**Міністерство освіти і науки України**
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету )
Кафедра комп'ютерних наук
(повна назва кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів

Виконав: студент VI курсу, групи СНнм-61
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Сороківський О.В
(підпис) (прізвище та ініціали)

Керівник _____ Литвиненко Я.В
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Мацюк О.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Яцишин В.В
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
**Тернопільський національний технічний університет імені Івана Пулюя**

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«____» _Травень_ 2023 р.

# ЗАВДАННЯ
## НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Сороківському Олександру Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів

Керівник роботи Литвиненко Я.В. , доктор технічних наук, професор кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 22 » листопада 2022 року № 4/7-950

2. Термін подання студентом завершеної роботи .

3. Вихідні дані до роботи Наукові публікації про проектування систем для отримання статистичної інформації про гравців з записів відео матчів

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз систем формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів. 1.1 Загальна інформація про системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів. 1.2 Огляд конкурентних рішень. 1.3 Висновки до першого розділу. 2 Загальні підходи та основні методи досліджень. 2.1 Опис основних базових функцій. 2.2 Загальний огляд задач комп'ютерного бачення. 2.3 Існуючі дослідження та підходи знаходження об'єктів на зображенні. 2.4 Існуючі дослідження та підходи визначення позиції гравців на полі. 2.5 Існуючі дослідження та підходи. 2.6 Висновки до другого розділу. 3 Розробка системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів. 3.1 Основні вимоги до системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів. 3.2 Знаходження гравців та м'яча. 3.3 Визначення позиції гравців і м'яча на полі. 3.4 Ре-ідентифікація гравців. 3.5 Обрахунок статистичних даних. 3.6 Прикладний програмний інтерфейс (API). 3.7 Висновки до третього розділу. 4 Охорона праці та безпека в надзвичайних ситуаціях. 4.1 Вимоги електробезпеки у приміщеннях, де Встановлені персональні комп'ютери та відеодисплейні термінали. 4.2 Підвищення стійкості роботи підприємств приладобудівної галузі у воєнний час.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Охорона праці | Мацюк О.В., доцент | | |
| Безпека в надзвичайних ситуаціях | Клепчик В.М., ст. викладач | | |

7. Дата видачі завдання 14 листопада 2022 р.

КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів роботи | Термін виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Ознайомлення з завданням до кваліфікаційної роботи | 14.11.2022-15.11.2022 | *Виконано* |
| 2. | Підбір наукових джерел про системи формування | 16.11.2022-27.11.2022 | *Виконано* |
| | статистичної інформації на основу відео записів | | |
| | | | |
| 3. | Опрацювання наукових публікацій та збір даних по | 28.11.2022-25.12.2022 | *Виконано* |
| | темі роботи | | |
| 4. | Виконання дослідження згідно мети кваліфікаційної | 09.01.2023-12.03.2023 | *Виконано* |
| | роботи | | |
| 5. | Оформлення розділу «Аналіз систем формування | 13.03.2023-19.03.2023 | *Виконано* |
| | статистичної інформації про проведені спортивні | | |
| | Матчі на основі аналізу відео записів» | | |
| 6. | Оформлення розділу «Загальні підходи та основні | 20.03.2023-26.03.2023 | *Виконано* |
| | методи досліджень» | | |
| | | | |
| 7. | Оформлення розділу «Розробка системи формування | 27.03.2023-02.04.2023 | *Виконано* |
| | статистичної інформації про проведені спортивні | | |
| | матчі на основі аналізу відео записів» | | |
| 8. | Виконання завдання до підрозділу «Охорона праці» | 10.04.2023-16.04.2023 | *Виконано* |
| 9. | Виконання завдання до підрозділу «Безпека в | 17.04.2023-23.04.2023 | *Виконано* |
| | надзвичайних ситуаціях» | | |
| 10. | Оформлення кваліфікаційної роботи | 24.04.2023-30.04.2023 | *Виконано* |
| 11. | Нормоконтроль | 01.05.2023-07.05.2023 | *Виконано* |
| 12. | Перевірка на плагіат | 08.05.2023 | *Виконано* |
| 13. | Попередній захист кваліфікаційної роботи | 15.05.2023 | *Виконано* |
| 14. | Захист кваліфікаційної роботи | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Студент _____ Сороківський О.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

# АНОТАЦІЯ

Розробка системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів // Кваліфікаційна робота освітнього рівня «Магістр» // Сороківський Олександр Вікторович// Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНнм-61 // Тернопіль, 2023 // С. 78 , рис. – 25, табл. – 2, кресл. – 0, додат. – 2, бібліогр. – 50.

