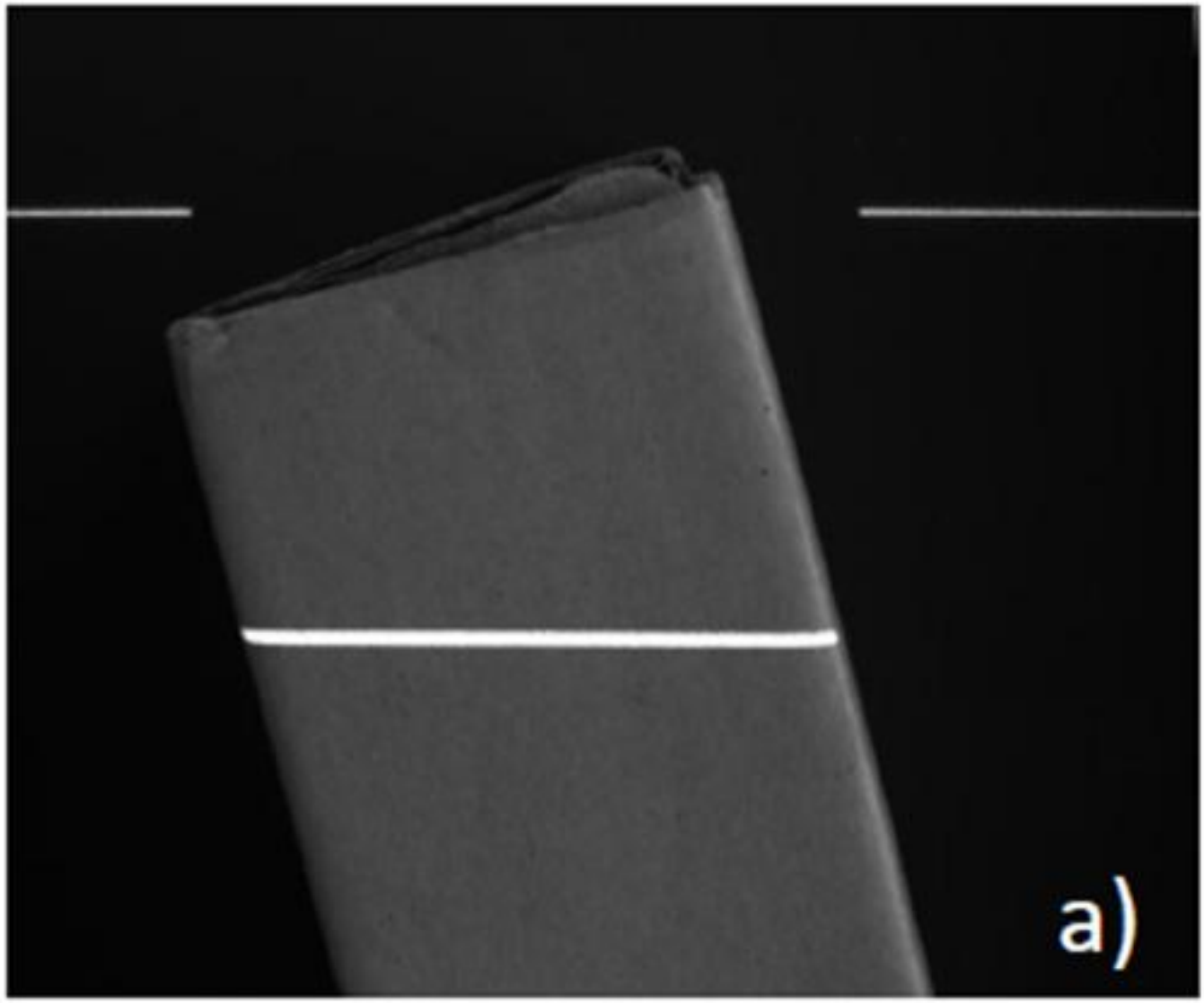

Рисунок 1.4 – Ілюстрація принципу лазерної тріангуляції

Недоліками такого підходу є залежність від лінійного джерела світлу, що не розсіюється, та ускладнення алгоритмів при використанні системи для жолобчастого стрічкового конвеєру – на зображенні камери буде представлена крива замість лінії.

Інша комп’ютерна система, описана у роботі [8], працює за таким само принципом, та має схожу структуру (зображена на рисунку Рисунок 1.5) , але призначена для виявлення пошкоджень стрічки. На відміну від попередньої, в даній системі використана лінійна камера, а у якості вхідних даних – значення яскравостей точок, а не величини інтервалів між розривами лінії світла.

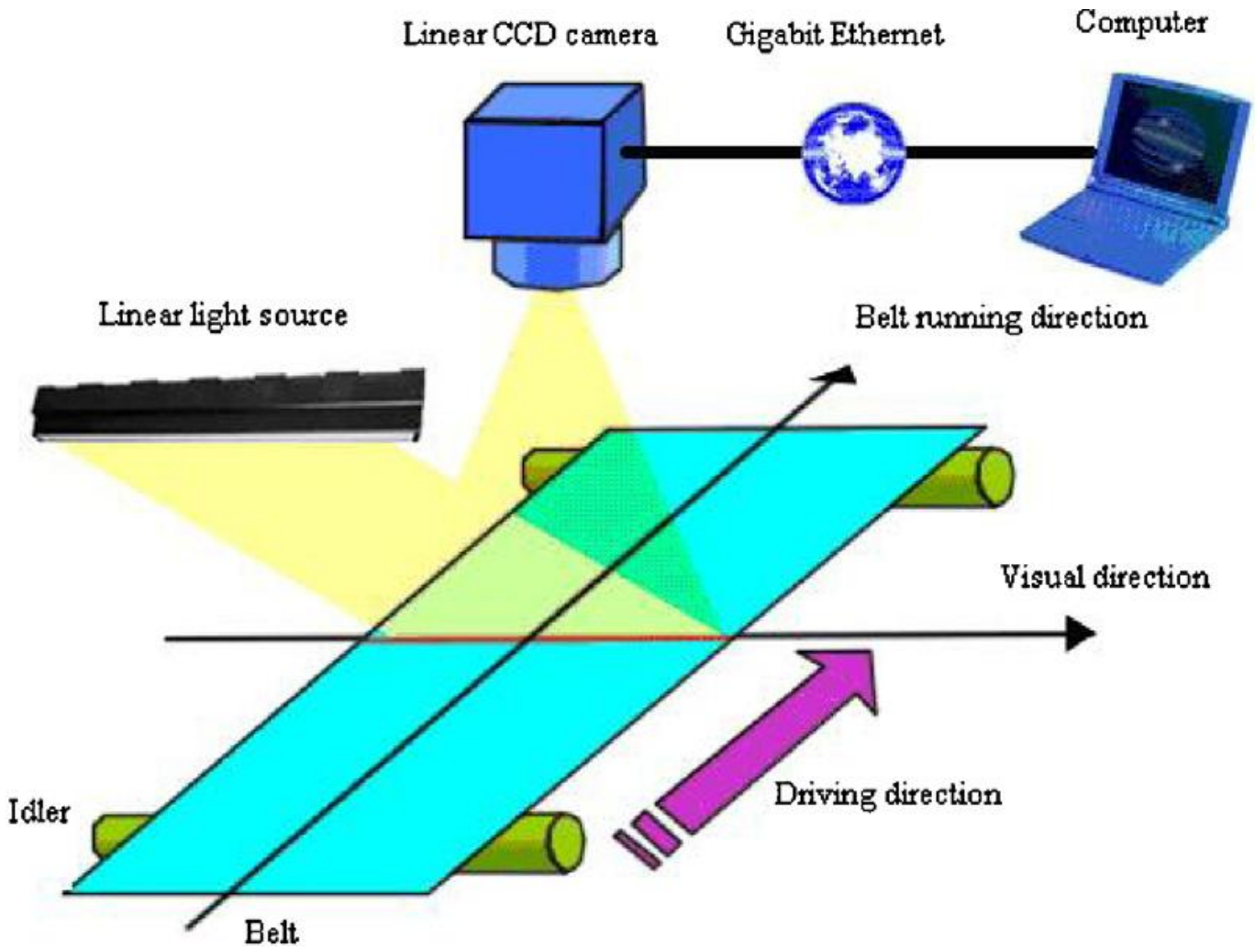

Рисунок 1.5 – Модель системи, що використовує лазерну тріангуляцію та алгоритми комп’ютерного зору

Пошкодження стрічки цією системою визначаються за відмінністю у величині яскравості порівняно із сусідніми ділянками, тобто за відмінністю кольору. Фактично, камера розташовується знизу конвеєрної стрічки, як буде показано на рисунку Рисунок 1.6.

Автори роботи зробили спробу адаптувати підхід для застосування з жолобчастими конвеєрами, як проілюстровано на рисунку Рисунок 1.6, розташувавши джерела світла уздовж вигину стрічки. Такий підхід допустимий лише через специфіку задачі.

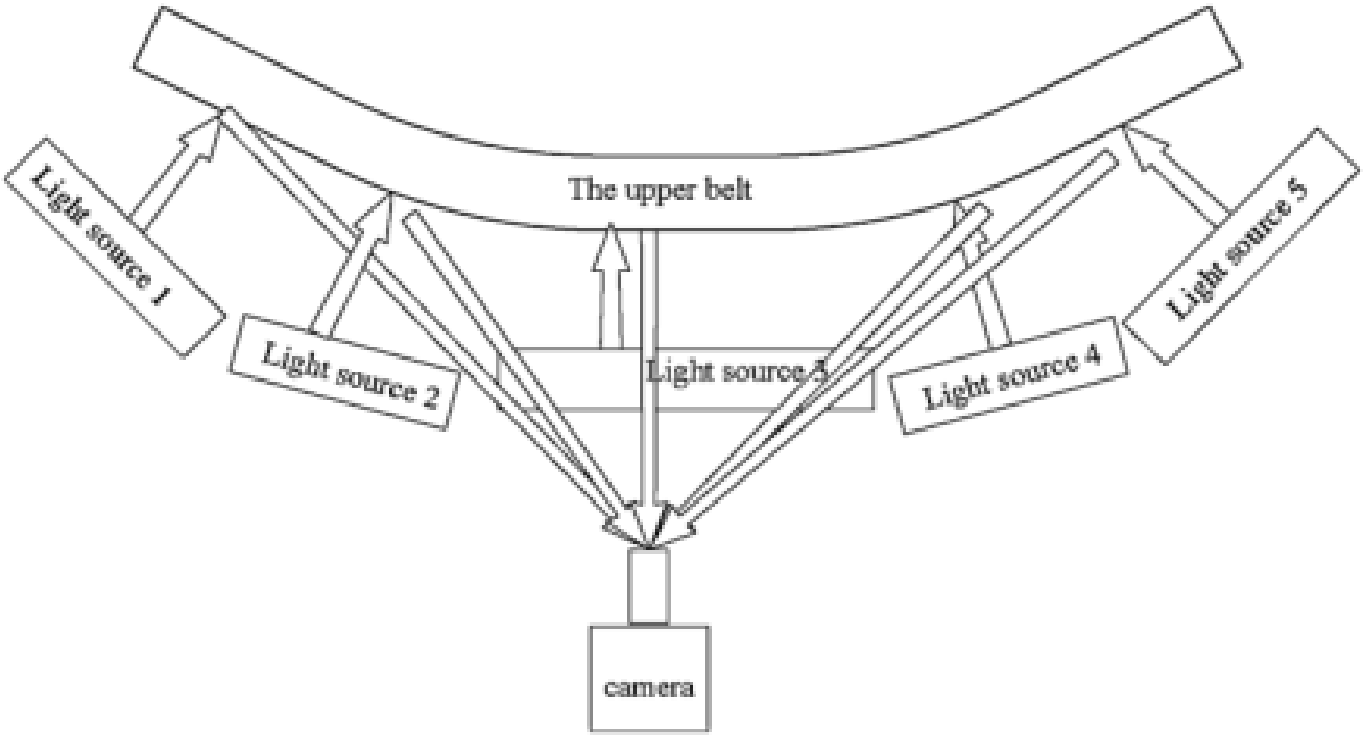

Рисунок 1.6 – Адаптація наведеної системи до використання з жолобчастим конвеєром

Описана у роботі система має такі ж недоліки, як і попередня: залежність від лінійного джерела світла, або спеціальної лінійної камери.

1.3 Постановка завдання дослідження

Завданням дослідження є розробка комп’ютерної системи контролю завантаження стрічкових конвеєрів, що не потребує значних витрат, має невеликий обсяг монтажних робіт, є надійною та простою у обслуговуванні.

Комп’ютерна система має складатись з компонент, що широко представлені на ринку та мають невисоку вартість, задля заміни у разі поламки. Вона має бути спроектована з дотриманням принципів модульності та гнучкості, що дозволить адаптувати її до потреб широкого спектру завдань.

Програмне забезпечення системи має бути простим у конфігуруванні, дотримуватись модульної архітектури, що дозволить змінювати окремі її частини, або розширювати її.

Для передачі інформації система має спиратись на існуючі системи та засоби комунікації, або пропонувати такі, що не потребують надмірних затрат при їх прокладанні та підтримці.

1.4 Висновки по розділу

Транспортування вугілля з шахти на поверхню є однією з основних компонент процесу видобутку. Найефективнішим засобом транспортування вугілля горизонтальними та похилими виробками є конвеєрний транспорт.

Стрічковий конвеєр безперервно приводиться у рух з використанням електроприводу. Найбільша ефективність роботи конвеєра досягається при максимально можливому навантаженні та рівномірному розподілі матеріалу, що пересувається, але фактично конвеєр функціонує у такому режимі не більше ніж у 5% часу.

Для зменшення затрат електроенергії пропонується використовувати конвеєри зі змінною швидкістю стрічки, але нерівномірний розподіл вантажу не дозволяє досягнути значного підвищення ефективності.

Автоматизоване регулювання швидкості конвеєрів для більш рівномірного розподілу матеріалу, що пересувається, неможливе без вхідного сигналу, що характеризує завантаженість конвеєра на деякій окремій ділянці.

Такий сигнал може формуватись комп’ютерною системою контролю завантаження стрічкових конвеєрів, що розробляється в представленій кваліфікаційній роботі.

Розглянуті існуючі варіанти систем контролю завантаження стрічки конвеєра, для функціонування яким потребується джерело лінійного світла, такого, як лінійний лазер.

Сформульоване завдання дослідження і вимоги до комп’ютерної системи, що розробляється.

2 ТЕОРЕТИЧНИЙ РОЗДІЛ

2.1 Принцип функціонування комп’ютерної системи

Комп’ютерна система здійснює захват зображення стрічкового конвеєра з відеокамери, у режимі реального часу виконує над ним необхідні перетворення, застосовує алгоритми комп’ютерного зору для ідентифікації і отримання геометричних параметрів вантажопотоку та стрічки конвеєра.

На деякій ділянці, визначеної конфігурацією програми, обчислюється співвідношення візуальної ширини потоку матеріалу до ширини стрічки, яке виступає у ролі вихідного сигналу. Воно відображається на інтерфейсі комп’ютерної системи, перетворюється в цифровий або аналоговий сигнал та передається через відповідні інтерфейси або виводи.

Визначення контуру потоку вугілля засобами комп’ютерного зору можливе, коли візуально колір стрічки конвеєра відрізняється від кольору вугілля. Навіть без застосування механічних або інших засобів очищення стрічки від вугільного пилу, все ще можливо виділити потік за кольором.

Це можливе завдяки тому, що кускове вугілля менше розсіює світло, ніж вугільний пил. Вугілля утворюється в умовах високого тиску, та навіть при механічному подрібненні його грані залишаються рівними. Допоміжним фактором при знаходженні контуру потоку є тінь, що відкидається ним на стрічку конвеєра.

Таким чином, комп’ютерна система може працювати в умовах натурального освітлення, штучного розсіяного світла, або з використанням декількох джерел освітлення. Якщо людське око здатне без проблем відрізнити вантажопотік на стрічці конвеєра, то існує можливість створення системи комп’ютерного зору, що зможе виконувати ту ж задачу.

2.2 Структура комп’ютерної системи

Комп’ютерна система контролю завантаження стрічкових конвеєрів складається з модулю вводу відеоінформації, модулю обробки відеоінформації, або обчислювального модулю, та модулю користувацького інтерфейсу.

Такі окремі одиниці комп’ютерних систем об’єднуються у мережу за допомогою шини CAN, що дозволяє централізувати збір інформації про стан конвеєрного транспорту. Приклад подібної мережі наведено на рисунку Рисунок 2.1.

Розміщення ліній зв’язку вздовж конвеєрної транспортної системи та шинна топологія мережі дозволяють розмістити екземпляр комп’ютерної системи контролю завантаження в майже будь-якій ділянці конвеєра без прокладання окремих виділених каналів.

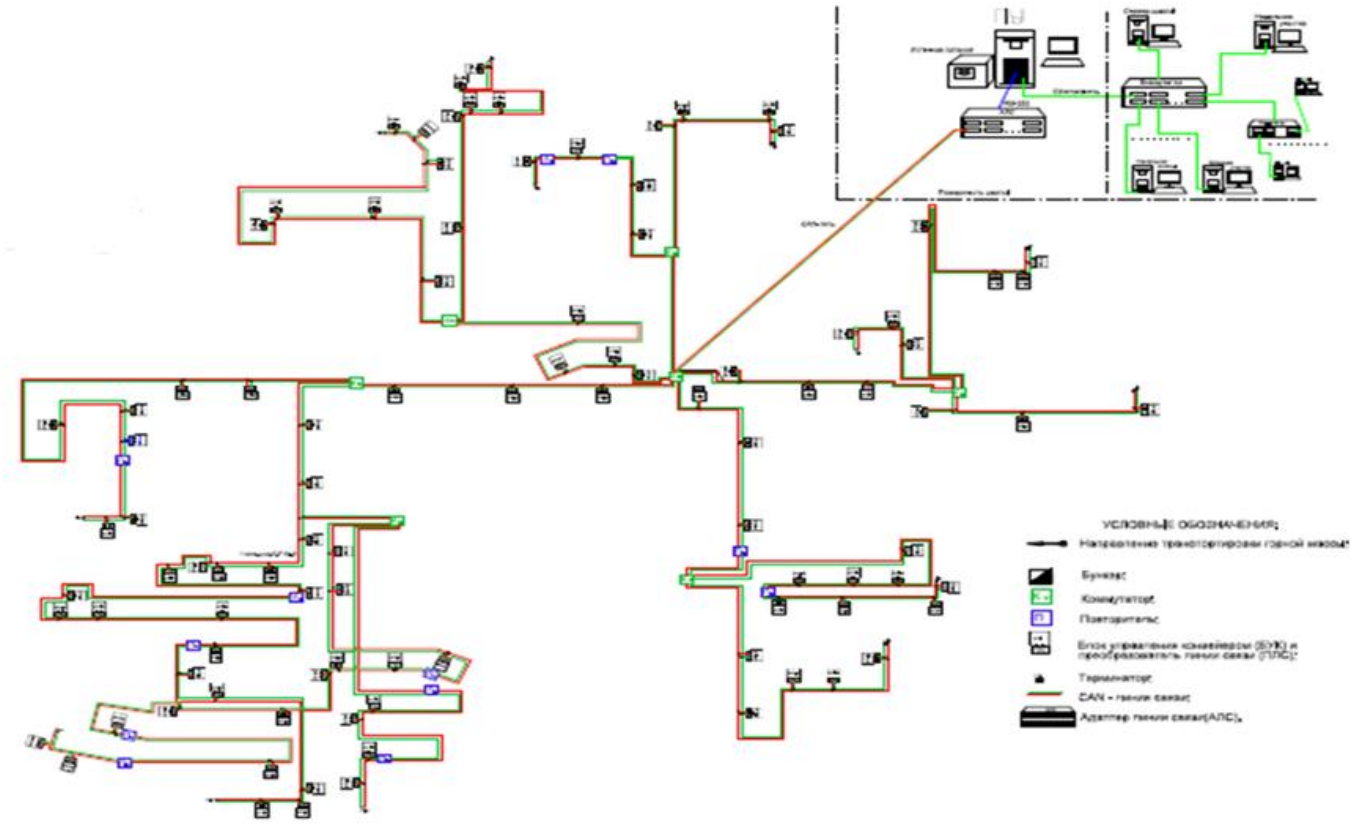

Рисунок 2.1 – Схема CAN мережі, побудованої на основі схеми конвеєрного транспорту шахти

Комп’ютерна система безпосередньо отримує інформацію про об’єкт застосування – стрічковий конвеєр, та взаємодіє з автоматизованою системою

керування швидкістю стрічки через канал передачі інформації. Внутрішні взаємодії між компонентами системи, та зовнішні – з іншими системами, мережею та об’єктами, проілюстровані на рисунку Рисунок 2.2.

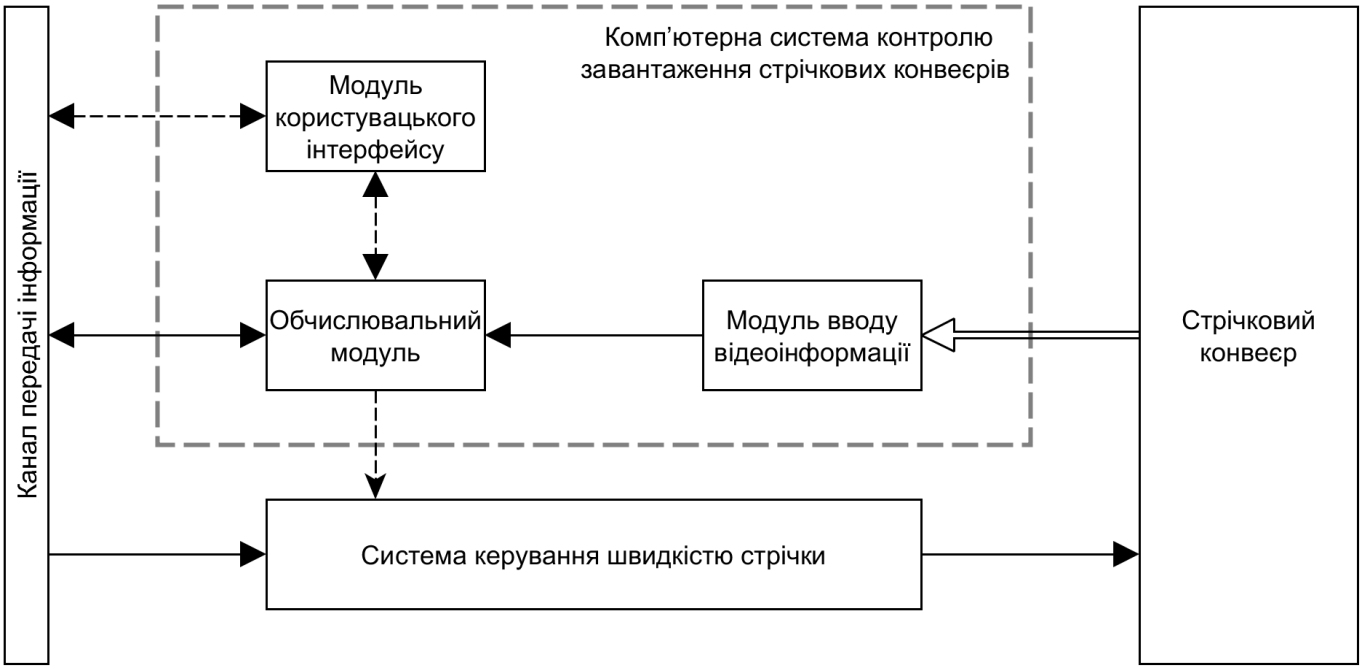

Рисунок 2.2 – Структура комп’ютерної системи

Стрілками на ілюстрації структури відображено напрямок передачі інформації. Біла стрілка символізує пасивний збір інформації одержувачем – модулем вводу відеоінформації. Пунктирними лініями показані альтернативні інформаційні зв’язки.

Модуль вводу відеоінформації відповідає за отримання відеозображення стрічкового конвеєру, перетворення його у цифрову форму та передачу до обчислювального модулю.

Обчислювальний модуль виконує всі необхідні перетворення та застосовує алгоритми комп’ютерного зору до зображення з метою отримання величини завантаження стрічки, та виставляє її як вихідний сигнал для автоматизоавної системи керування швидкістю стрічки. Обчислювальний модуль обмінюється інформацією з модулем користувацького інтерфейсу. Зі сторони користувацького інтерфейсу обчислювальний модуль отримує необхідні налаштування, та надсилає результати обробки зображення.

Обмін інформацією між обчислювальним модулем та АСК швидкістю стрічки допустимий з використанням каналу передачі даних загального призначення, або безпосередньо – шляхом фізичного об’єднання систем виділеним каналом. Так само модуль користувацького інтерфейсу може бути програмно об’єднаним з обчислювальним модулем, або обмінюватись інформацією через комп’ютерну мережу.

Модуль вводу відеоінформації також може бути представлений окремим пристроєм у комп’ютерній мережі, але це не рекомендується через підвищену ймовірність відмови, збільшення часу реакції системи, залежно від стану та завантаженості мережі.

Модуль користувацького інтерфейсу обмінюється інформацією з обчислювальним модулем для відображення отриманого зображення конвеєра разом з текстовими і графічними індикаторами завантаженості стрічки. Оператор може взаємодіяти з інтерфейсом, використовуючи клавіатуру та комп’ютерну мишу, таким чином змінюючи налаштування комп’ютерної системи. Модуль інтерфейсу забезпечує передачу нових налаштувань до обчислювального модуля.

За відсутності потреби у модулі користувацького інтерфейсу, він може бути дезактивований для скорочення використання обчислювальних ресурсів.

2.3 Сценарії застосування комп’ютерної системи

Основне призначення комп’ютерної системи контролю завантаження стрічкових конвеєрів – функціонування у складі автоматизованої системи регулювання швидкості стрічки конвеєра. Але комп’ютерна система може бути модифікована для накопичення статистичних даних, побудови графіку, виведення величини завантаження стрічки у реальному часі.

Завдяки принципу модульності, програмне забезпечення системи може бути доповнене наступними часто використовуваними функціями:

- цифро-аналогове перетворення обчисленої величини апаратними засобами одноплатного комп’ютера, що застосовується у системі;
- накопичення даних у СКБД, використовуючи мову SQL;
- відправки електронних повідомлень з використанням протоколу SMTP;
- виклику програм та процедур на віддалених машинах з використанням RPC.

За тим же принципом деякі структурні компоненти системи можна перемістити або відключити. Наприклад, користувацький інтерфейс, що необхідний для налаштування програми та переглядання рівня завантаження у реальному часі, можна розмістити на іншому ПК, організувавши обмін даними між програмними модулями комп’ютерною мережею. Або використовувати у ролі модулю вводу відеоінформації IP-камеру.

2.4 Обґрунтування і вибір методів дослідження

2.4.1 Аналіз методів вирівнювання гістограми зображення

Гістограма зображення являє собою розподіл інтенсивностей пікселей серед їх загальної кількості. Вирівнювання гістограми – це процес перерозподілу значень інтенсивностей таким чином, щоб вони займали весь доступний діапазон. У загальному випадку, вирівнювання гістограми призводить до підвищення контрасту зображення.

Вирівнювання гістограми може бути описане наступним виразом:

$$V_i' = \frac{V_i - V_{min}}{V_{max} - V_{min}},$$

де V_i – старе значення інтенсивності пікселя, V_i' – нове значення, V_{min} та V_{max} – мінімальне та максимальне значення серед усіх пікселів відповідно.

Приклад застосування вирівнювання гістограми, отриманий під час експериментальних досліджень застосування алгоритму, наведено на рисунку Рисунок 2.3.

Рисунок 2.3 – Приклад застосування вирівнювання гістограми

Адаптивне вирівнювання гістограми – це розвинений варіант звичайного вирівнювання. Сутність цього методу полягає в розбитті вихідного зображення на сітку з невеликих фрагментів, та застосування вирівнювання гістограми індивідуально до кожного з них. Для згладжування переходів на границях фрагментів застосовується білінійна інтерполяція.

Приклад застосування адаптивного вирівнювання гістограми наведено на рисунку Рисунок 2.4.

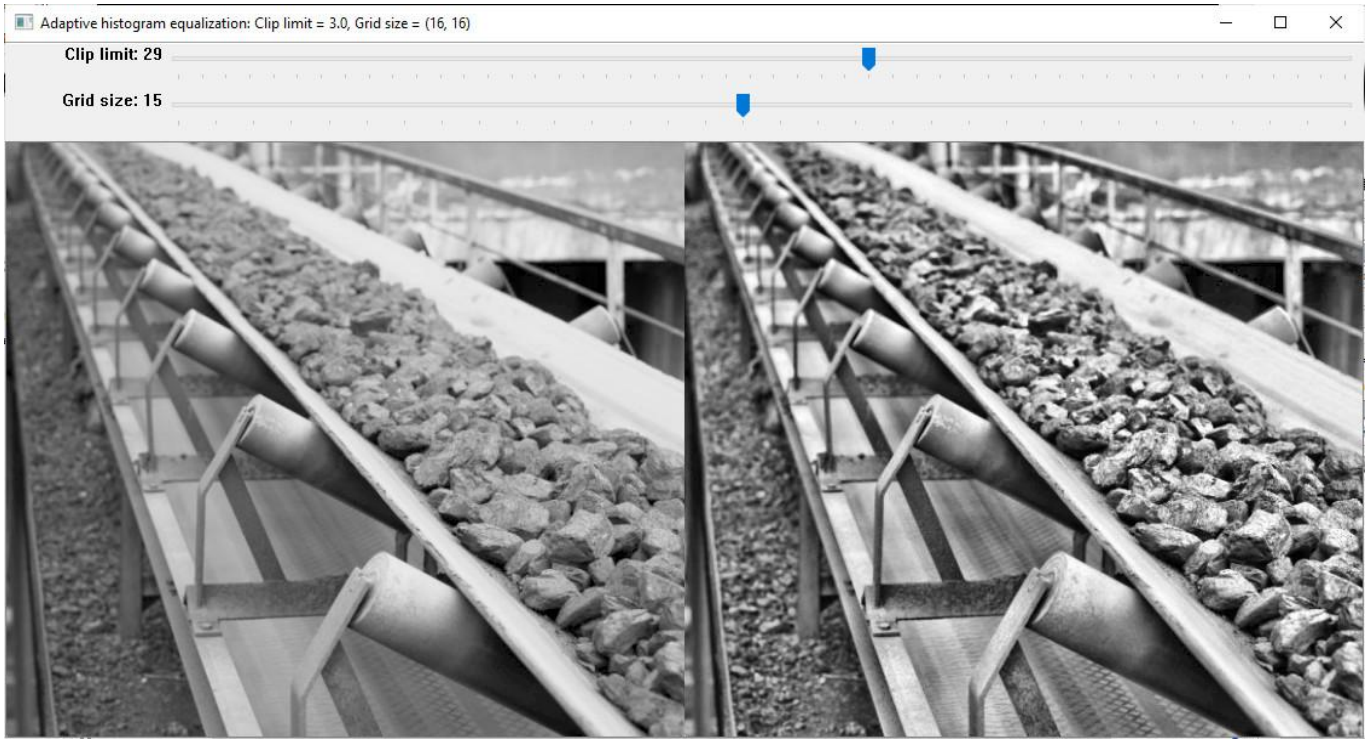

Рисунок 2.4 – Приклад застосування адаптивного вирівнювання гістограми

Застосування будь якого з алгоритмів вирівнювання гістограми для комп'ютерної системи, що розробляється, не є виправданим. В умовах штучного освітлення шахти яскравість досліджуваних об'єктів не буде істотно змінюватись, але поява у кадрі дуже світлих або темних предметів значно впливає на величини інтенсивностей пікселів. Цей вплив нівелюється при застосуванні адаптивного вирівнювання, але підвищується рівень шумів на зображенні, та пропадає можливість бінаризації за загальним значенням порогу для всього зображення.

2.4.2 Аналіз методів бінаризації зображення

Реалізація деяких алгоритмів з обробки зображень, таких, як відстеження контурів, перетворення Хафа, для зображень у градаціях сірого може бути надмірно складна та потребувати великих обчислювальних потужностей. З метою застосування таких алгоритмів зображення перетворюється на бінарне, тобто кожен піксель приймає одне з двох логічних значень: «0» та «1».

Математично це можна виразити наступним чином:

$$V_i' = \begin{cases} 255, V_i \geq t \\ 0, V_i < t \end{cases},$$

де t – величина порогу.

Алгоритм порогової бінаризації прирівнює всі пікселі, чия інтенсивність менше за деякий поріг, до «0», та інші – до «1». Величина порогу може бути вказана вручну, або обчислена за методом Оцу. За цим методом величина порогу обирається таким чином, що розділяє два класи пікселів: переднього плану і фону, таким чином, що їхня внутрішньокласова дисперсія є мінімальною, а міжкласова – максимальною.

Приклад застосування порогової бінаризації наведений на рисунку Рисунок 2.5.

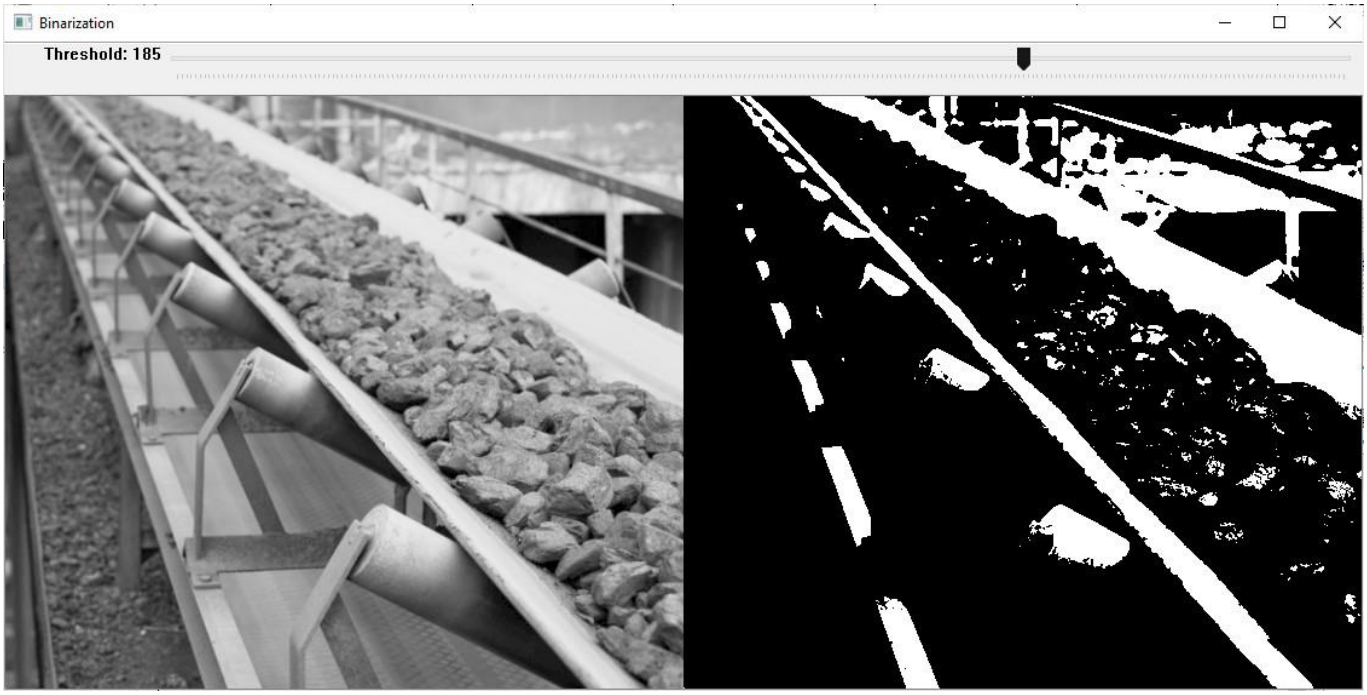

Рисунок 2.5 – Приклад застосування порогової бінаризації

Такий підхід вимагає встановлення сталого значення порогу, або його ручне регулювання. При обробці зображень, знятих у різних умовах освітленості, оптимальне значення порогу може відрізнятися навіть після вирівнювання гістограми. Для автоматичного визначення величини порогу був розроблений метод Оцу. За цим методом величина порогу обирається таким чином, що розділяє два

класи пікселів: переднього плану і фону, таким чином, що їхня внутрішньокласова дисперсія є мінімальною, а міжкласова – максимальною.

Недоліком порогової бінаризації є те, що вона використовує єдине порогове значення для всього зображення. Такий підхід складно застосовувати до нерівномірно освітлених зображень, тому у подібних випадках використовується адаптивна бінаризація.

Алгоритм адаптивної бінаризації визначає порогове значення для пікселю на основі невеликого регіону навколо нього, таким чином, отримуючи кращий результат для зображення з нерівномірним освітленням. Загальний принцип алгоритму подібний до адаптивного вирівнювання гістограми.

Приклад застосування адаптивної бінаризації наведено на рисунку Рисунок 2.6.