***Ключові слова:*** комп'ютерний зір, нейронні мережі. гомографія, знаходження об'єктів на зображенні, api, python, flask, футбол, персональна статистика про гравців

Кваліфікаційна робота присвячена розробці системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів. В першому розділі кваліфікаційної роботи проведено аналіз основних типових рішень, їх переваги та недоліки. В другому розділі кваліфікаційної роботи досліджено загальні підходи досліджень в галузі. В третьому розділі кваліфікаційної роботи проаналізовані існуючі підходи для вирішення задач знаходження об'єктів, визначення позиції їх на полі, трекінг об'єктів, ре-ідентифікація. Проведено розробку системи для отримання персональної статистики про гравців на основі відео записів.

Об'єкт дослідження: процеси опрацювання відео записів футбольних матчів.

Предмет дослідження: розробка автоматизованої системи для покращення процесу формування статистичних даних.

# ANNOTATION

Development of a System of Sports Matches Statistical Information Acquiring Based on Video Records Analysis// Qualification work of the educational level "Master" // Oleksandr Viktorovych Sorokivskшy // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, SNnm-61 group // Ternopil, 2023 // P.78, fig. – 25, tables – 2, chair. – 0, annexes – 2, references. – 50.

***Key words:*** computer vision, neural networks, homography, object detection on images, api, python, flask, football, personal player statistics.

The qualification work is dedicated to the development of a system for generating statistical information about conducted sports matches based on video analysis. The first chapter of the qualification work presents an analysis of systems, their main typical solutions, as well as their advantages and disadvantages. The second chapter of the qualification work explores general approaches to research in the field. The third chapter of the qualification work examines existing approaches to solving tasks such as object detection, determining their positions on the field, object tracking, and re-identification. The development of a system for obtaining personalized player statistics based on video recordings has been conducted as well.

The research object is the process of processing video recordings of football matches.

The research subject is the development of an automated system to improve the process of generating statistical data.

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ – програмне забезпечення.

CV (від англ. Computer Vision) – Комп’ютерне бачення.

AI (від англ. Artificial Intelligence) – Штучний інтелект. AI означає розвиток комп'ютерних систем, які можуть виконувати завдання, які зазвичай вимагають людського інтелекту, таких як розпізнавання мови, машинне навчання та прийняття рішень.

ML (від англ. Machine Learning) – Машинне навчання. ML – це галузь AI, яка вивчає алгоритми та моделі, щоб комп'ютерні системи могли навчатися та покращувати свою продуктивність на основі даних.

R-CNN (від англ. Region Convolutional Neural Network) – Регіон-конволюційна нейромережа. R-CNN – це архітектура нейромережі, яка використовується для об'єктного розпізнавання в зображеннях шляхом виділення та класифікації регіонів зображень.

YOLO (від англ. You Only Look Once) – Тільки один раз дивись. YOLO – це алгоритм об'єктного розпізнавання в реальному часі, який шукає об'єкти на зображенні шляхом одноразового прогляду всього зображення в цілому.

SSD (від англ. Single Shot MultiBox Detector) – Одноразовий багатошаровий детектор.

SVM (від англ. Support Vector Machine) – Метод опорних векторів. SVM – це алгоритм машинного навчання, який використовується для класифікації та регресії, шляхом розбиття даних за допомогою гіперплощини з максимальною відстанню до найближчих точок даних.

IOU (від англ. Intersection over Union) – Перетин на об'єднання. IOU – це метрика, що використовується для оцінки якості об'єктного розпізнавання або сегментації.

GAN (від англ. Generative Adversarial Network) – Генеративна змагальна мережа. GAN – це архітектура нейромережі, яка складається з двох основних компонентів: генератора і дискримінатора.

PCA (від англ. Principal Component Analysis) – Аналіз головних компонентів. PCA – це статистичний метод, що використовується для зменшення розмірності даних.

LDA (від англ. Linear Discriminant Analysis) – Лінійний дискримінантний аналіз. LDA – це метод зменшення розмірності даних та класифікації, який шукає нові ознаки, що максимально відрізняють класи одна від одної.

mAP (від англ. mean Average Precision) – середня середня точність. mAP – це метрика, що використовується для оцінки точності об'єктного розпізнавання в задачах комп'ютерного зору. Вона вимірює середнє значення точності по всіх класах об'єктів, де точність обчислюється як співвідношення правильно визначених об'єктів до всіх передбачених об'єктів.