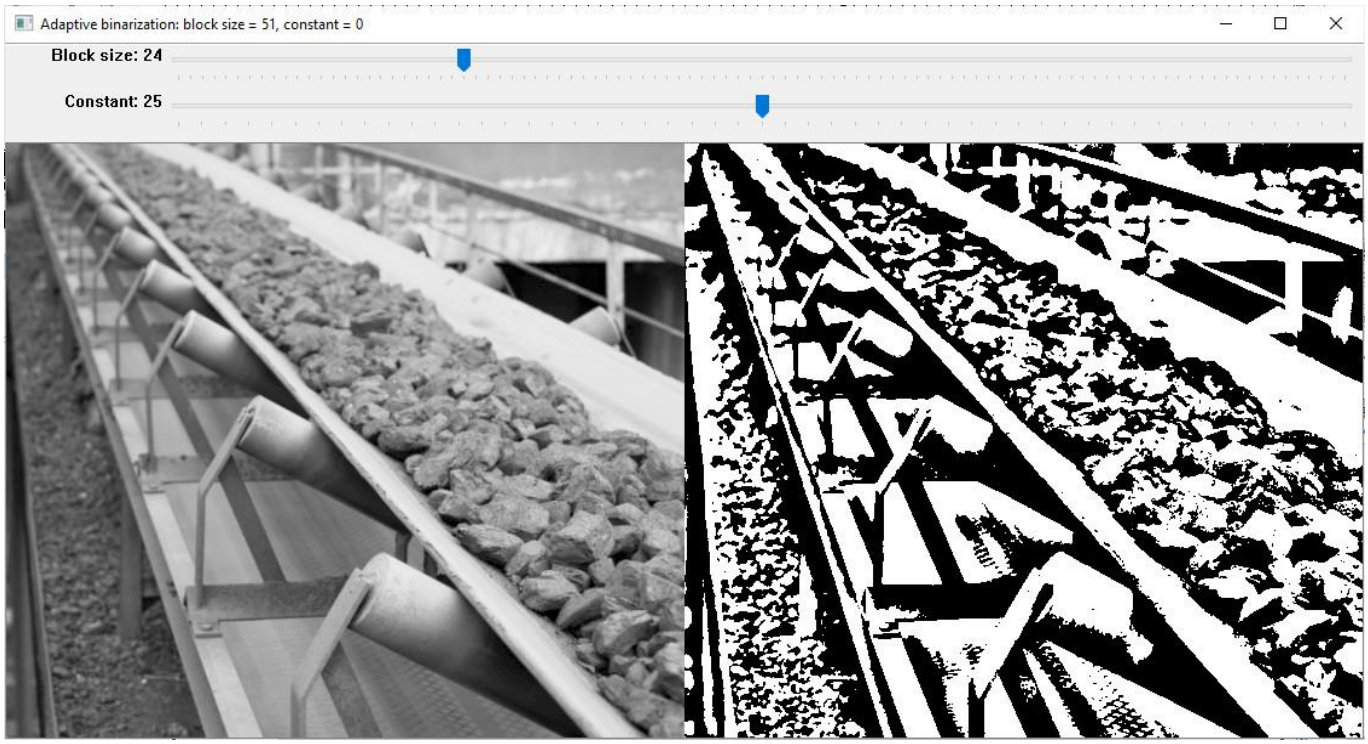

Рисунок 2.6 – Приклад застосування адаптивної бінаризації

Для вирішення завдання дослідження алгоритм адаптивної бінаризації не надасть значної переваги відносно звичайної бінаризації, але потребує більшої обчислювальної потужності та налаштування додаткових параметрів.

2.4.3 Аналіз алгоритму Кенні для виявлення границь

Як і порогова бінаризація, алгоритм Кенні призначений для зменшення даних, що обробляються при застосуванні методів комп’ютерного зору, шляхом виділення найбільш значимої структурної інформації зображення.

Алгоритм складається з п’яти кроків:

- Застосування фільтру Гауса для розмиття зображення з метою видалення шуму;
- Знаходження градієнтів інтенсивності зображення по горизонталі та вертикалі з використанням оператора Собеля;
- Застосування «придушення немаксимумів» (Non-Maximum Suppression) для видалення хибних відкликів.
- Застосування подвійного порогу для визначення потенційних границь
- Відстеження границь за гістерезисом: завершення алгоритму шляхом видалення «слабких» границь, що не з’єднані з «сильними».

Приклад застосування алгоритму Кенні для виявлення границь на рисунку Рисунок 2.7.

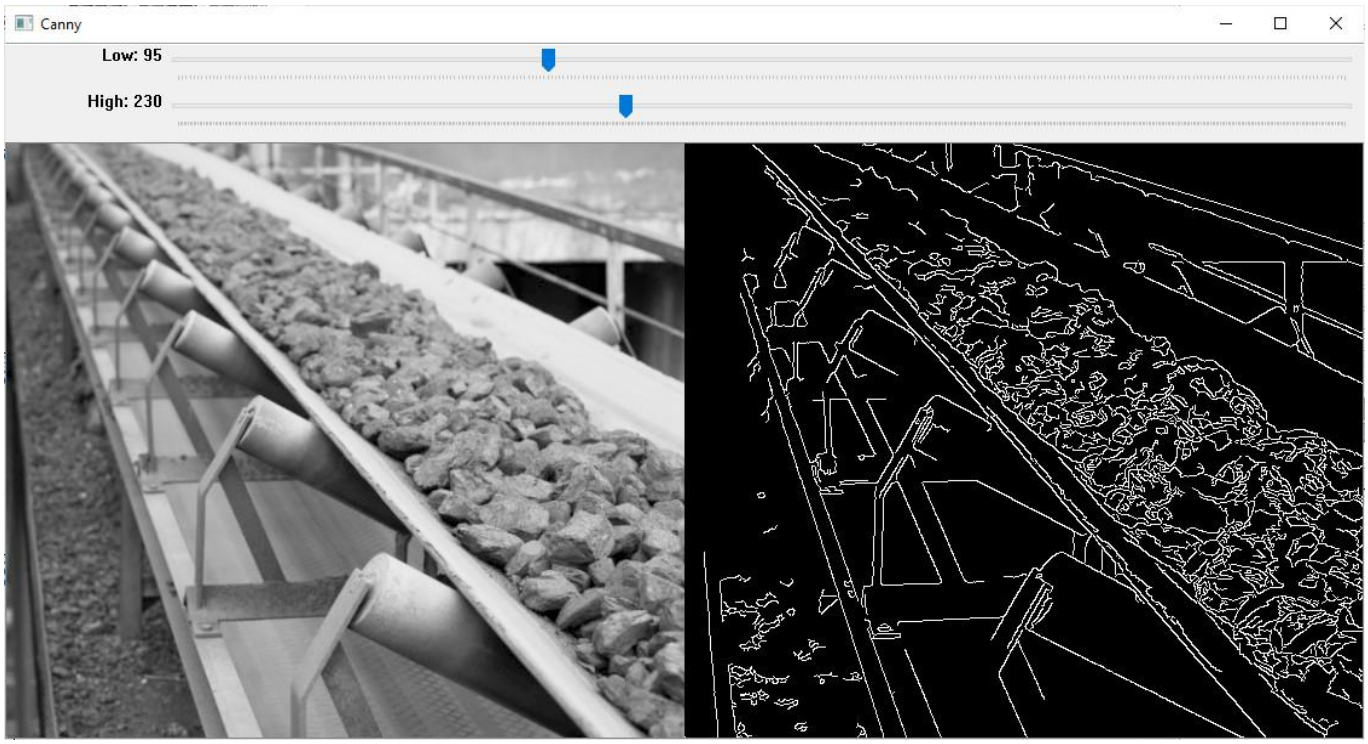

Рисунок 2.7 – Приклад застосування алгоритму Кенні

В умовах поставленої задачі алгоритм Кенні придатний лише до застосування разом з перетворенням Хафа для пошуку ліній, тому його ефективність визначається ефективністю наступного алгоритму.

2.4.4 Аналіз перетворення Хафа для пошуку прямих

Перетворення Хафа – це техніка, що дозволяє виявити на зображенні будь-яку фігуру, що може бути описана математично, навіть якщо вона порушена, або дещо деформована. Така результативність досягається шляхом здійснення процедури голосування в просторі параметрів, кандидати у об’єкти з якого отримуються як локальні максимуми у «накопичувальному просторі» (accumulator space).

При застосуванні перетворення Хафа для пошуку прямих, ті представляються в параметричній формі як:

$$\rho = x\,cos(\theta) + y\,sin(\theta),$$

де ρ – довжина перпендикуляру, опущеного на пряму з вихідної точки, а θ – кут між перпендикуляром та горизонтальною віссю, що вимірюється проти годинникової стрілки.

Накопичувальний простір є двовимірним масивом, розмір якого визначається параметрами ρ і θ. За допустимої точності куту в 1°, θ може приймати 180 значень. Найбільша можлива дистанція ρ – це довжина діагоналі зображення.

Для кожної ненульової точки бінарного зображення в параметричне рівняння прямої підставляються всі можливі значення θ = 0, 1, 2, …, 179, обчислюється величина ρ , та в накопичувальному просторі за відповідними координатами (ρ, θ) значення збільшується на одиницю.

Результатом перетворення є масив параметрів (ρ, θ), що задовольняють мінімальній величині голосу після процедури голосування (порогове значення).

Приклад застосування перетворення Хафа разом з алгоритмом Кенні наведено на рисунку Рисунок 2.8.

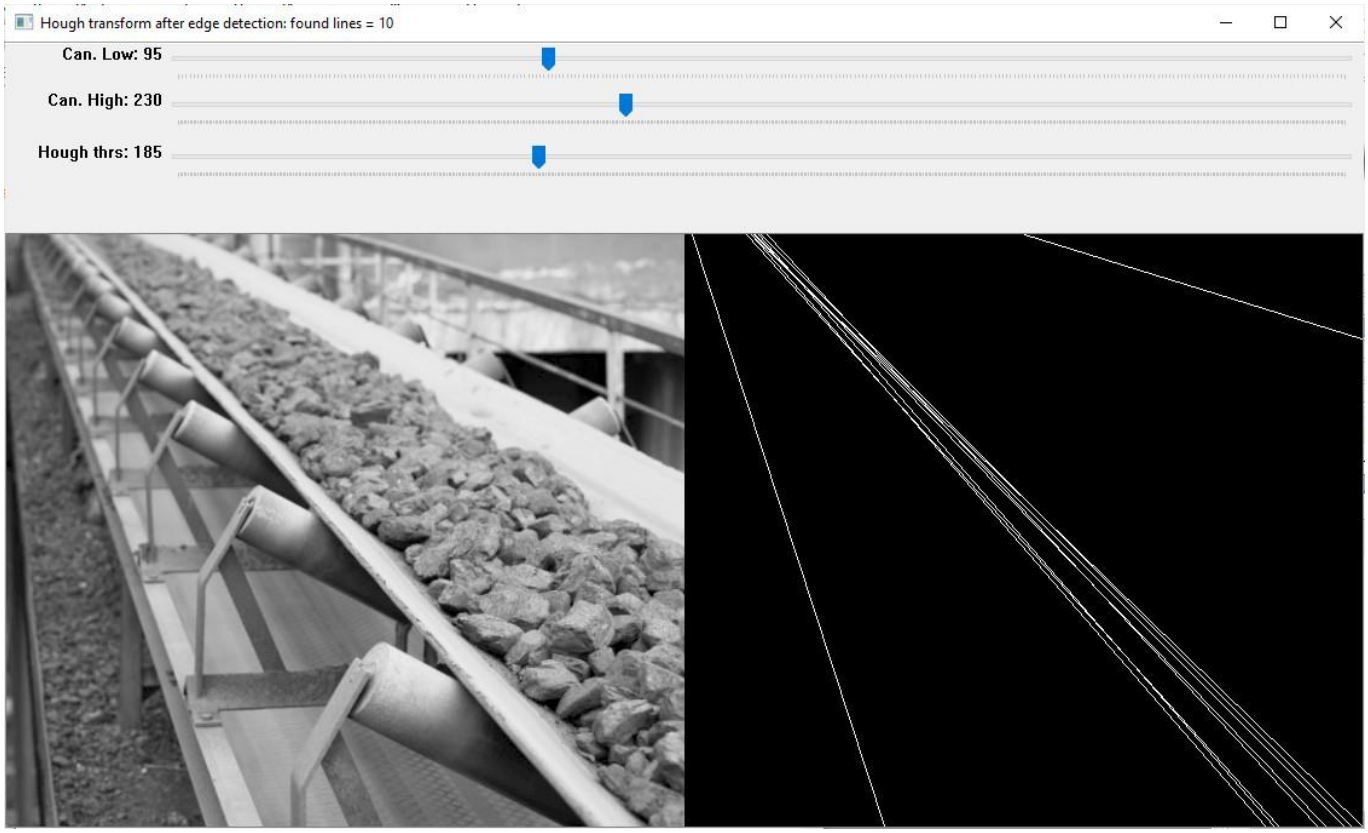

Рисунок 2.8 – Приклад застосування перетворення Хафа

Перетворення Хафа разом із алгоритмом Кенні для виділення границь вимагає індивідуального налаштування декількох параметрів. Також, при застосуванні на практиці, нерівний край стрічки, подряпини, сліди у вигляді ліній та сам вантажопотік вносять численні похибки в параметричний простір,що призводить до неправильного виявлення, або дублювання прямих. У більшості випадків неможливо за одних і тих же параметрів однозначно виділити обидва краї стрічки.

2.4.5 Аналіз методів згладжування зображення

Згладжування, або розмиття зображення допомагає усунути шум на зображенні, але розмиває границі об’єктів. Воно досягається шляхом застосування згортки.

Згортка – це процес додавання елемента зображення та його сусідів з коефіцієнтами, визначеними матрицею-ядром. Ядро рівномірного розмиття розміром 3 на 3 виглядає наступним чином:

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

При застосуванні згортки з таким ядром до пікселю зображення, його значення буде замінене середнім арифметичним самого пікселя та його сусідів. Різновиди розмиття відрізняються принципом утворення ядра згортки.

Приклад застосування рівномірного розмиття наведено на рисунку Рисунок 2.9.

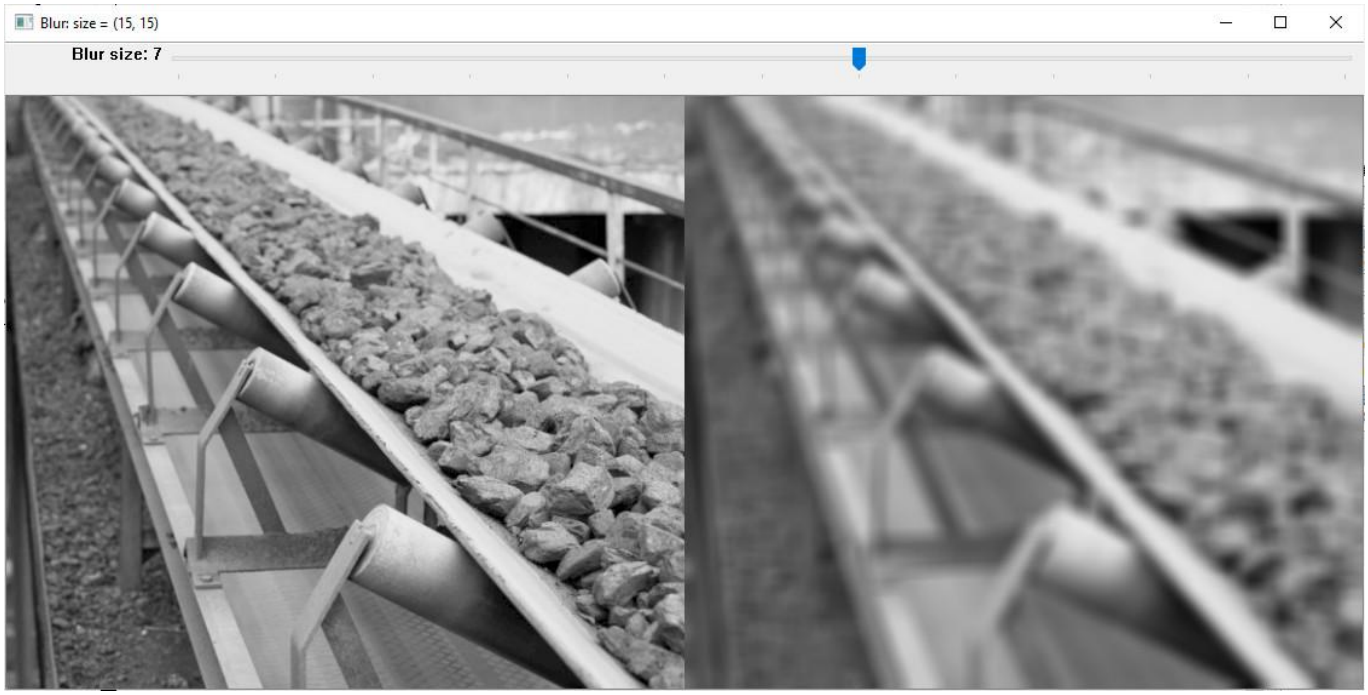

Рисунок 2.9 – Приклад застосування рівномірного розмиття

Ядро згортки Гаусова розмиття виглядає наступним чином:

$$K = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Двостороння фільтрація – це алгоритм згладжування зображення, що бере до уваги різницю між значеннями пікселів, та застосовує Гаусове розмиття до регіонів,

де різниця не є великою. Таким чином, зменшення рівню шуму досягається без розмиття границь. Але даний вид згладжування використовує дуже багато обчислювальних ресурсів, тому доцільність його використання у системах реального часу ставиться під сумнів.

Приклад застосування двосторонньої фільтрації наведено на рисунку Рисунок 2.10.

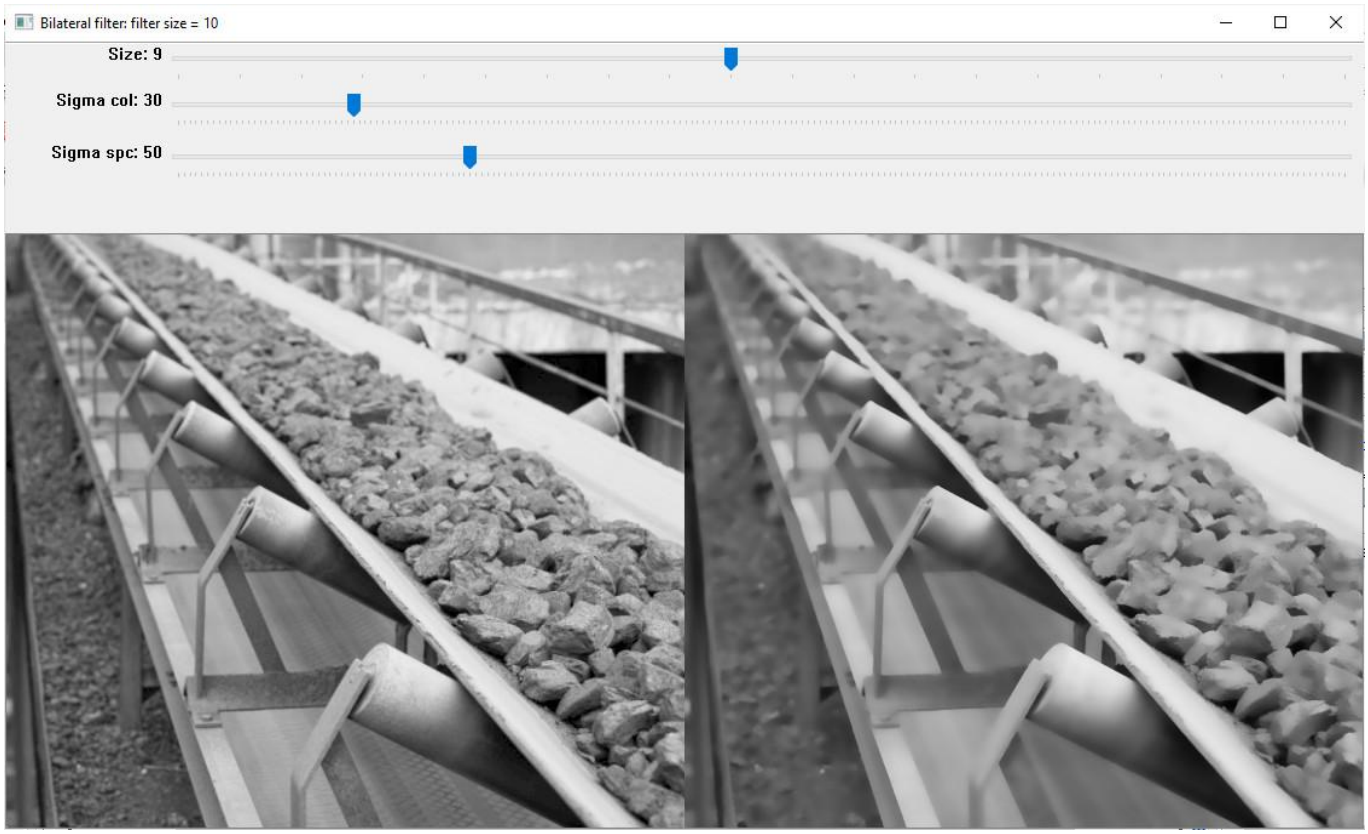

Рисунок 2.10 – Приклад застосування двосторонньої фільтрації

2.4.6 Метод пошуку контурів на бінарному зображенні

Контурами називають криві, що об’єднують точки (уздовж границі) фігури, що мають однаковий колір або інтенсивність. Контури використовуються для маскування, аналізу форми, виявлення та розпізнавання об’єктів. Для більшої точності відстеження контурів виконується на бінарних (чорно-білих) зображеннях.

Можливість виконувати операції над областями зображення, що описані контурами, дозволить застосовувати метод для зменшення кількості пропущених ділянок всередині фігури вантажопотоку після застосування бінаризації.

У прикладі пошуку і заповнення контурів на бінарному зображенні на рисунку Рисунок 2.11 було знайдено і заповнено 72 замкнені області. Порівняти ефективність застосування цієї послідовності методів можна з зображенням на рисунку Рисунок 2.5.

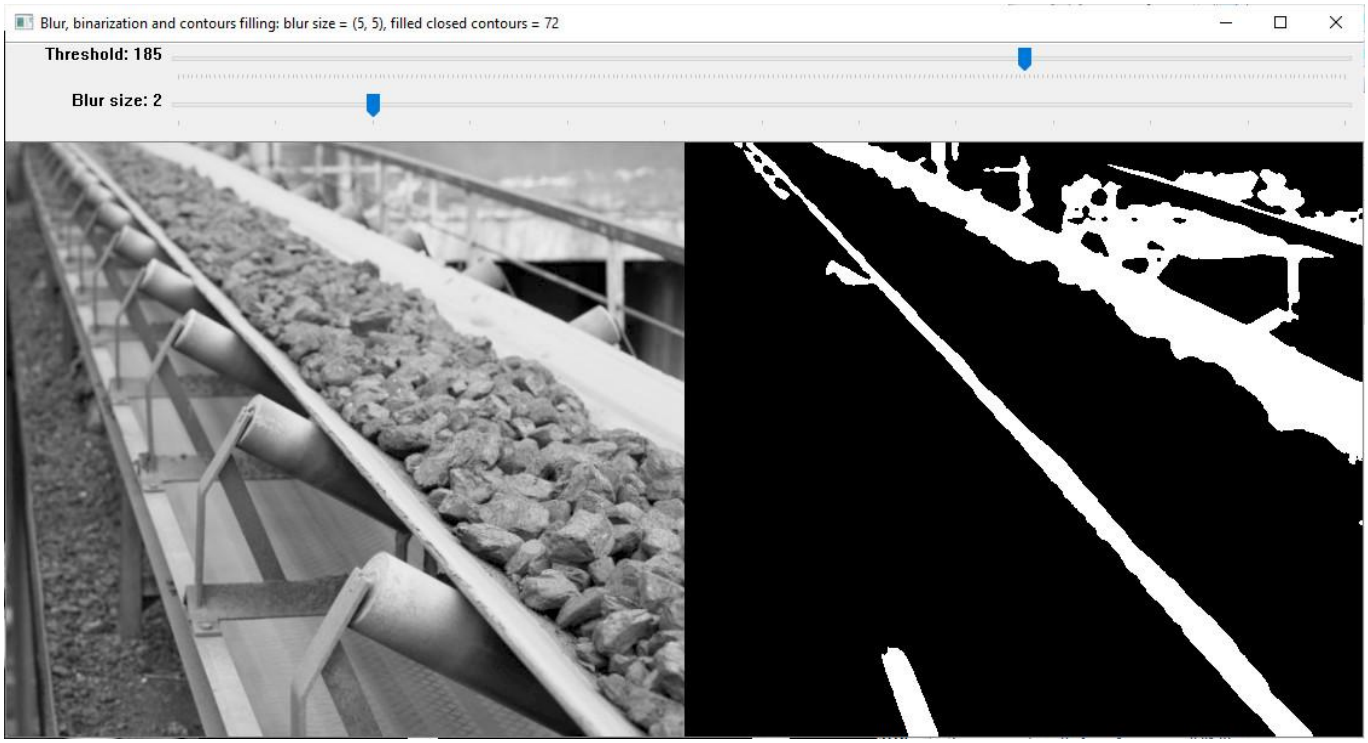

Рисунок 2.11 – Приклад пошуку і заповнення контурів на бінарному зображенні

2.5 Висновки по розділу

У розділі визначено принцип функціонування комп’ютерної системи контролю завантаження стрічки конвеєра, структуру її компонент, внутрішніх та зовнішніх інформаційних зв’язків складових системи. Досліджені алгоритми обробки цифрових зображень, що можуть бути застосовані для функціонування комп’ютерної системи.

Модульна структура системи дозволяє обрати та запровадити один з декількох варіантів побудови, використовуючи комп’ютерні мережі загального призначення, що реалізують протокол IP.

З оглядом на доцільність, ефективність та функціональну простоту, для використання у комп’ютерній системи контролю рекомендуються алгоритм розмиття зображення та порогової бінаризації.

Алгоритми вирівнювання гістограми не підходять для попередньої обробки зображення через те, що можуть непередбачуваним шляхом внести похибку у значення всіх пікселів зображення, та підвищують рівень шуму у результаті застосування.

Перетворення Хафа не дає бажаний результат через вплив високого рівню шумів на параметричний простір при застосуванні на реальних зображеннях конвеєра.

Двостороння фільтрація ефективно вирішує завдання видалення шуму зі статичного зображення, але, з оглядом на низьку швидкість застосування, доцільність використання алгоритму у системі реального часу, що має обробляти до декількох десятків кадрів у секунду, є сумнівною.

3 СИНТЕЗ СИСТЕМИ КОНТРОЛЮ

3.1 Вимоги до комп’ютерної системи та каналу передачі даних

Комп’ютерна система контролю завантаження стрічки конвеєра є періодичною системою м’якого реального часу. Цикл роботи системи, що складається з отримання відеозображення конвеєра, застосування необхідних перетворень і алгоритмів, та отримання величини завантаженості стрічки, має не перевищувати 1000 мс.

Обчислювальний модуль комп’ютерної системи має забезпечувати своєчасну обробку зображень, достатній об’єм оперативної пам’яті для збереження оригінального зображення та результатів його проміжних перетворень, достатній об’єм енергонезалежної пам’яті для збереження налаштувань програмного забезпечення. Модуль повинен підтримувати протокол IP, та відповідно мати інтерфейси, що застосовуються при організації IP мереж: IEEE 802.3 (Ethernet), IEEE 802.11 (Wi-Fi) та інші. Додатковою перевагою буде підтримка стандартів промислових мереж, таких, як CAN, Modbus та інших.

Модуль вводу відеоінформації комп’ютерної системи має забезпечувати отримання чіткого, освітленого зображення у цифровому вигляді висотою не менш за 600 точок . Камера модулю має розташовуватись над стрічкою конвеєра, та бути спрямованою перпендикулярно до нього, як зображено на рисунку Рисунок 3.1. Незначне відхилення від цієї вимоги допустиме, але призводить до виникнення погрішності результату.

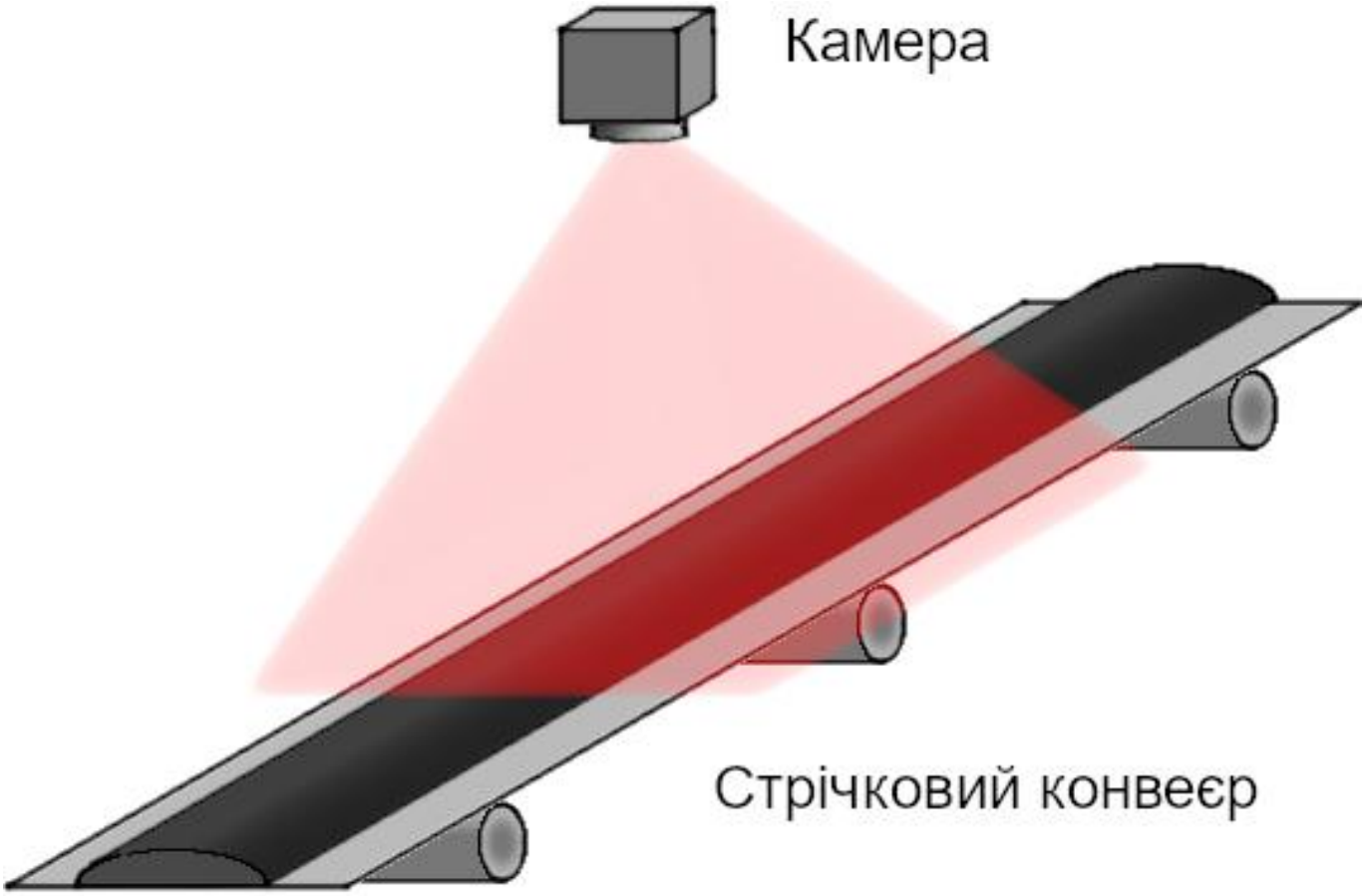

Рисунок 3.1 – Розташування камери модулю вводу відеоінформації

Ділянка стрічки конвеєра, що контролюється системою, має бути рівномірно освітлена з використанням джерела штучного розсіяного світла, або декількох джерел точкового світла, віддалених один від іншого.

3.2 Вибір апаратних складових комп’ютерної системи

Для вирішення завдань з області комп’ютерного зору продуктивності мікроконтролерів вже не вистачає. Тому необхідно звернутись до більш продуктивного класу портативних обчислювальних пристроїв – одноплатних комп’ютерів (SBC).

Станом на сьогодні на ринку існує багато варіантів конфігурації одноплатних комп’ютерів, які об’єднані загальними рисами: невеликий розмір монтажної плати, наявність виводів загального призначення GPIO, інтерфейсу Ethernet, процесор з ARM архітектурою та SD-карта у ролі постійного накопичувача (Flash-пам’ять).

Окрім Ethernet, одноплатні комп’ютери зазвичай підтримують інтерфейси I2C та SPI. Останній дозволяє підключати сторонні мікросхеми, наприклад, адаптер шини CAN.

Для застосування у якості обчислювального модулю комп’ютерної системі рекомендується популярна модель Raspberry Pi 3B з 4-ядерним процесором на сучасній платформі ARM версії 8, фотографія якого наведена на рисунку Рисунок 3.2.

Рисунок 3.2 – Фотографія мікрокомп’ютера Raspberry Pi 3B

Основні технічні характеристики одноплатного комп’ютера Raspberry Pi 3B:

- Чотириядерний 64-бітний процесор Broadcom BCM2837 з частотою 1.2GHz ;
- 1 Гбайт оперативної пам’яті;
- Інтерфейс Ethernet 100Base-TX;
- 40-контактний розширений GPIO;
- 4 порти USB 2.0;
- 4 Pole stereo output and composite video port;

- Порт камери CSI для підключення камери Raspberry Pi;
- Порт дисплея DSI для підключення сенсорного дисплея Raspberry Pi;
- Micro SD порт для завантаження операційної системи та зберігання даних;
- Живлення через Micro USB з силою струму до 2.5A.

Для застосування у якості модулю введення відеозображення рекомендується модуль Raspberry Pi Camera Module v2 розроблений для використання з даним одноплатним комп’ютером. що зображений на рисунку Рисунок 3.3.

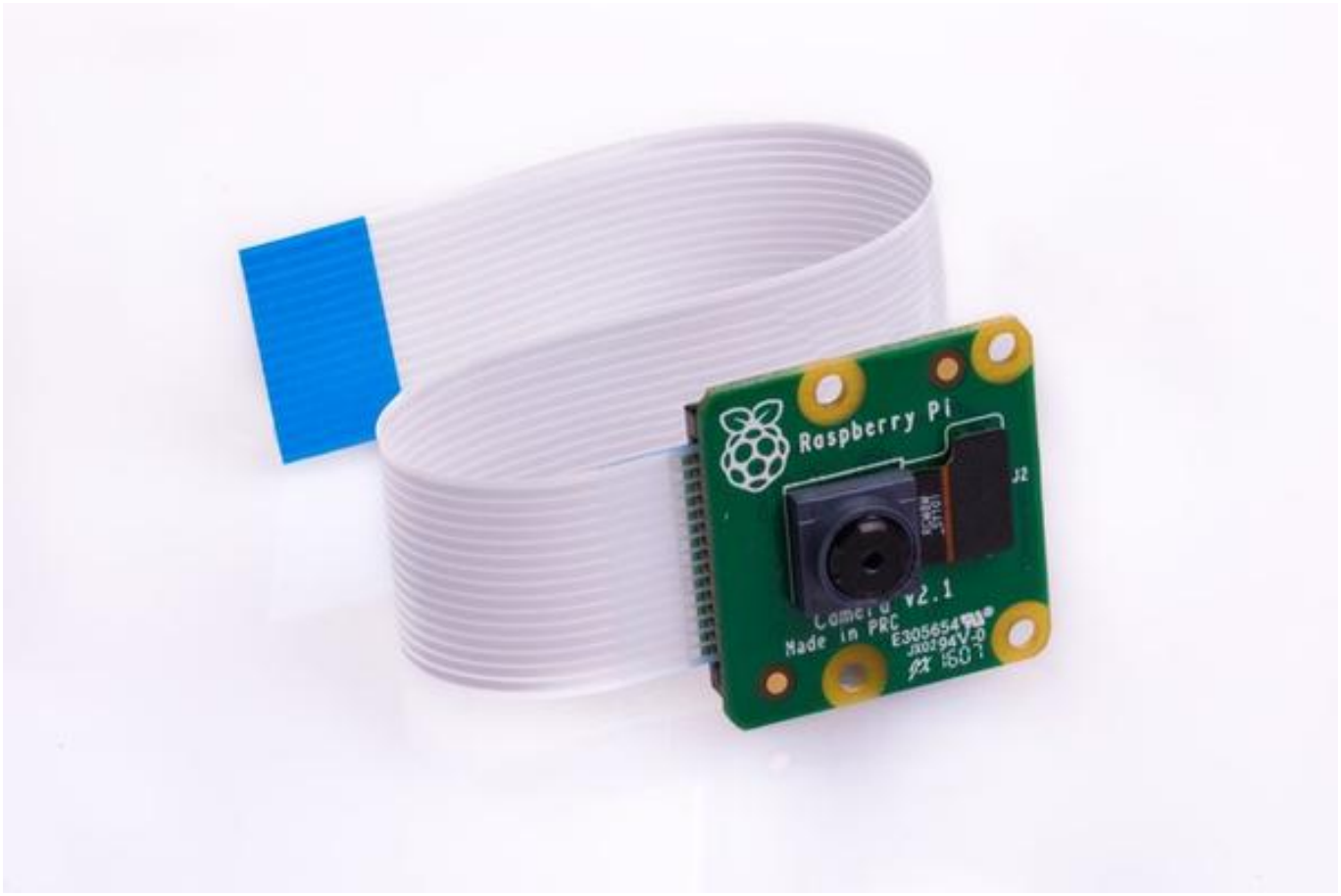

Рисунок 3.3 – Фотографія модулю камери Raspberry Pi

Основні технічні характеристики модулю камери другої версії:

- Сенсор камери: Sony IMX219;
- Роздільна здатність сенсору матриці: 8 Мегапікселів;
- Режими зйомки відео: 1080p30, 720p60, 480p 60/90 (висота кадру у пікселях та частота кадрів).