# ЗМІСТ

# ВСТУП

**Актуальність теми.** Розробка системи обрахунку статистики з відеозаписів футбольних матчів є дуже актуальною темою, особливо в контексті сучасних технологій і популярності футболу в світі. За допомогою такої системи можна забезпечити точний і об'єктивний аналіз гри, виявляти сильні і слабкі сторони команд, а також допомагати тренерам в підготовці до наступних матчів. Окрім того, збір і аналіз статистичних даних з відеозаписів може бути корисним для організації змагань та підвищення їх рівня, підвищення інтересу глядачів до гри, а також розвитку футбольної індустрії в цілому. Також варто зазначити, що застосування технологій штучного інтелекту, які використовуються в розробці такої системи, може значно полегшити та пришвидшити процес аналізу відеозаписів, що дозволить збільшити ефективність роботи і знизити витрати часу і коштів.

**Мета і задачі дослідження.** Метою даної магістерської роботи є дослідження та розробка системи формування статистичних даних персональних показників гравців на основі опрацювання відео записів минулих футбольних матчів. Основними цілями були забезпечення ефективного та автоматизованого процесу отримання статистики про грацїв на основі відео записів матчів. Для досягнення цієї мети необхідно було виконати низку завдань, а саме:

1. Проаналізувати стан досліджень у сфері розробки автоматизованих систем формування статистики зі спортивних мачів базуючись на відео записах матчів.
2. Дослідити існуючі підходи до сучасної розробки автоматизованих систем формування статистики зі спортивних мачів базуючись на відео записах матчів.

3. Порівняти існуючі системи формування статистики зі спортивних мачів базуючись на відео записах матчів.

4. Дослідити які кроки мають бути виконані для успішного отримання статисти.

5. Розробити та натренувати моделі машинного навчання необхідні для отримання статистики.

6. Обґрунтувати вибір технологій, для виконання проектованої системи.

7. Розробити автоматизовану систему формування статистики зі спортивних мачів базуючись на відео записах матчів.

**Об’єкт дослідження** є процеси опрацювання відео записів футбольних матчів.

**Предмет дослідження** є розробка автоматизованої системи для покращення процесу формування статистичних даних.

**Наукова новизна одержаних результатів** полягає в тому, що розроблена автоматизована система формування статистичинх даних має великий потенціал застосування в індустрії аналізу футбольних матчів. Результати допоможуть розвитоку автоматизованих систем формування статистики про гравців на базі відео записів матчів.

**Практичне значення одержаних результатів.** Розроблено автоматизовану систему формування статистики з записів футбольних матчів, що оптимізує та вдосконалить наявні процеси у роботі тренера зменшить кількість помилок та пришвидшить розвиток команди. Результати можуть бути використані для подальших досліджень в області автоматизації формування статистики. Отримані результати також можуть бути використані для покращення роботи в інших сферах, пов'язаних з спортом.

**Апробація результатів магістерської роботи.** Основні результати проведених досліджень обговорювались на конференціях: X науково-технічна конференція «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 7-8 грудня 2022 року) і на XI Міжнародна науково-технічна конференція молодих учених та студентів «актуальні задачі сучасних технологій» Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль 7-8 грудня 2022 року).

**Публікації.** Основні результати кваліфікаційної роботи опубліковано у двох працях конференції (Див. додатки А).

**Структура й обсяг кваліфікаційної роботи.** Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку літератури з 50 найменувань та 2 додатків. Загальний обсяг кваліфікаційної роботи складає 78 сторінок, з них 73 сторінки основного тексту, який містить 25 рисунків та 2 таблиці.

# 1 АНАЛІЗ СИСТЕМ ФОРМУВАННЯ СТАТИСТИЧНОЇ ІНФОРМАЦІЇ ПРО ПРОВЕДЕНІ СПОРТИВНІ МАТЧІ НА ОСНОВІ АНАЛІЗУ ВІДЕО ЗАПИСІВ

## 1.1 Загальна інформація про системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записі

Статистична інформація про проведені спортивні матчі є важливим інструментом для аналізу та оцінки ефективності гравців, команд та тренерських стратегій. Зараз на основі аналізу відео записів створюються системи формування статистичної інформації про проведені спортивні матчі, які дозволяють отримувати більш точну та повну картину змагання.