На роль модулю користувацького інтерфейсу пропонується сенсорний дисплей Raspberry Pi (на рисунку Рисунок 3.4). Наявність сенсорного вводу та віртуальної екранної клавіатури в ОС Raspberry Pi дозволить використовувати систему без фізичної клавіатури з мишею, які швидко запилюються та потребують розбирання для чистки.

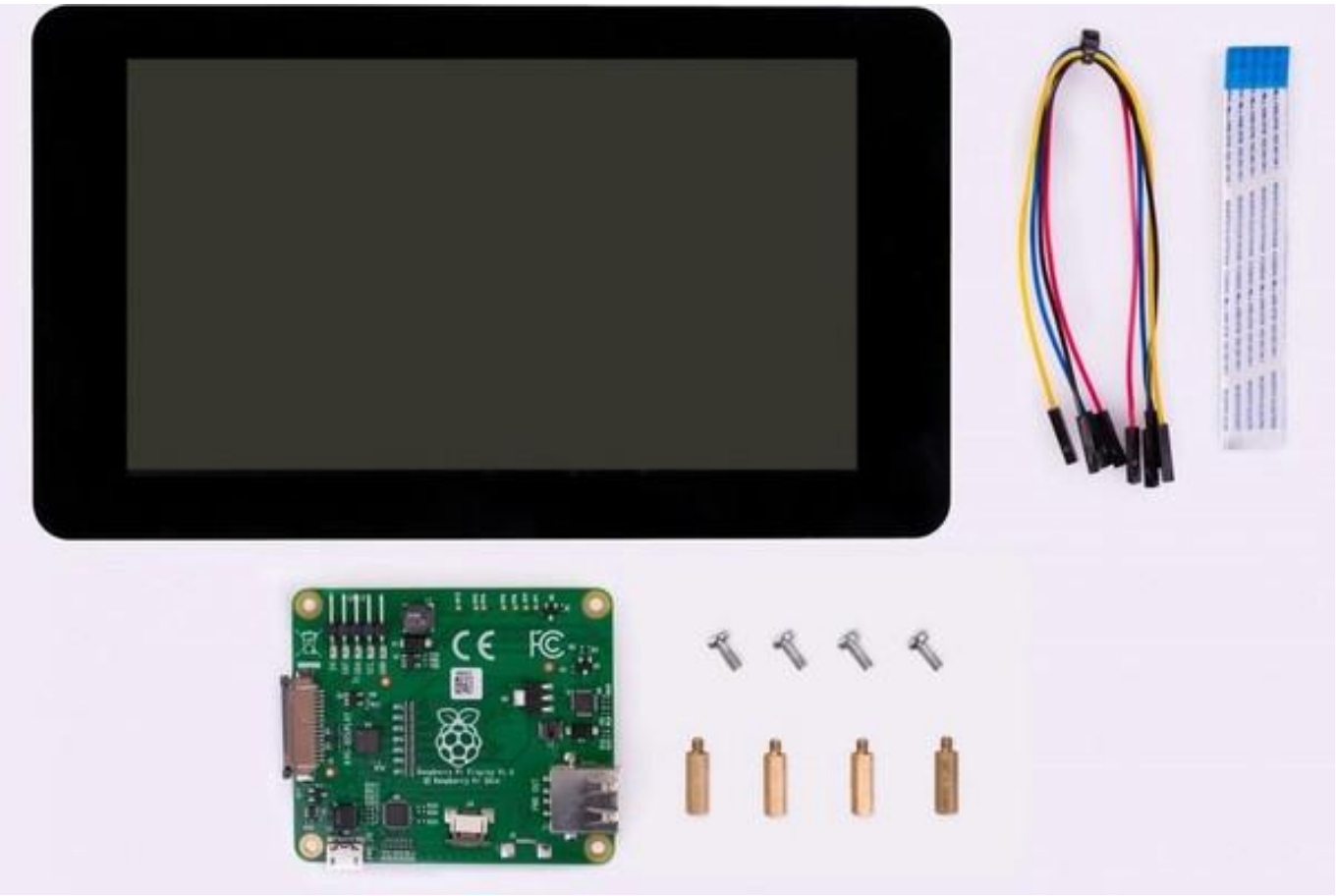

Рисунок 3.4 – Фотографія сенсорного дисплею Raspberry Pi

Основні технічні характеристики сенсорного дисплею:

- Розмір екрану: 7”;
- Роздільна здатність матриці екрану: 800x480 пікселів;
- Кількість дотиків, що одночасно розпізнаються: 10.

3.3 Розробка структурної схеми комп’ютерної системи

Виходячи з обраних апаратних складових комп’ютерної системи розроблена структурна схема (на рисунку Рисунок 3.5), що демонструє зв’язки між ними. У якості прикладу було припущено, що комп’ютерна система об’єднується в CAN мережу для використання у складі автоматичної системи контролю швидкості стрічки, та має власний дисплей з сенсорним екраном або клавіатурою з комп’ютерною мишкою. У якості альтернативного варіанту, можна задіяти розташований на платі Raspberry Pi Ethernet-порт для передачі обчисленої величини та відеоінформації.

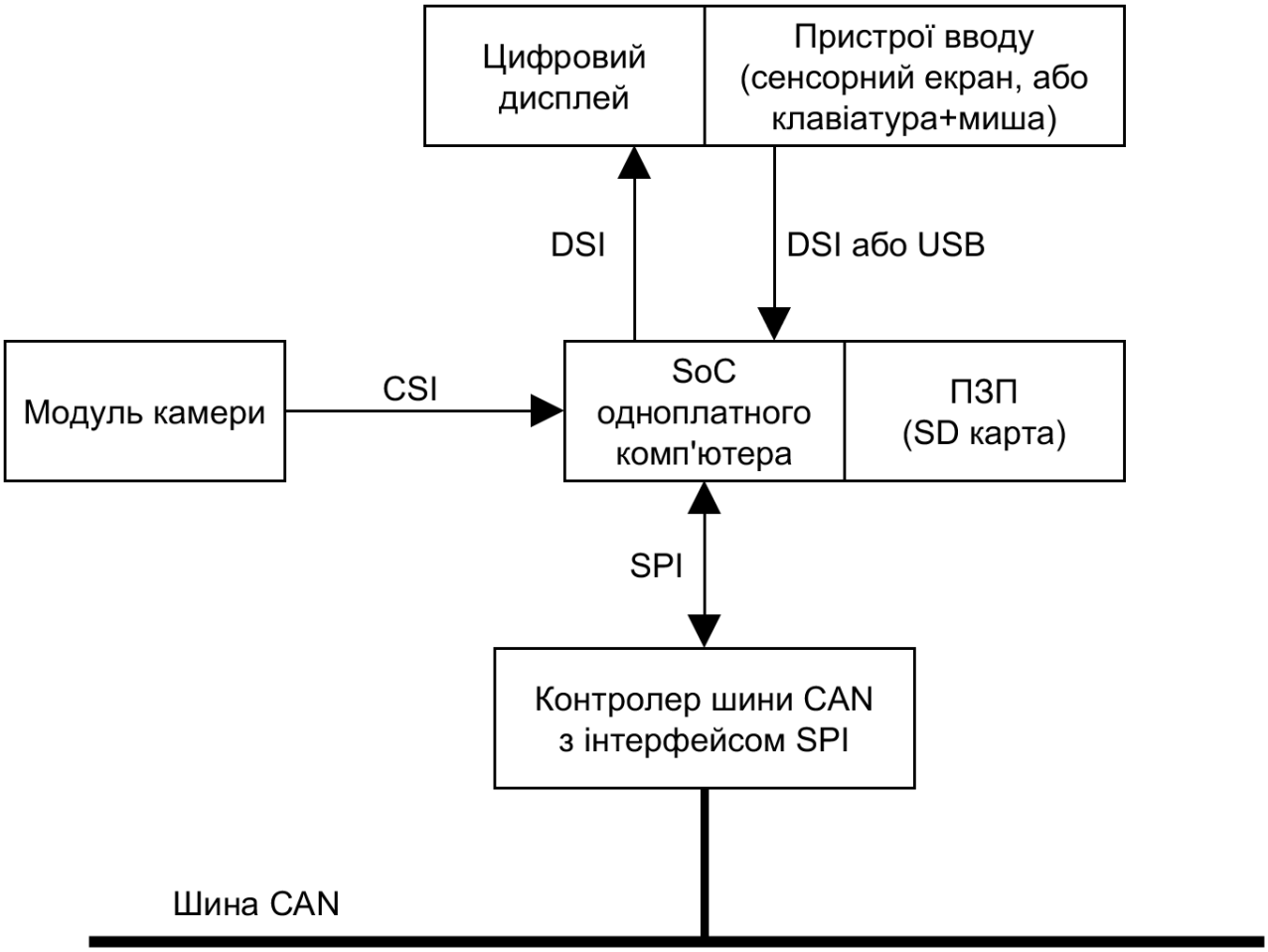

Рисунок 3.5 – Структурна схема комп’ютерної системи

На схемі стрілками позначено напрям передачі даних,а надписи біля них вказують на тип використовуваного інтерфейсу. Через те, що схема одноплатного

комп’ютера не є загальнодоступною, інтерфейси зв’язку з SD-картою та контролером Ethernet (не присутній на схемі, бо не задіяний у прикладі) невідомі.

3.4 Розробка функціональної схеми модуля вводу відеоінформації

Використовуючи обрані в другому розділі алгоритми комп’ютерного зору та структурну схему комп’ютерної системи, було розроблено функціональну схему модулю вводу відеоінформації, наведену на рисунку Рисунок 3.6.

Схема відображає напрямки та перетворення вхідної інформації, необхідні для отримання результату. Стрілками показаний напрямок інформації, надпис над (або справа від) стрілки вказує на тип, зміст або призначення порції інформації, а блоками показані функціональні модулі (функції), що здійснюють відповідну операцію.

Проілюстрований процес обробки є одиничною ітерацією у циклі роботи комп’ютерної системи, що має повторюватись не рідше ніж раз на 1000 мс.

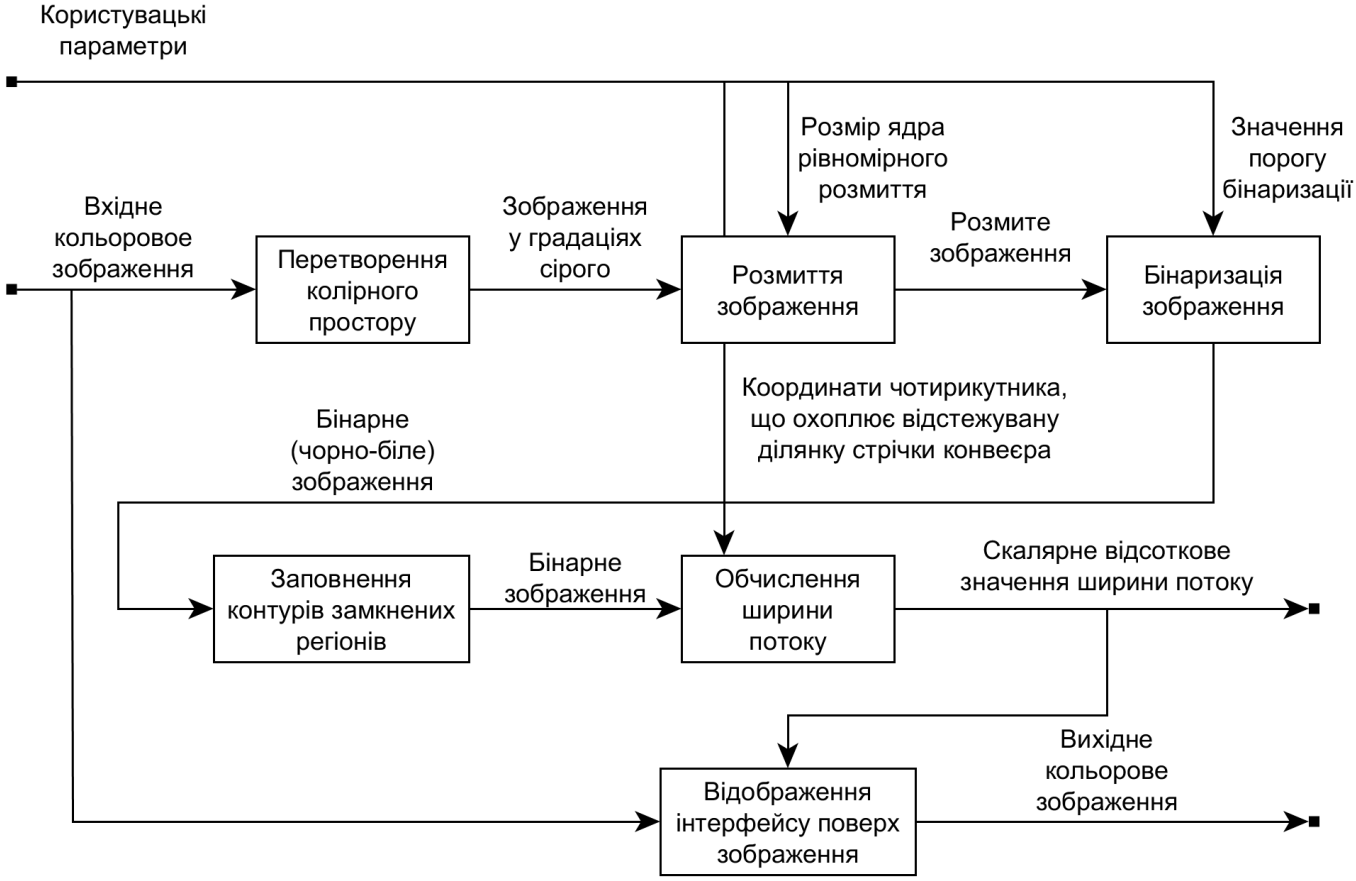

Рисунок 3.6 – Функціональна схема модулю вводу відеоінформації

Вхід «користувацькі параметри» на схемі являє собою структуру параметрів, які задаються оператором з використанням пристрою вводу інформації. Структура складається з величини розміру сторони квадрату ядра, що застосовується у згортці для розмиття зображення, величини порогу бінаризації зображення у градації сірого та чотирьох пар координат, що утворюють чотирикутник – регіон на відеозображенні, за яким відбувається контроль.

3.5 Висновки по розділу

У розділі було сформульовано вимоги до комп’ютерної системи та до умов її функціонування, вибрані апаратні засоби, на основі яких можлива розробка системи, створено структурну схему з використанням обраних засобів, та функціональну схему, що узгоджує досліджувані алгоритми разом зі структурою системи в єдиному процесі функціонування системи.

Для розробки комп’ютерної системи рекомендується лінійка одноплатних комп’ютерів Raspberry Pi, та модулів розширення к ним. Ця продукція відома тим, що застосовується у різноманітних проектах вбудованих систем, має детальну документацію та широку підтримку з боку виробника та сторонніх розробників.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

4.1 Призначення й сфера застосування програми

Програмне забезпечення комп’ютерної системи призначене для застосування у складі систем контролю, вимірювання, автоматизації конвеєрного транспорту, що базуються на розпізнаванні об’єктів на цифровому зображенні.

Програма може бути пристосована до таких систем у сферах видобування корисних копалин, металургії, будівництві, та інших, де стрічковий конвеєр використовується для транспортування сипучих матеріалів.

4.2 Обґрунтування технічних характеристик програми

4.2.1 Постановка завдання на розробку програми

На основі обраних в результаті аналізу алгоритмів та методів комп’ютерного зору, розробленої функціональної схеми процесу обробки інформації, розробити програму, що визначає та відображає видиму ширину потоку матеріалу на цифровому зображенні конвеєра у процентному відношенні до видимої ширини стрічки конвеєра.

Із застосуванням алгоритмів комп’ютерного зору та користувацьких налаштувань програма має однозначно виділяти потік транспортованого матеріалу шляхом фільтрації та усунення некоректно визначених ділянок зображення.

Програма має бути простою та ефективною у використанні. Інтерфейс програми повинен мати мінімальну кількість віджетів для налаштування, які дозволять в очевидний спосіб змінювати конфігурацію. Для наочності роботи програма має виводити отримане зображення контрольованого об’єкту, та за допомогою текстових і графічних індикаторів відображати отриманий результат та користувацькі налаштування.

Конфігурація програми повинна зберігатись у енергонезалежну пам’ять, та відновлюватись після повторного запуску.

4.2.2 Структура і алгоритм функціонування програми

Функціонування комп’ютерної системи має періодичний характер, тому основний процес виконання програми буде являти собою цикл, в якому після виконання необхідних операцій присутня часова затримка такої довжини, що задовольняє вимогам з періодичності системи. Окрім циклу, програма має містити процедури ініціалізації та завершення роботи.

Для дотримання принципу модульності, програма має застосовувати парадигму ООП. Ключове поняття парадигми – об’єкт, який являє собою набір даних та функцій (методів), що застосовуються до цих даних. Класи – це шаблони об’єктів, що описують структуру даних та реалізації методів.

Застосування принципів ООП дозволить відокремити алгоритм знаходження величини завантаженості стрічкового конвеєра, та використовувані ним дані, від функцій керування графічним користувацьким інтерфейсом.

4.2.3 Метод організації вхідних і вихідних даних

Програма отримує вхідне зображення у вигляді багатовимірного масиву цілих чисел, що впорядковані за стовпчиками и рядками відповідних пікселів цифрового зображення.

У процесі перетворення та застосування інших методів обробки зображень у якості проміжних результатів утворюватимуться бінарні зображення такого ж розміру, як і вихідне. Для економії обчислювальних ресурсів, проміжні зображення, отримані у результаті виконання функцій будуть зберігатись до тих пір, поки параметри функції не зміняться.

Вихідні дані у варіанті програми, що розробляються, виводитимуться на екран у вікні програми. Скалярні величини будуть відображені у текстовому вигляді, а геометричні та бінарні фрагменти зображень за допомогою графічних примітивів.

4.2.4 Вибір складу програмних засобів

Для розробки програми була обрана мова програмування Python через відносно менший середній час розробки програм у порівнянні з іншими мовами програмування та наявності широкого спектру модулів та розширень.

Python – інтерпретована мова програмування високого рівня. Вона набуває популярності завдяки різноманіттю пакетів розширення – бібліотек. Найбільш відомі галузі застосування Python: веб-розробка, наукові та числові обчислення, графічний користувальницький інтерфейс, автоматизація розробки та тестування ПЗ, системи електронної комерції.

OpenCV – бібліотека функцій, переважно спрямованих на обробку зображень в реальному часі. Вона реалізовує понад 2500 оптимізованих алгоритмів, що включають як класичні алгоритми комп'ютерного зору, так і алгоритми машинного навчання. Бібліотека може бути застосована для ідентифікації облич, предметів, відстеження руху об'єктів на відео, створення тривимірних моделей зі стереозображень, та для багато чого іншого.

OpenCV надає інтерфейси для мов програмування C++, Python, Java, та пакету MATLAB, та підтримує ОС Windows, Linux, Android та Mac OS. Для підвищення ефективності бібліотека використовує, за наявністю, спеціальні інструкції процесору (SSE та MMX), та паралельні обчислення засобами графічного прискорювача (CUDA, OpenCL).

Бібліотека OpenCV застосовує бібліотеку NumPy для структуризації, зберігання та виконання деяких операцій над цифровими зображеннями у пам'яті при роботі програми.

Бібліотека NumPy додає підтримку багатовимірних масивів та матриць разом з набором високорівневих математичних функцій, що оперують ними. Модуль бібліотеки реалізовано на мові програмування C, що надає обчисленням переваги у швидкості виконання.

4.3 Опис розробленої програми

4.3.1 Загальні відомості

Розроблена програма має назву «Scanner», та її виконавчий файл іменований `scanner.pyw`. Програма може бути запущена шляхом натискання на іконку виконавчого файлу, введення його ім’я у командному рядку як самостійного виконавчого файлу, або як параметру виконавчого файлу інтерпретатора `python`.

Програма написана мовою програмування Python версії 3. Для запуску програми на комп’ютер необхідно встановити інтерпретатор CPython, доступний для безкоштовного завантаження за посиланням `https://www.python.org` . За допомогою утиліти командного рядку `pip` для завантаження пакетів розширення необхідно встановити бібліотеку OpenCV for Python командою:

```
pip install opencv-python
```

Інтерпретатор та бібліотека є кросплатформеними, що дозволяє їм запускатись в операційних системах Windows, Linux і macOS, та у системах з процесорами на базі 32-бітної та 64-бітної архітектури сімейства x86, або ARM архітектури. Дана реалізація програми передбачає запуск у графічному середовищі операційної системи.

4.3.2 Функціональне призначення

Програма призначена для застосування у окремому модулі введення відеоінформації комп’ютерної системи контролю завантаження стрічкових конвеєрів. Вона виконує фільтрацію зображення за обраними алгоритмами, застосовує бінаризацію для відокремлення фігури транспортованого матеріалу від фігури стрічки конвеєра, у обраному чотирикутному регіоні на зображенні малює лінію, перпендикулярну до напрямку потоку матеріалу, та визначає, скільки довжини лінії перетинається з фігурою потоку у відсотковому вираженні.

Функціонування програми неможливе, якщо вхідне зображення є сильно зашумленим, або недостатньо контрастним, що може виникати через некоректне налаштування цифрової камери чи розташування її у місці з неналежним

освітленням. Занадто малі зображення, або зображення із занадто віддаленими об’єктами можуть призводити до погіршення точності.

4.3.3 Опис логічної структури програми

Згідно з принципом модульності, що застосовується під час розробки системи, та парадигми ООП, функціонал програми поділений на класи, структура яких відображена на рисунку Рисунок 4.1.

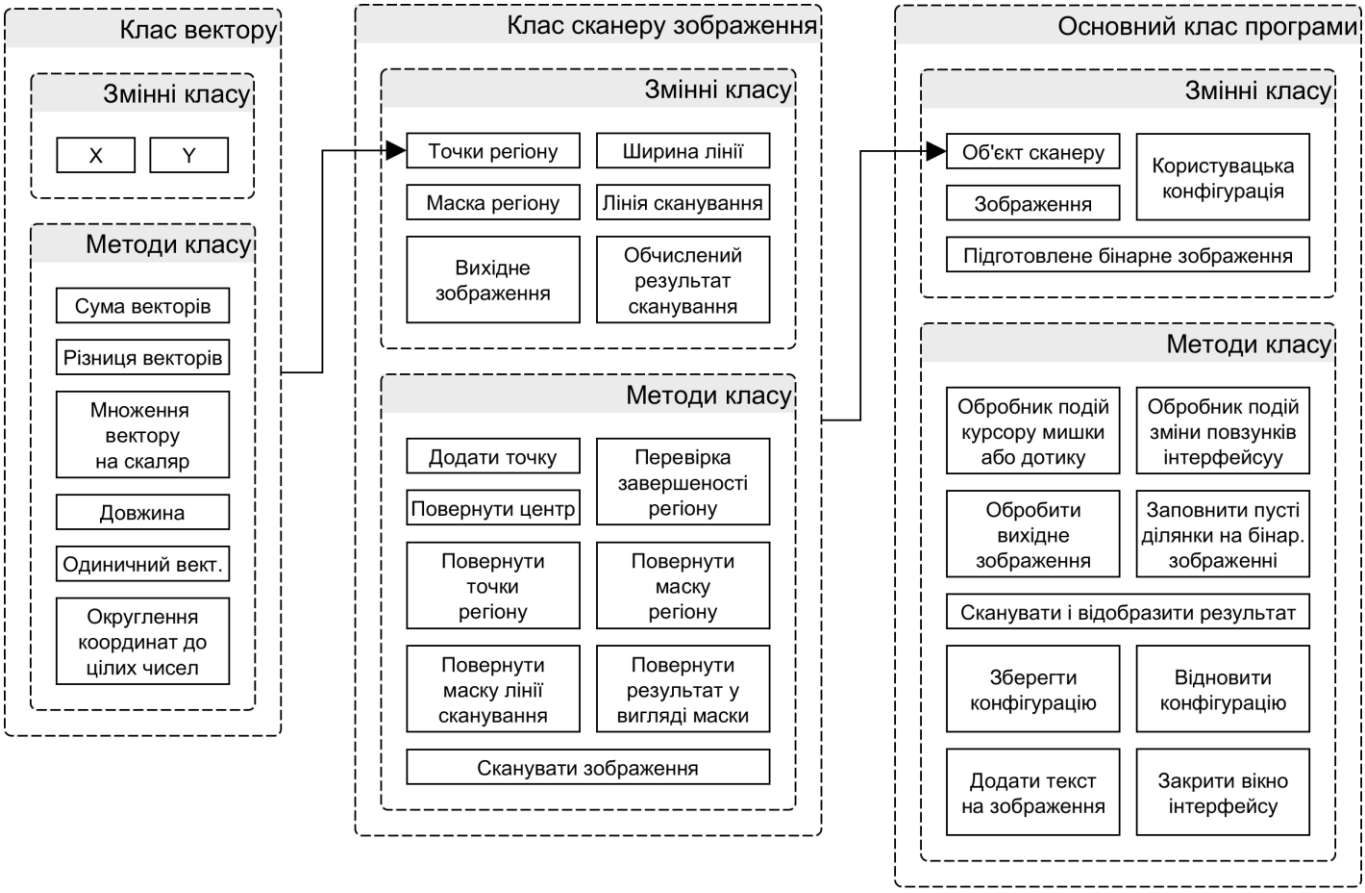

Рисунок 4.1 – Структура класів розробленої програми

Клас вектору призначений для обчислення координат перпендикуляру до напрямку вантажопотоку на стрічці конвеєра. Він реалізує функції деяких простих операцій, що дозволяє застосовувати об’єкти цього класу у виразах з використанням звичних символів «+, -, *, /».

Клас сканеру зображення призначений для обчислення величини завантаження стрічки конвеєра. Він містить функції з побудови перпендикулярної

до потоку лінії всередині регіону, створення різноманітних бінарних масок для побітових операцій над зображеннями, та саму функцію з обчислення величини завантаженості стрічки конвеєра. Клас зберігає проміжні результати обчислень за принципом кешу для економії процесорного часу. Він стирає кеш та виконує обчислення заново, якщо регіон був замінений на новий.

Основний клас програми відповідає за створення вікна програми, та реагує на взаємодію користувача з віджетами для налаштування параметрів. Він також виконує послідовне застосування досліджуваних алгоритмів та фільтрів з області комп'ютерного зору перед тим, як передати його до об'єкта класу сканера для обчислення величини завантаженості.

4.3.4 Використовувані технічні засоби

Для функціонування у складі комп'ютерної системи програма застосовує цифрову відеокамеру, підключену до комп'ютера, хоча у розробленому демонстраційному прикладі використовується зображення, що загружається з графічного файлу.

Тестування програми на ПК з архітектурою процесору x86_64, тактовою частотою 2.3 ГГц показали, що обсяг використання програмою ресурсів процесора не перевищує 20% при встановленні мінімальної затримки між ітераціями тривалістю 1 мс, а обсяг використання оперативної пам'яті коливається на рівні декількох десятків мегабайт. Фактично, затримка отримання фотознімку з пристрою цифрової камери становитиме не менш ніж 16 мс у режимі 60 кадрів/с, під час якої програма не використовує ресурс процесора.

4.3.5 Цикл роботи програми

Як зазначалось раніше, функціонування комп'ютерної системи має періодичний характер, тому програма містить основний цикл, що виконує необхідні операції впродовж обмеженого інтервалу часу. Повний цикл роботи програми

складається з етапу ініціалізації, основного циклу, та завершення. Більш детально процеси, виконувані програмою, описані в схемі на рисунку Рисунок 4.2.

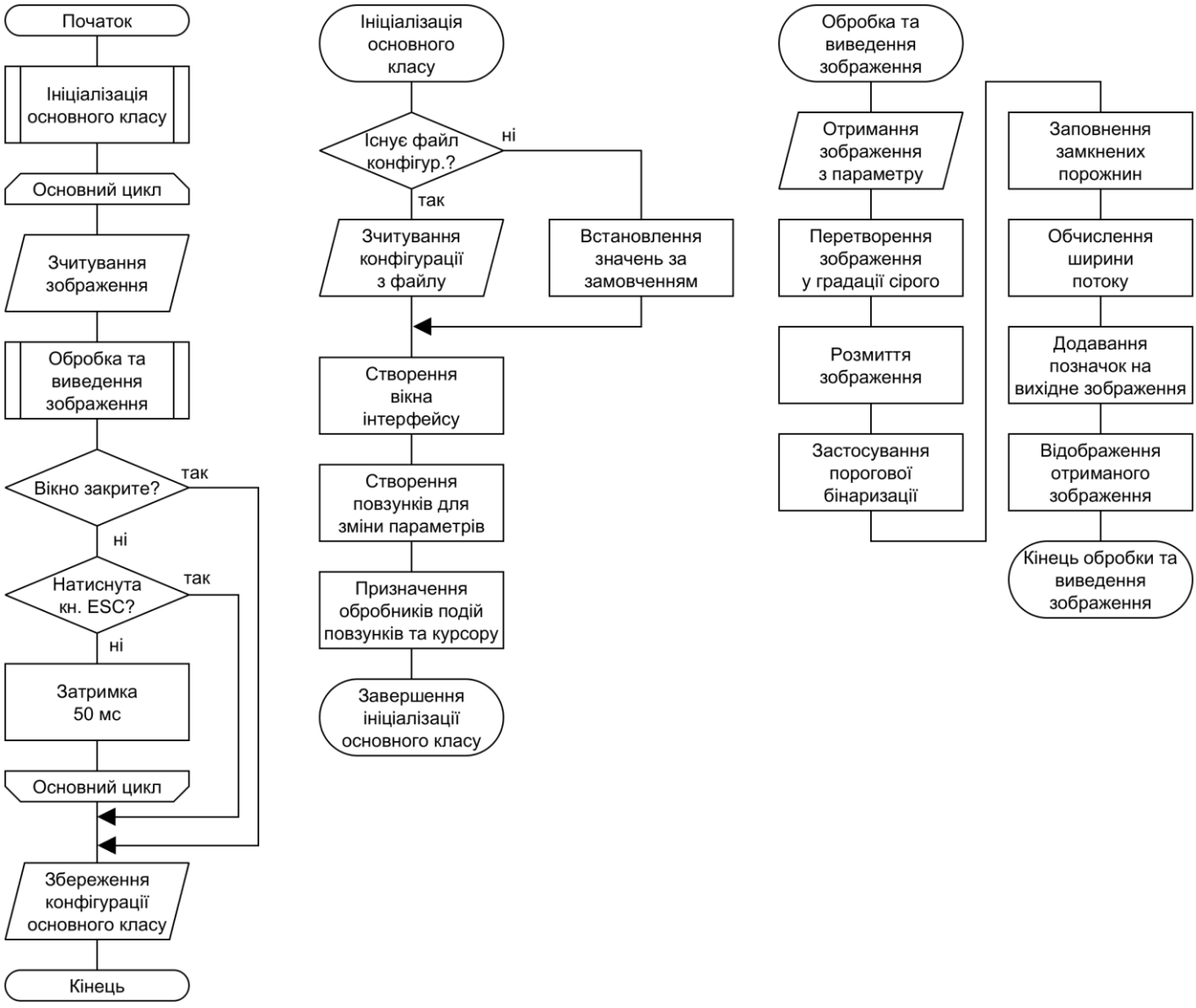

Рисунок 4.2 – Схема програми та підпрограм основних методів класу

Як видно зі схеми програми та відомо з опису логічної структури, всі алгоритми комп’ютерного зору з обробки зображення реалізовані у методах основного класу програми.

Методи з обчислення ширини потоку матеріалу виділені у окремий клас, та не відображені на наведеній схемі. Їх сутність полягає у знаходженні площі перетину фігур лінії сканування та потоку матеріалу, та обчисленні відсоткового відношення величини площі перетину до площі лінії. Ідентифікація фігури потоку здійснюється

з використанням методом заливки (floodfill), вихідною точкою для якої є центр чотирикутнику регіону.

4.3.6 Вхідні та вихідні дані

При роботі з зображенням, отриманим з відеопристрою, у програмі, що розробляється, воно представляється у вигляді масиву X на Y на Z цілих величин від 0 до 255, де X та Y – висота та ширина зображення у пікселях, а Z – кількість колірних компонент, тобто 3 для колірної системи RGB. Тип величин масиву являє собою беззнакове ціле розміром 1 байт.

Користувацькі параметри являють собою наступну структуру даних:

- Чотири пари координат досліджуваного регіону на зображенні, що є невід'ємними цілими числами
- Величина порогу бінаризації, що є цілим значенням та змінюється від 0 до 255
- Розмір ядра для рівномірного розмиття у пікселях, ціле непарне невід'ємне значення.

Вихідна величина, що обчислюється програмою та характеризує ступінь завантаження стрічки конвеєра, є натуральною величиною, що приймає значення в діапазоні від 0 до 1.

4.4 Висновки по розділу

У розділі були сформульовані вимоги до програмного забезпечення комп'ютерної системи, з урахуванням принципу модульності запропонована структура із застосуванням парадигми ООП, описані методи організації даних при роботі програми, обрано склад програмних засобів, що будуть використані при реалізації алгоритму.

Було розроблене програмне забезпечення системи мовою програмування Python з використанням кросплатформенної бібліотеки OpenCV for Python. У описі програми представлена її функціональне призначення, логічна структура та схема

алгоритму програми. Визначено види та обсяги вхідних і вихідних даних, проаналізовано використання ресурсів системи.

Невеликий обсяг вихідних даних дозволяє використовувати CAN мережу шахти для їх передачі на центральний вузол системи, а застосування кешу зменшує кількість операцій в одиницю часу, що є важливим при використанні менш продуктивного процесору одноплатного комп’ютера у порівнянні з процесором ПК.

5 ЕКСПЕРИМЕНТАЛЬНИЙ РОЗДІЛ

5.1 Експеримент з аналізу роботи і оцінки результату досліджуваних алгоритмів з обробки зображень

5.1.1 Мета і умови експерименту

Мета експерименту полягає у встановленні здатності алгоритмів, методів обробки цифрових зображень з області комп'ютерного зору, або їх комбінацій, до виявлення чи відокремлення об'єктів вантажопотоку та стрічки на зображенні реального стрічкового конвеєру, та оцінки точності і доцільності використання такого алгоритму у комп'ютерній системі.

Для здійснення експерименту розроблена програма, яка по черзі застосовує пропоновані алгоритми і методи до визначеного набору зображень, та дозволяє регулювати їх параметри. Експериментатор має оцінити результати застосування алгоритмів за критеріями, наведеними у методиці експерименту, та зробити висновки.

5.1.2 Методика експерименту

Згідно поставленої задачі дослідження та запропонованого напрямку її вирішення, необхідно знайти алгоритм перетворення вхідного зображення у градаціях сірого у вихідне бінарне зображення, на якому контури фігур білого та чорного кольору будуть найбільш точно співпадати з контурами стрічки конвеєра та вантажопотоку на оригінальному зображенні.

Досліджувані алгоритми можна розділити на дві категорії: ті, що перетворюють вхідне зображення у двійкове для визначення контурів фігур, та ті, що змінюють зображення з метою підвищення якості розпізнавання перших (наприклад, вирівнюють контраст, фільтрують та ін.).

Не існує універсальних критеріїв оцінки алгоритмів, тому що доцільність і ефективність їх використання розглядається відносно поставленої задачі та

доступних ресурсів. Для оцінки досліджуваних алгоритмів в умовах поточного дослідження були сформульовані та пропонуються наступні критерії.

Надійність алгоритму підготовки зображення до виявлення фігур:

- Алгоритм має високу якість розпізнавання фігур в більшості випадків, та не потребує зміни конфігурації програми під час роботи ***-2 бали***;
- Алгоритм має невисоку якість розпізнавання фігур в деяких випадках, що може бути усунене налаштуванням в автоматичному або ручному режимі – ***1 бал***;
- Алгоритм має низьку якість розпізнавання фігур, та в деяких випадках конфліктує з іншими алгоритмами – ***0 балів***.