Такі системи використовують технології комп'ютерного зору та машинного навчання для визначення різних параметрів гри, таких як позиція гравців, швидкість руху м'яча, кількість передач та інших дій. Ці дані аналізуються та використовуються для створення статистичних звітів, які можуть допомогти тренерам та гравцям в удосконаленні своїх стратегій та підвищенні результативності.

У цьому розділі ми розглянемо детальніше процес створення та використання систем формування статистичної інформації на основі аналізу відео записів спортивних матчів.

Командна гра у футболі настільки ж важлива наскільки і персональна кожного гравця. Статистичний аналіз матчу має враховувати не лише фінальний рахунок матчу, гольові удари і паси, але й різні характеристики кожного гравця, адже велика перемога складається з дрібних деталей.

Для того, щоб отримати інформацію про гравця і те, яка в нього була продуктивність під час гри потрібно, насамперед, обладнання. Зазвичай, на стадіоні встановлюється декілька камер, які стежать за всією грою. Далі, записи з камер потрапляють на комп’ютер експерта, який аналізує кожного

футболіста і його взаємодію з м’ячем в певний момент часу та записує ці дані. Після того вони аналізуються тренерами і вносяться корективи в гру.

Існують, також, системи які дозволяють робити частину цієї роботи автоматично. Ці системи використовують власні камери, які потребують попереднього налаштування спеціалістами.

### 1.2 Огляд конкурентних рішень

Spiideo – це шведська компанія, яка спеціалізується на розробці відеоаналітики для спорту. Вона пропонує інноваційні рішення для збору та аналізу відеоданих з метою підвищення ефективності тренувань та розвитку командного духу.

Однією з головних переваг Spiideo є їхній підхід до збору та аналізу відеоданих. Вони пропонують рішення, яке дозволяє збирати відеодані з кількох камер, що забезпечує більш повну картину того, що відбувається на полі/майданчику. Крім того, Spiideo має потужні інструменти для аналізу даних, які допомагають тренерам та спортсменам зрозуміти, як краще підготуватися до матчу або тренування.

Ще однією перевагою Spiideo є їхня здатність працювати з будь-яким видом спорту. Вони мають досвід у співпраці з футбольними, баскетбольними, хокейними, гандбольний та іншими командами, що дозволяє їм розуміти потреби різних видів спорту та надавати належну підтримку.

Крім того, Spiideo має вражаючий список клієнтів, який включає в себе професійні та коледжні команди з різних країн світу. Це свідчить про те, що їхні продукти і послуги вже успішно використовуються в спортивних галузях та здатні задовольнити потреби найвибагливіших клієнтів. Приклад послуг та програмного забезпечення компанії зображений на рисунку 1.1.


Рисунок 1.1 – Приклад послуг та програмного забезпечення

Основні переваги Spiideo полягають у його високій якості відео та можливості безпосереднього доступу до відеоданих у режимі реального часу. Це робить Spiideo ідеальним рішенням для використання в тренуваннях та змаганнях, де потрібна точна інформація про гру, яка може бути використана для підготовки та аналізу команди.

Окрім цього, Spiideo має досить простий та зручний інтерфейс, що дозволяє легко використовувати програму. Крім того, Spiideo може бути інтегровано з іншими програмами, такими як системи відео статистики та аналізу даних.

Проте, є кілька недоліків у Spiideo. По-перше, вартість програми може бути досить високою для деяких клубів та тренерів. По-друге, Spiideo має обмежені можливості збору та аналізу даних порівняно з іншими програмами, такими як Hudl або Sportscode. Крім того, Spiideo може бути менш ефективним

у віддалених регіонах зі слабкою Інтернет-зв'язком або обмеженими можливостями мережі.

Крім того, Spiideo має обмежені можливості щодо використання в спорті з незвичайними формами та правилами, такі як волейбол, бейсбол або гандбол. У таких випадках можуть знадобитись додаткові налаштування та адаптація програми.

Отже, хоча Spiideo має свої переваги та недоліки, він все ще є досить потужним та ефективним інструментом для відеоаналізу в спорті, який може допомогти тренерам та командам підготуватися до змагань та підвищити свій рівень гри.