Надійність алгоритму виявлення фігур:

- Алгоритм здатен однозначно розділяти фігури вантажопотоку і стрічки конвеєра в більшості випадків – ***2 бали***;
- Алгоритм здатен розділяти фігури, та допускає похибки, що можуть бути усунені в більшості випадків, або успішно розрізняє одну з фігур – ***1 бал***;
- Алгоритм допускає похибки при визначенні фігур, що не можуть бути усунені в більшості випадків – ***0 балів***.

Універсальність алгоритмів:

- Алгоритм може бути застосований без зміни конфігурації в більшості випадків за нормальних умов – ***1 бал***;
- Алгоритм потребує конфігурації невеликої кількості параметрів, які не потребують значних змін під час роботи системи – ***0 балів***;
- Алгоритм неможливо конфігурувати для роботи в заданих умовах, або він потребує частої періодичної зміни конфігурації – ***мінус 1 бал***.

Ефективність виконання алгоритму комп'ютерною системою:

- Виконання алгоритму процесором займає незначну кількість часу при будь-якій конфігурації – ***1 бал***;

- Виконання алгоритму процесором може потребувати більше часу чи обчислювальних ресурсів за деяких налаштувань, але результат та час виконання є задовільними з використовуваними параметрами – ***0 балів***;
- Алгоритм може потребувати багато часу для обчислення, або не є достатньо ефективним в умовах даної конфігурації системи – ***мінус 1 бал***.

Запропонований варіант оцінювання не є вирішальним, та під час розробки комп'ютерної системи необхідно враховувати особливості алгоритму та його реалізації.

5.1.3 Експериментальний аналіз і оцінка алгоритмів

У таблиці Таблиця 5.1 приведено результат оцінки алгоритмів у ході експерименту за визначеною методикою. Більш детальний аналіз приведено у теоретичному розділі.

Таблиця 5.1 – оцінка алгоритмів

	Надійність	Універсальність	Ефективність	Сумарно
Алгоритми покращення якості розпізнавання фігур				
Вирівнювання гістограми	0	1	1	2
Адаптивне вирівнювання гістограми	1	0	1	2
Рівномірне згладжування	2	0	1	3
Білатеральна фільтрація	2	0	-1	1
Пошук і заповнення пустих замкнених регіонів	2	1		3
Алгоритми розпізнавання фігур				
Бінаризація за порогом	1	0	1	2
Бінаризація за методом Оцу	0	-1	1	0
Адаптивна бінаризація	2	0	0	2
Алгоритм Кенні для виявлення границь	0	0	1	1
Перетворення Хафа для пошуку прямих	0	-1	0	-1

5.1.4 Аналіз результатів експерименту

За результатами оцінювання роботи алгоритмів рекомендоване застосування алгоритмів рівномірного згладжування, заповнення пустих замкнених регіонів та бінаризації за порогом. Комп'ютерна система у такому разі матиме достатню надійність обчислення величини завантаження стрічкового конвеєра, не потребуватиме регулярних налаштувань та відповідатиме вимогам до періодичності.

5.2 Тестування розробленого програмного забезпечення

Для перевірки функціонування програми, як і в попередньому експерименті, використовується набір фотографій конвеєрів, що транспортують вугілля. Згідно вимогам застосування комп'ютерної системи, ці зображення мають бути чіткими та рівномірно освітленими.

Сутність перевірки полягає в використанні фотографій тих ділянок конвеєру, де матеріал розподіляється рівномірно. Виконується попередня конфігурація фільтру розмиття та величини порогу, після чого у довільних ділянках конвеєра на фотографії розміщується регіон, що контролюється системою. Величина завантаженості стрічки, обчислена програмою у декількох регіонах вздовж рівномірно розподіленого потоку вугілля, буде приблизно однаковою.

Допустиме деяке незначне коливання величини через нерівномірність границі потоку кускового вугілля, та через людський фактор при налаштуванні вершин регіонів користувачем програми.

На рисунках Рисунок 5.1 та Рисунок 5.2 представлені знімки вікна програми, під час виконання якої був тричі змінений досліджуваний регіон. Результати обчислення ширини потоку матеріалу: 59.27%, 56.84%, 58.81% та 57.23%. Середнє арифметичне цих величин складає 58.04%, а максимальне відхилення з отриманих значень від середнього становить всього лише 1.23%.

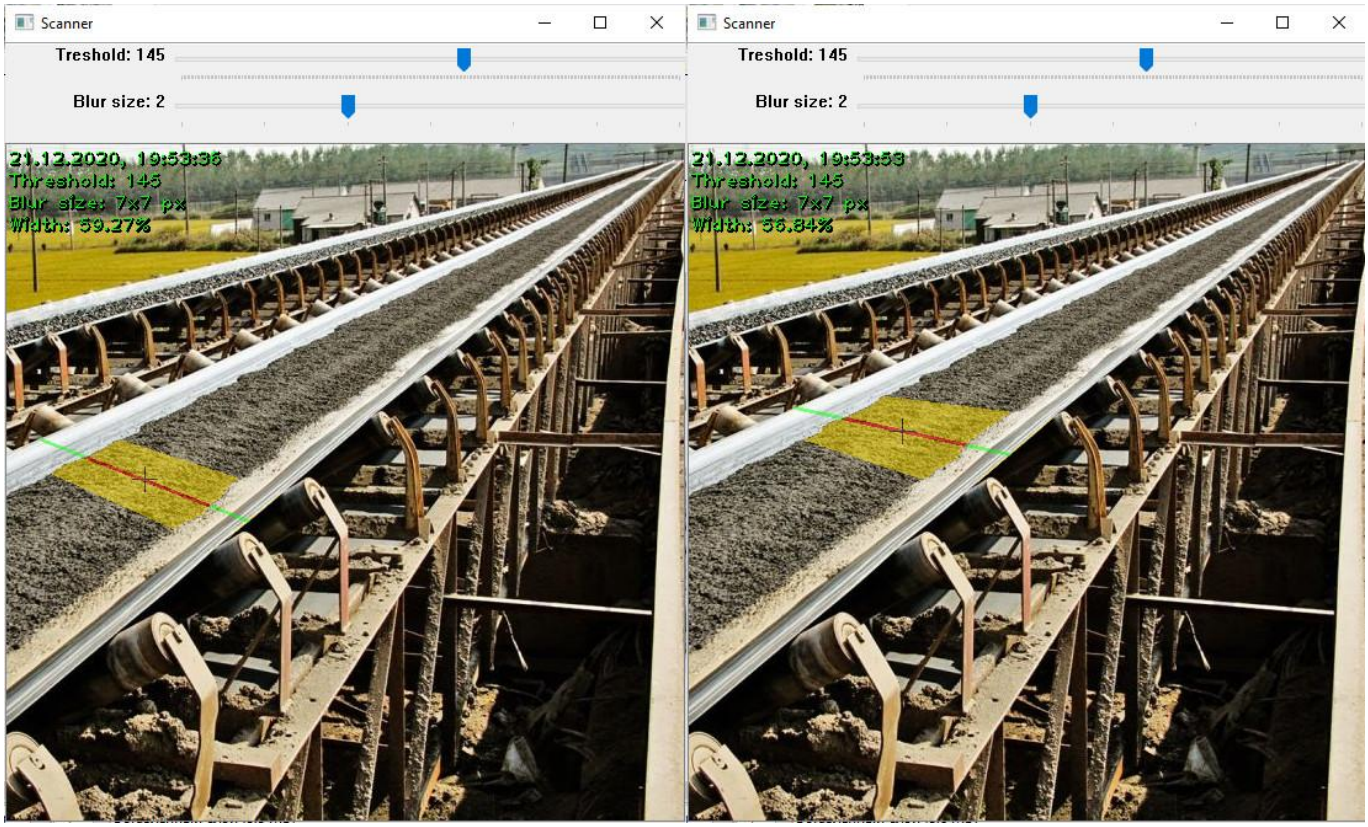

Рисунок 5.1 – Знімки вікна програми, отримані в ході експерименту

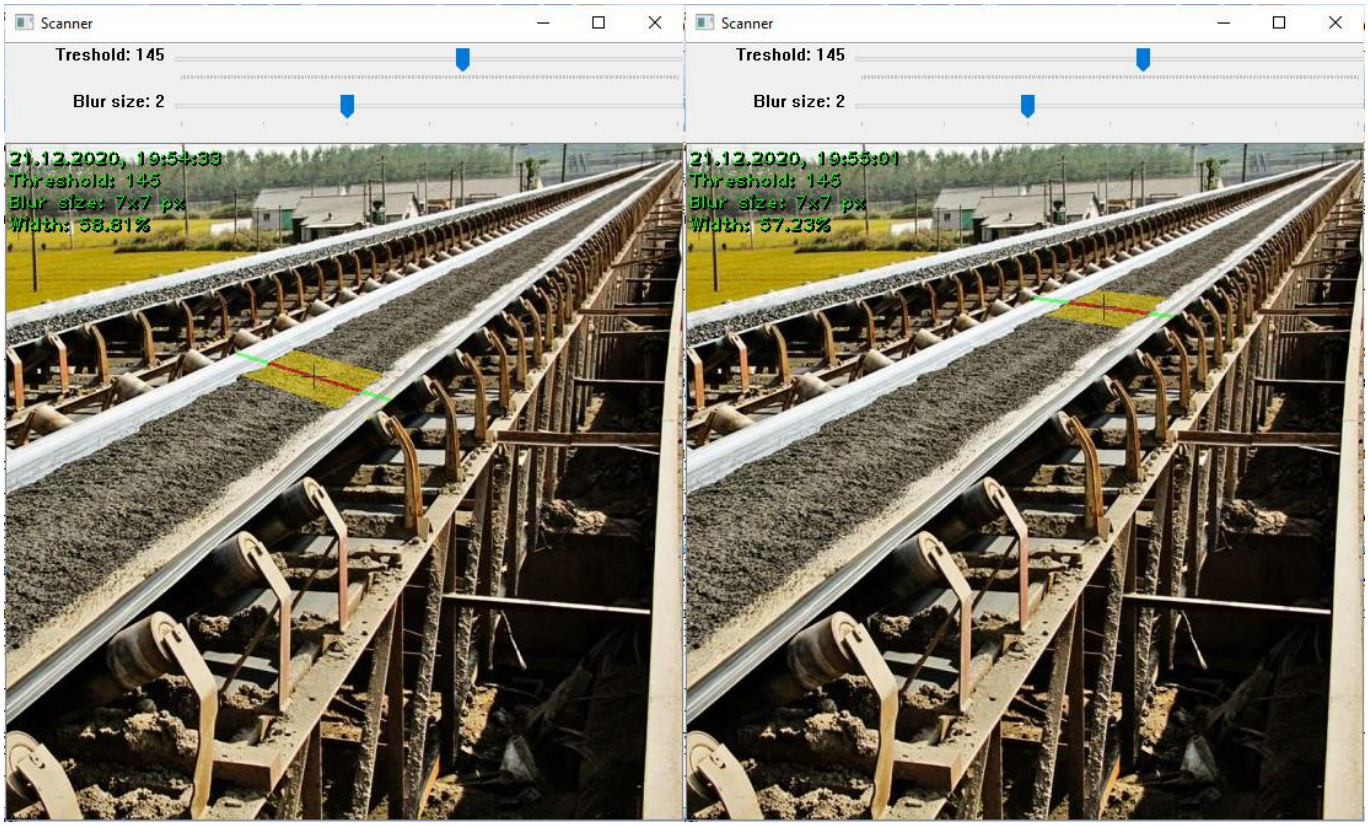

Рисунок 5.2 – Знімки вікна програми, отримані в ході експерименту, продовження

5.3 Висновки по розділу

У даному розділі описані методики і результати експериментів, що проводились в ході дослідження та розробки комп’ютерної системи.

Експеримент з аналізу роботи та оцінки результату досліджуваних алгоритмів має мету аналітичного обґрунтування вибору алгоритмів для застосування під час розробки програмного забезпечення комп’ютерній системи. Були сформульовані критерії та категорії досліджуваних алгоритмів, що є важливими для конфігурації та умов функціонування комп’ютерної системи, призначені умовні оцінки.

Під час взаємодії з алгоритмами у експериментальній програмі шляхом регулювання параметрів та застосування алгоритмів до різних цифрових фотографій стрічкових конвеєрів, за відповідними категоріями алгоритмам були присвоєні бали. На основі суми накопичених балів для реалізації програмного забезпечення було запропоновано використовувати алгоритми рівномірного згладжування, заповнення пустих замкнених регіонів та бінаризації за порогом. Більш детальний розбір алгоритмів наведено у теоретичній частині.

В ході розробки програмного забезпечення проводилось тестування обчислених ним результатів величини завантаженості стрічки. На одній й тій самій фотографії конвеєра з рівномірно розподіленим вугіллям змінювали регіон, у якому програма виконувала контроль. Отримані результати відхилялись не більше ніж на 2% від їх середнього значення, що дозволяє стверджувати про достатньо високу точність обчислень для застосування у складі інших автоматизованих систем керування.

ВИСНОВКИ

Кваліфікаційна робота є завершеною науковою роботою, в якій вирішена науково-практична задача визначення завантаженості вугіллям окремої ділянки шахтного стрічкового конвеєра, використовуючи алгоритми та методи обробки з області комп'ютерного зору в комп'ютерній системі контролю завантаження стрічки конвеєра.

Основні висновки і результати роботи полягають у наступному:

1. Досліджені існуючі підходи до побудови систем контролю завантаження стрічки конвеєра, показана відсутність моделі системи контролю без застосування джерела лінійного світла, обґрунтована доцільність розробки й застосування такої системи.
2. Проаналізовано алгоритми обробки зображень з області комп'ютерного зору, визначено їх ефективність, доцільність використання у комп'ютерній системі, сформульовано критерії та обґрунтований вибір конкретних алгоритмів для впровадження в програмному забезпеченні системи. З цією метою розроблено програму для дослідження їх застосування до існуючих зображень для.
3. Спроектовано структурну модель комп'ютерної системи з урахуванням особливостей, типу і пропускної спроможності каналу передачі даних існуючої інформаційної мережі шахти.
4. Описано функціональну модель застосування досліджуваних алгоритмів комп'ютерного зору до вхідного зображення для отримання вихідної величини завантаженості конвеєра. Виділені структурні модулі програми, показана їх взаємодія в алгоритмі виконання програми
5. Розроблено програмне забезпечення комп'ютерної системи контролю завантаження стрічки конвеєра, описано її структуру, області і умови її застосування, можливості і обмеження. Виконано експериментальну перевірку коректності та точності її функціонування в заданих умовах.

ПЕРЕЛІК ПОСИЛАНЬ

1. Мінеральні ресурси України та світу на 01.01.2008 р. – Київ: Державне науково-виробниче підприємство «Геоінформ України», 2009. – 602 с.

2. «Пріоритети та важелі модернізації вугільної галузі в Україні». Аналітична записка. А.В. Шевченко, С.Л. Воробйов, http://www.niss.gov.ua/articles/1495/

3. Анализ влияния регулирования скорости конвейерной ленты на энергоэффективность транспортных систем, Ю.Т. Разумный, В.Н. Прокуда, УДК 621.316.7:622.647.2

4. Посібник з курсу лекцій "Основи гірничого виробництва" під редакцією доц. Носача О.К., ДонНТУ, 2011.

5.Енергоефективність магістрального конвеєрного транспорту вугільних шахт [Електронний ресурс]: монографія / Ю.Т. Разумний , В.М. Прокуда ; М-во освіти і науки України, Нац. гірн. ун-т. – Електрон. текст. дані. – Дніпро: НГУ, 2018. – 120 с.

6. Прокуда В.Н. Исследование и оценка грузопотоков на магистральном конвейерном транспорте ПСП «Шахта «Павлоградская» ПАО ДТЭК «Павлоградуголь» / В.Н. Прокуда, Ю.А. Мишанский, С.Н. Проценко // Гірнича електромеханіка. – 2012. – № 88. – С. 107-111.

7. Machine vision methods for estimation of size distribution of aggregate transported on conveyor belts Michał Kontny Silesian University of Technology, Gliwice, Poland E-mail:michal.kontny@polsl.pl Received 18 September 2017; accepted 19 September 2017 DOI https://doi.org/10.21595/vp.2017.1915

Додаток А

Текст программи контролю завантаження стрічки конвеєра

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ КОНТРОЛЮ ЗАВАНТАЖЕННЯ СТРІЧКИ КОНВЕЄРА

Текст програми

804.02070743.20002-01 12 01

Аркушів 9

2020

АНОТАЦІЯ

Даний документ містить вихідний код єдиного файлу `scanner.pyw` програми контролю завантаження стрічки конвеєра.

Текст програми реалізований на мові програмування Python 3.