Ще одним кокурентом є Hudl. Hudl – це американська компанія, яка була заснована у 2006 році та спеціалізується на розробці програмного забезпечення для відеоаналізу спортивних змагань та тренувань. Компанія пропонує рішення для команд та тренерів з різних спортивних дисциплін, таких як футбол, баскетбол, волейбол, американський футбол, гандбол та інші. Приклад інтерфейсу програми Hudl зображений на рисунку 1.2.


Рисунок 1.2 – Приклад інтерфейсу Hudl

Track160 – це програмне забезпечення яке дозволяє аналізувати та отримувати статистику про гру використовуючи лише одну камеру з широким кутом обзору. Це програмне забезпечення сертифіковане FIFA та містить в собі безліч корисних функцій таких як:

- Аналіз продуктивності. Цей функціонал дозволяє детально проаналізувати виступи команди, виділити області вдосконалення, отримати дані як і про окремих гравців так і про цілу команду.
- Аналіз відео. Ця функція дає можливість переглянути та оцінити ключові моменти матчу через колекцію автоматично створення відеокліпів, які охоплюють кожну подію. Також, можна створити спеціальний плейлист високої якості з кліпів ключових передач, голів, штрафних ударів або небезпечних передач, щоб в подальшому поділитись ними.
- Наука про спорт. Дозволяє досліджувати детальні звіти про фізичні показники гравців та переглядати їхні дані про фітнес та тактику протягом матчу. Також, за допомогою цієї функції можна аналізувати дані, щоб переглянути фізичні показники, такі як спринти, пройдена відстань та прискорення.

Приклад інтерфейсу та можливостей програми зображено на рисунку 1.2.

Рисунок 1.2 – Приклад інтерфейсу та статистики

Nacsport – це іспанська компанія, яка спеціалізується на розробці програмного забезпечення для відеоаналізу в спорті. Вони надають інструменти для збору, аналізу та візуалізації відеоданих, що дозволяє тренерам та аналітикам отримати розширений аналіз гри та статистику гравців та команд. Приклад інтерфейсу програми зображено на рисунку 1.3.


Рисунок 1.3 – Приклад інтерфейсу програми Nacsport

Основна мета Nacsport – це допомогти тренерам та аналітикам підвищити ефективність тренувань та змагань, використовуючи відео аналіз. Пакет Nacsport включає в себе ряд інструментів для збору та обробки відеоданих, таких як можливість записувати відео та аудіо, ручне або автоматичне відстеження дій гравців, різноманітні інструменти для побудови діаграм, статистичних таблиць, графіків та інших візуалізацій даних.

Одним з основних переваг Nacsport є простота використання та налаштування. Користувачі можуть налаштувати програму під свої потреби та додати свої власні параметри для збору та аналізу даних. Крім того, Nac Sport працює з більшістю відеоформатів та може інтегруватись з іншими програмами для подальшого аналізу даних.

Наголос в Nacsport зроблений на взаємодію з клієнтами та наданням розумних рішень для їхніх потреб. Компанія надає підтримку клієнтам, які

можуть звертатися до них з запитами щодо роботи програмного забезпечення та можливостями його використання для своїх потреб.

Ця система володіє своїми перевагами та недоліками.

Переваги Nacsport:

- Великий вибір функцій: Nacsport має широкий набір функцій, які допомагають тренерам та гравцям розуміти, як краще грати на полі, зокрема функцію відтворення, можливість позначення на відео, створення візуалізацій гри тощо.
- Легка інтеграція з іншими системами: Nacsport може інтегруватися з іншими системами, що дозволяє користувачам легко обмінюватися даними та використовувати їх для подальшого аналізу.
- Підтримка різних видів спорту: Nacsport може використовуватися для аналізу різних видів спорту, включаючи футбол, баскетбол, хокей, регбі та інші.
- Надійна та безпечна система: Nacsport забезпечує безпеку та захист даних користувачів, зокрема за допомогою шифрування та автоматичного резервного копіювання.

Недоліки Nacsport:

- Висока вартість: Nacsport може бути досить дорогим, що може бути проблемою для невеликих спортивних клубів або команд з обмеженим бюджетом.
- Вимоглива система: Nacsport має деякі вимоги до обладнання та ПЗ, що може бути проблемою для користувачів з менш потужними комп'ютерами.
- Висока крутизна навчання: Nacsport має складну інтерфейс та велику кількість функцій, що може потребувати часу на навчання та освоєння програми.

Dartfish – це компанія, що спеціалізується на розробці програмного забезпечення для відеоаналізу в спорті та фітнесі. Основний продукт компанії – це Dartfish 360, який дозволяє збирати, аналізувати та візуалізувати відеодані з тренувань та змагань для подальшого використання в процесі підготовки спортсменів. Приклад інтерфейсу програми зображено на рисунку 1.4.


Рисунок 1.4 – Приклад інтерфейсу програми Dartfish

Dartfish 360 містить у собі різні інструменти, які дозволяють користувачам збирати відеодані, використовуючи різні джерела, такі як камери відеоспостереження, смартфони, планшети, та інші. Крім того, програмне забезпечення має функції автоматичного визначення дій гравців, що дозволяє швидко та легко створювати відеоаналіз гри та працювати з даними.

Dartfish 360 містить у собі більше 40 інструментів для аналізу даних, включаючи можливість створювати діаграми, графіки, таблиці та інші

візуалізації даних. Крім того, програмне забезпечення має можливість додавати коментарі та маркувати відеодані, що дозволяє тренерам та аналітикам створювати детальний аналіз гри та підготовки спортсменів.

Однією з ключових особливостей Dartfish є його можливості для обробки великої кількості даних. Програмне забезпечення має можливість обробляти відео у реальному часі та швидко створювати візуалізації даних. Крім того, Dartfish 360 може інтегруватись з іншими програмами та системами, що дозволяє легко обмінюватись даними та розширювати функціональність програм.

Ця система також містить певні переваги та недоліки

Переваги Dartfish:

- Великий вибір функцій: Dartfish має великий набір функцій, які допомагають тренерам та гравцям аналізувати та покращувати їх гру, зокрема, функцію відтворення, можливість позначення на відео, створення візуалізацій гри тощо.
- Легка інтеграція з іншими системами: Dartfish може інтегруватися з іншими системами, що дозволяє користувачам легко обмінюватися даними та використовувати їх для подальшого аналізу.
- Підтримка різних видів спорту: Dartfish може використовуватися для аналізу різних видів спорту, включаючи футбол, баскетбол, хокей, регбі та інші.
- Простота використання: Dartfish має інтуїтивно зрозумілий інтерфейс та просту навігацію, що дозволяє користувачам легко використовувати програму.
- Надійна та безпечна система: Dartfish забезпечує безпеку та захист даних користувачів, зокрема за допомогою шифрування та автоматичного резервного копіювання.

Недоліки Dartfish:

- Висока вартість: Dartfish може бути досить дорогим, що може бути проблемою для невеликих спортивних клубів або команд з обмеженим бюджетом.
- Вимоглива система: Dartfish має вимоги до обладнання та ПЗ, що може бути проблемою для користувачів з менш потужними комп'ютерами.

Ще одним з конкурентів є програмне забезпечення Track160. Це програмне забезпечення дозволяє отримувати статистику про гравців, вимірювати їх характеристики для загальну силу гри. На рисунку 1.5 зображено приклад інтерфейсу програми.


Рисунок 1.4 – Приклад інтерфейсу програми Track160

Дане програмне забезпечення не працює з користувацькими камерами, а потребує встановлення додаткових трьох камер, що є не дешево.

Переваги цього програмного забезпечення:

- Великий вибір статистичних фукнцій.
- Інтерактивна карта поля гри.

Недоліки Track160:

- Ціна.
- Потреба у встановлення власних камер.

Порівняльна таблиця конкурентних рішень наведена нижче.

Таблиця 1.1 – Порівняння існуючих рішень

| Назва | Функція мануального лейбування відео і вираховування статистики | Знаходження гравців на відео | Трекінг гравців на відео | Автоматичний обрахунок статистичних даних | Потреба у встановлені камер компанії |
|---|---|---|---|---|---|
| Spiideo | Так | Так | Так | Ні | Так |
| Hudl | Так | Ні | Ні | Ні | Так |
| Nacsport | Так | Ні | Ні | Ні | Так |
| Dartfish | Так | Ні | Ні | Ні | Так |
| Track160 | Так | Так | Так | Так | Так |

З порівняльної таблиці стає зрозуміло, що є потреба в програмному забезпеченні, яке може аналізувати відео з власних камер, якщо якості відео достатньо.

### 1.3 Висновки до першого розділу

Spiideo – програма, яка надає можливість аналізувати відеоматеріали в режимі реального часу та зберігати їх у високій якості. Програма має можливість підключення до багатьох камер та автоматичного розпізнавання подій на полі. Однак, вона може бути вимоглива до ресурсів комп'ютера та не має такого рівня кастомізації, який можуть надати інші програми.