Середовище розробки та налагодження − IDLE.

```
import cv2 as cv
import numpy as np
import json
from datetime import datetime
from numbers import Number
from pathlib import Path
from tkinter import Tk
from tkinter.filedialog import askopenfilename

'''Vector with (x, y) coordinates class.
A simple implementation of vector or point in Cartesian 2-dimentional
space with minimal required arithmetics
'''
class Vector:
    def __init__(self, x, y):
        self.x_y = x, y

    def __repr__(self):
        return f'<Vector {self.x_y}>'

    def __add__(self, other):
        assert isinstance(other, Vector)
        x1, y1 = self.x_y
        x2, y2 = other.x_y
        return Vector(x1+x2, y1+y2)

    def __neg__(self):
        x1, y1 = self.x_y
        return Vector(-x1, -y1)

    def __sub__(self, other):
        assert isinstance(other, Vector)
        return self + (-other)

    def __mul__(self, scalar):
        assert isinstance(scalar, Number), \
            'Only multiplication by scalar is supported'
        x1, y1 = self.x_y
        return Vector(x1*scalar, y1*scalar)

    def __truediv__(self, scalar):
        assert isinstance(scalar, Number), \
            'Only division by scalar is supported'
        assert scalar != 0, 'Division by zero'
        return self * (1/scalar)

    def len(self):
        x1, y1 = self.x_y
        return np.sqrt(x1**2 + y1**2)

    def normalized(self):
        return self / self.len()

    def rounded(self):
        x1, y1 = self.x_y
        return Vector(round(x1), round(y1))
```

```
'''

Creates a scan line from the RegionBuilder, then
Class calculates and stores a region mask,
a scan line mask, and a central point
'''
class Scanner:
    def __init__(self, scanline_thickness):
        self._points = []
        self._mask_region = None
        self._mask_scanline = None
        self._mask_floodfill = None
        self._mask_found = None
        self._scanline_thickness = scanline_thickness
        self._scanline_area = None
        self._image = None

    def isComplete(self):
        return len(self._points) == 4

    def addPoint(self, x, y):
        if len(self._points) < 4:
            self._points.append(Vector(x, y))
        else:   # Start again
            self._points = [Vector(x, y)]
            self._mask_region = None
            self._mask_scanline = None
            self._mask_floodfill = None
            self._mask_found = None
            self._scanline_area = None

    def getCenter(self):
        assert self.isComplete()
        p1, p2, p3, p4 = self._points
        center = ((p1 + p2+ p3 + p4) / 4).rounded()
        return center.x_y

    def getPoints(self):
        return [point.x_y for point in self._points]

    def getRegionMask(self):
        if type(self._mask_region) is np.ndarray:
            return self._mask_region.copy()

        mask = np.full(self._image.shape, 0, np.uint8)
        contour = np.array([self.getPoints()])
        cv.fillConvexPoly(mask, contour, 255)

        self._mask_region = mask
        return mask.copy()

    def getScanlineMaskPoints(self):
        assert self.isComplete()
        p1, p2, p3, p4 = self._points
        SL_p1 = ((p1+p2) / 2).rounded()
```

```
    SL_p2 = ((p3+p4) / 2).rounded()
    return SL_p1.x_y, SL_p2.x_y

def getScanlineMask(self):
    if type(self._mask_scanline) is np.ndarray:
        return self._mask_scanline.copy()

    mask = np.full(self._image.shape, 0, np.uint8)
    point1, point2 = self.getScanlineMaskPoints()
    thickness = self._scanline_thickness
    cv.line(mask, point1, point2, 255, thickness)

    self._scanline_area = np.count_nonzero(mask)

    self._mask_scanline = mask
    return mask.copy()

def getFloodfillMask(self):
    if type(self._mask_floodfill) is np.ndarray:
        return self._mask_floodfill.copy()

    mask = self.getScanlineMask()
    x, y = mask.shape

    # Add extra 1 pixel border to mask for floodfill function
    mask = np.r_[np.zeros([1, y], np.uint8),
                 mask,
                 np.zeros([1, y], np.uint8)]

    mask = np.c_[np.zeros([x+2, 1], np.uint8),
                 mask,
                 np.zeros([x+2, 1], np.uint8)]

    # Mask for floodfill function is inverted
    ffmask = 255 - mask

    self._mask_floodfill = ffmask
    return ffmask.copy()

def getFoundAsBw(self):
    return cv.bitwise_and(self._mask_found[1:-1, 1:-1], self._mask_scanline)

def scanImage(self, image_bin):
    self._image = image_bin
    center = self.getCenter()
    x, y = center
    ffmask = self.getFloodfillMask()
    ff_flags = cv.FLOODFILL_MASK_ONLY | (255<<8)
    area = 0

    if image_bin[y, x] == 255:  # Point hits detected area
        area,image,ffmask,rect = cv.floodFill(image_bin, ffmask, center, 0,
                                              flags=ff_flags)

    self._mask_found = ffmask
```

```
        if area > 0:
            ratio = area / self._scanline_area
        else:
            ratio = 0

        return ratio

class MainWindow:
    def __init__(self, *, threshold=128, blur_size=5, scanline_thickness=5,
                 window_name='Scanner'):
        self._scanner = Scanner(scanline_thickness)
        self._winname = window_name
        self._image = None
        self._image_proc = None
        self._threshold = threshold
        self._blur_size = blur_size

        self.loadSettings()

        (w, h), baseline = cv.getTextSize('Gg', cv.FONT_HERSHEY_PLAIN, 1, 1)
        self._text_height = h + baseline + 2

        cv.namedWindow(window_name)
        cv.createTrackbar('Treshold', window_name, self._threshold,
                          256, self.onTrackbarChange)
        cv.createTrackbar('Blur size', window_name, (self._blur_size//2) - 1,
                          6, self.onTrackbarChange)
        cv.setMouseCallback(window_name, self.onMouseEvent)

    def onMouseEvent(self, event, x, y, flags, param):
        scanner = self._scanner

        if event is cv.EVENT_LBUTTONDOWN:
            scanner.addPoint(x, y)
            self.scanAndDisplay()

    def onTrackbarChange(self, _):
        self._threshold = cv.getTrackbarPos('Treshold', self._winname)
        self._blur_size = 3 + 2*cv.getTrackbarPos('Blur size', self._winname)
        self.processImage(self._image)

    def scanAndDisplay(self):
        image = self._image.copy()
        scanner = self._scanner

        date = datetime.now().strftime('%d.%m.%Y, %H:%M:%S')
        thrs = 'Threshold: {}'.format(self._threshold)
        blur = 'Blur size: {0}x{0} px'.format(self._blur_size)

        if scanner.isComplete():
            image2 = image.copy()
            image_proc = self._image_proc

            ratio = scanner.scanImage(image_proc)

            proc_region = cv.bitwise_and(image_proc, scanner.getRegionMask())
```

```
            image2[proc_region.nonzero()] = [0,224,255]
            image2[scanner.getScanlineMask().nonzero()] = [0,255,0]
            image2[scanner.getFoundAsBw().nonzero()] = [0,0,255]
            cv.drawMarker(image2, scanner.getCenter(), [0,0,0])

            image = cv.addWeighted(image, 0.5, image2, 0.5, 1)

            width = 'Width: {}%'.format(round(ratio*100,2))
            self.putText(image, width, 3)

            cv.imshow(self._winname, image)
        else:
            for point in self._scanner.getPoints():
                cv.drawMarker(image, point, [0, 0, 255])

        self.putText(image, date, 0)
        self.putText(image, thrs, 1)
        self.putText(image, blur, 2)
        cv.imshow(self._winname, image)

    def processImage(self, image):
        image_gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
        image_blurred = cv.blur(image_gray, (self._blur_size,)*2)
        t, image_bin = cv.threshold(image_blurred, self._threshold, 255,
                                    cv.THRESH_BINARY)
        image_bin_inv = 255 - image_bin
        image_filled = self.fillHoles(image_bin_inv)

        self._image = image
        self._image_proc = image_filled

        self.scanAndDisplay()

    def fillHoles(self, image_bin):
        contours, hierarchy = cv.findContours(image_bin, cv.RETR_TREE,
                                              cv.CHAIN_APPROX_SIMPLE)

        if (contours):
            for contour, c_info in zip(contours, hierarchy[0]):
                next_, prev, child, parent = c_info
                if parent != -1:    # is not external contour
                    cv.fillPoly(image_bin, [contour], 255)

        return image_bin

    def putText(self, image, text, n_line):
        text_y = (n_line + 1)*self._text_height - 2
        cv.putText(image, text, (2, text_y+1), cv.FONT_HERSHEY_PLAIN,
                   1, [0,0,0], 2)
        cv.putText(image, text, (1, text_y), cv.FONT_HERSHEY_PLAIN,
                   1, [0,255,0], 1)

    def saveSettings(self):
        settings = {}
        settings['threshold'] = self._threshold
```

```
        settings['blur_size'] = self._blur_size
        settings['points'] = self._scanner.getPoints()

        path = Path('scanner_settings.json')
        with path.open('w') as file:
            json.dump(settings, file)

    def loadSettings(self):
        path = Path('scanner_settings.json')

        if not path.exists():
            return

        with path.open('r') as file:
            settings = json.load(file)
            self._threshold = settings['threshold']
            self._blur_size = settings['blur_size']
            for point in settings['points']:
                self._scanner.addPoint(*point)

    def destroy(self):
        cv.destroyWindow(self._winname)


def draw_region():
    global canvas, scanner

    canvas2 = canvas.copy()
    if scanner.isComplete():
        width = round(scanner.scanImage(canvas2), 2)

        canvas2[scanner.getScanlineMask().nonzero()] = 128

        cv.setWindowTitle('Canvas', f'Canvas, {width*100}% width')
    else:
        for point in scanner.getPoints():
            cv.drawMarker(canvas2, point, 255)
        cv.setWindowTitle('Canvas', 'Canvas')
    cv.imshow('Canvas', canvas2)

def mouse_callback(event, x, y, flags, param):
    global scanner

    if event is cv.EVENT_LBUTTONDOWN:
        scanner.addPoint(x, y)
        draw_region()


if __name__ == '__main__':
    # Use Tcl/Tk file dialog to ask for image file path
    tk_mainwindow = Tk()
    tk_mainwindow.iconify()
    tk_mainwindow.title('Open image')
    path = Path(askopenfilename())
    tk_mainwindow.destroy()
    Conv = cv.imread(str(path))
```

```
scanner = MainWindow(scanline_thickness=2)

while True:
    scanner.processImage(Conv)

    # Loop until <ESC> pressed or window closed
    if (cv.waitKey(50) == 27 or
        cv.getWindowProperty('Scanner', cv.WND_PROP_FULLSCREEN) == -1):
        break

scanner.saveSettings()
scanner.destroy()
```

Додаток Б

Текст программи для проведення експерименту з аналізу і оцінки алгоритмів обробки зображень з області комп’ютерного зору

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ
ДЛЯ АНАЛІЗУ І ОЦІНКИ АЛГОРИТМІВ ОБРОБКИ ЗОБРАЖЕНЬ

Текст програми

804.02070743.20002-01 12 01

Аркушів 8

2020

АНОТАЦІЯ

Даний документ містить вихідний код файлу `experiment.py` програми для аналізу і оцінки алгоритмів обробки зображень з області комп’ютерного зору.

Текст програми реалізований на мові програмування Python 3.

Середовище розробки та налагодження − IDLE.

```
import cv2 as cv
import numpy as np
import sys

'''Abstract (example) class for method'''
class Method:
    def __init__(self):
        self.info = ''

    '''Overwrites initial settings, has to be overriden'''
    def setup(self):
        self.title = 'Method'
        self.settings = {'Value': 10}   # Name and maximum value

    '''Implements algoritm, has to be overriden'''
    def transform(self, image):
        return image

    def run_experiment(self, image):
        self.setup()

        title = self.title
        self.image = image

        self._create_window()
        self._update()

        while True:
            keycode = cv.waitKey(200)
            key_esc = (keycode == 27)
            key_q = (keycode == ord('q'))
            closed = (cv.getWindowProperty(title, cv.WND_PROP_FULLSCREEN) == -1)

            if closed or key_esc:
                self._destroy_window()
                break

            if key_q:
                self._destroy_window()
                sys.exit()  # close program completely

    def set_info(self, values):
        self.info = ', '.join([n + ' = ' + repr(v) for n, v in values.items()])

    def _create_window(self):
        title = self.title
        image = self.image

        cv.imshow(title, np.hstack([image, image])) # Placeholder

        for name, limit in self.settings.items():
            cv.createTrackbar(name, title, limit//2, limit, self._update)

    def _destroy_window(self):
        self.image = None
        cv.destroyWindow(self.title)
```

```
    def _update(self, val=None):
        title = self.title
        image = self.image

        for name, _ in self.settings.items():
            self.settings[name] = cv.getTrackbarPos(name, title)

        transformed = self.transform(image)
        result = np.hstack([image, transformed])
        cv.imshow(title, result)

        if self.info:
            cv.setWindowTitle(title, title + ': ' + self.info)


'''Histogram equalization'''
class HistEq(Method):
    def setup(self):
        self.title = 'Histogram equalization'
        self.settings = {}

    def transform(self, image):
        image_eq = cv.equalizeHist(image)
        return image_eq


'''Adaptive histogram equalization'''
class AdaptHistEq(Method):
    def setup(self):
        self.title = 'Adaptive histogram equalization'
        self.settings = {'Clip limit': 49,
                         'Grid size': 31}

    def transform(self, image):
        clip = 0.1 + 0.1*self.settings['Clip limit']
        grid_size = (1 + self.settings['Grid size'],) * 2

        clahe = cv.createCLAHE(clip, grid_size)
        image_eq = clahe.apply(image)

        self.set_info({'Clip limit': round(clip, 3), 'Grid size': grid_size})
        return image_eq


'''Binarization with threshold'''
class Binarization(Method):
    def setup(self):
        self.title = 'Binarization'
        self.settings = {'Threshold': 255}

    def transform(self, image):
        threshold = self.settings['Threshold']
        t, image_bw = cv.threshold(image, threshold, 255, cv.THRESH_BINARY)
        return image_bw
```

```
'''Binarization using threshold determined by Otsu method'''
class BinarizationOtsu(Method):
    def setup(self):
        self.title = 'Binarization using Otsu method for threshold'
        self.settings = {}

    def transform(self, image):
        t, image_bw = cv.threshold(image, 0, 255,
                                   cv.THRESH_BINARY | cv.THRESH_OTSU)

        self.set_info({'threshold': t})
        return image_bw


'''Adaptive binarization'''
class AdaptiveBinarization(Method):
    def setup(self):
        self.title = 'Adaptive binarization'
        self.settings = {'Block size': 98,
                         'Constant': 50}

    def transform(self, image):
        block_size = 3 + 2*self.settings['Block size']
        constant = self.settings['Constant'] - 25
        image_abw = cv.adaptiveThreshold(image, 255,
                                         cv.ADAPTIVE_THRESH_MEAN_C,
                                         cv.THRESH_BINARY, block_size, constant)

        self.set_info({'block size': block_size, 'constant': constant})
        return image_abw


'''Edge detection algoritm'''
class Canny(Method):
    def setup(self):
        self.title = 'Canny'
        self.settings = {'Low': 300,
                         'High': 600}

    def transform(self, image):
        low = self.settings['Low']
        high = self.settings['High']

        canny = cv.Canny(image, low, high)

        return canny


'''Apply Hough line transform after edge detection'''
class CannyAndHough(Method):
    def setup(self):
        self.title = 'Hough transform after edge detection'
        self.settings = {'Can. Low': 300,
                         'Can. High': 600,
```

```
                        'Hough thrs': 600}

    def transform(self, image):
        low = self.settings['Can. Low']
        high = self.settings['Can. High']
        h_thrs = self.settings['Hough thrs']

        canny = cv.Canny(image, low, high)
        lines = cv.HoughLines(canny, 1, np.pi/180, h_thrs)

        shape = image.shape
        diagonal = max(shape) * (2**0.5)
        mask = np.zeros(shape, np.uint8)
        n_lines = 0

        # Drawing found lines
        if type(lines) == np.ndarray:
            rho_s, theta_s = lines.transpose([1, 2, 0])[0]
            n_lines = len(rho_s)

            a = np.cos(theta_s)
            b = np.sin(theta_s)
            x0 = a * rho_s
            y0 = b * rho_s

            a1 = (diagonal)*(a)
            b1 = (diagonal)*(b)
            x1 = (x0 - b1).astype('int')
            y1 = (y0 + a1).astype('int')
            x2 = (x0 + b1).astype('int')
            y2 = (y0 - a1).astype('int')

            for i_x1, i_y1, i_x2, i_y2 in zip(x1, y1, x2, y2):
                cv.line(mask, (i_x1, i_y1), (i_x2, i_y2), 255, 1)

        self.set_info({'found lines': n_lines})
        return mask


'''Image smoothing using averaging'''
class Blur(Method):
    def setup(self):
        self.title = 'Blur'
        self.settings = {'Blur size': 12}

    def transform(self, image):
        kernel_side = self.settings['Blur size']
        kernel_size = (1 + 2*kernel_side,) * 2

        image_blurred = cv.blur(image, kernel_size)

        self.set_info({'size': kernel_size})
        return image_blurred


'''Image smoothing using bilateral filtering'''
```

```
class BilateralFilter(Method):
    def setup(self):
        self.title = 'Bilateral filter'
        self.settings = {'Size': 19,
                         'Sigma col': 200,
                         'Sigma spc': 200}

    def transform(self, image):
        size = 1 + self.settings['Size']
        sigma_col = self.settings['Sigma col']
        sigma_spc = self.settings['Sigma spc']

        image_blurred = cv.bilateralFilter(image, size, sigma_col, sigma_spc)

        self.set_info({'filter size': size})
        return image_blurred


'''Use binarization after blur'''
class BlurAndBinarization(Method):
    def setup(self):
        self.title = 'Blur and binarization'
        self.settings = {'Threshold': 255,
                         'Blur size': 12}

    def transform(self, image):
        threshold = self.settings['Threshold']
        kernel_side = self.settings['Blur size']
        kernel_size = (1 + 2*kernel_side,) * 2

        image_blurred = cv.blur(image, kernel_size)
        t, image_bw = cv.threshold(image_blurred, threshold, 255, cv.THRESH_BINARY)

        self.set_info({'blur size': kernel_size})
        return image_bw


'''Use binarization after blur, then fill closed contours'''
class BlurBinarizationFill(Method):
    def setup(self):
        self.title = 'Blur, binarization and contours filling'
        self.settings = {'Threshold': 255,
                         'Blur size': 12}

    def transform(self, image):
        threshold = self.settings['Threshold']
        kernel_side = self.settings['Blur size']
        kernel_size = (1 + 2*kernel_side,) * 2

        image_blurred = cv.blur(image, kernel_size)
        t, image_bw = cv.threshold(image_blurred, threshold, 255, cv.THRESH_BINARY)

        image_inv = 255 - image_bw
        contours, hierarchy = cv.findContours(image_inv, cv.RETR_TREE,
                                              cv.CHAIN_APPROX_SIMPLE)
```

```
        n_contours = 0
        if contours:
            for contour, c_info in zip(contours, hierarchy[0]):
                next_, prev, child, parent = c_info
                if parent != -1:    # is not external contour
                    cv.fillPoly(image_inv, [contour], 255)
                    n_contours += 1

        image_filled = 255 - image_inv

        self.set_info({'blur size': kernel_size,
                       'filled closed contours': n_contours})

        return image_filled


if __name__ == '__main__':
    model = cv.imread(r'images\conv_model.png', cv.IMREAD_GRAYSCALE)
    conv1 = cv.imread(r'images\conv1.png', cv.IMREAD_GRAYSCALE)
    conv2 = cv.imread(r'images\conv2.png', cv.IMREAD_GRAYSCALE)
    conv3 = cv.imread(r'images\conv3.png', cv.IMREAD_GRAYSCALE)
    conv4 = cv.imread(r'images\conv4.png', cv.IMREAD_GRAYSCALE)

    images = [model, conv1, conv3]

    hist_eq = HistEq()
    adapt_hist_eq = AdaptHistEq()
    binarization = Binarization()
    otsu = BinarizationOtsu()
    adapt_bin = AdaptiveBinarization()
    canny = Canny()
    hough_canny = CannyAndHough()
    blur = Blur()
    bilateral = BilateralFilter()
    blur_bw = BlurAndBinarization()
    blur_bw_fill = BlurBinarizationFill()

    methods = [hist_eq, adapt_hist_eq, binarization, adapt_bin, canny, hough_canny,
               blur, bilateral, blur_bw_fill]

    for method in methods:
        for image in images:
            method.run_experiment(image)
```