Hudl – програма, яка дозволяє тренерам та командам зберігати та аналізувати відеоматеріали з різних аспектів гри, таких як паси, удари, виконання тактичних комбінацій та інші. Програма має високий рівень кастомізації та можливостей для колективної роботи. Однак, вона може бути досить дорогим варіантом та вимагати додаткових ресурсів для її роботи.

Track160 – програма з високоточним трекінгом гравців та м'яча, яка надає велику кількість статистичних даних, що допомагає тренерам та командам розуміти, як краще готуватись до матчів та як ефективніше використовувати свої ресурси. Однак, ця програма може бути досить складною в освоєнні та вимагати певної кількості додаткового обладнання.

Nacsport – програма з простим інтерфейсом, яка надає користувачам можливість обробляти велику кількість відеоматеріалу та дозволяє легко відслідковувати гравців і події на полі.

Dartfish – програма, яка надає можливість створювати відеоаналізи з різних кутів та додавати до них різні ефекти та анімації. Вона також дозволяє спільну роботу над відео з іншими користувачами. Однак, програма може бути досить складною в освоєнні та вимагати додаткових ресурсів для роботи.

Усі вище перелічені програми мають як і свої переваги та недоліки. Найбільшим недоліком є те, що вони потребують великих затрат для того повного функціонування. В цій роботі пропонується вирішення задачі автоматизованого отримування статистики з матчу яке дозволяє інтегрувати його у будь-який інтерфейс з подальшим використанням.

# 2 ЗАГАЛЬНІ ПІДХОДИ ТА ОСНОВНІ МЕТОДИ ДОСЛІДЖЕНЬ

## 2.1 Опис основних базових функцій

Для отримання статистики про гру та гравців програмне забезпечення має містити наступні функції:

1. Знаходження гравців на картинці. Без інформації про гравців статистику визначити неможливо. Програма має точно вказувати координати кожного з гравців, а також їхні межі.

2. Знаходження м’яча на картинці. Деякі статистичні метрики потребують визначення положення м’яча на полі.

3. Слідкування за кожним з гравців. Якщо використовувати визначення гравців на картинці без попередньої інформації, то на кожному новому кадрі будуть нові гравців і статистику в такому випадку заміряти буде неможливо. Тому, важко не тільки виявляти гравців на картинці, а й слідкувати за тим куди воно переміщуються і маркувати тих самих гравців тим самим маркуванням.

4. Визначення кутів повороту поля відносно камери. Не маючи реальних координатів гравців їх визначення та слідкування не мають ніякого сенсу, адже статистична інформація, яка використовує фізичні показники не несе ніякого змісту.

З описаного вище стає ясно, що задача отримання статистики з відеозаписів матчів не є легкою та потребує врахування багатьох факторів. Більше деталей та основні підходити кожної функції будуть описані в наступних розділах.

## 2.2 Загальний огляд задач комп’ютерного бачення

До недавнього часу комп'ютерне бачення мало мало спільного з нашою повсякденністю. Хоча ідея розробки цієї технології з'явилася ще у 1950-х роках, для більшості людей це було щось з науково-фантастичних фільмів. Але ситуація змінилася. Сьогодні алгоритми комп'ютерного бачення є невід'ємною частиною багатьох сучасних програмних рішень, а рівень їх прийняття постійно зростає.

За поточної динаміки, глобальний ринок комп'ютерного бачення прогнозується досягти $21,17 мільярда до 2028 року, з річним темпом зростання в 6,9% з 2021 року.

Комп'ютерне бачення (CV) – це підгалузь штучного інтелекту (AI), що дозволяє комп'ютерним системам аналізувати та інтерпретувати візуальну інформацію. Сучасні CV системи можуть використовувати дані безпосередньо з камер та термальних сенсорів або обробляти готові набори даних. Ідея полягає в тому, що ми хочемо, щоб машини ідентифікували об'єкти реального світу та приймали рішення на основі візуальних даних швидко – найкраще в режимі реального часу.

Але як працює комп'ютерне бачення? Спочатку ви повинні забезпечити алгоритм CV візуальним посиланням, і ви повинні зробити це у числах, оскільки комп'ютери не добре вміють творчо мислити. Щоб система на основі CV могла ідентифікувати зображення миші потрібно зробити наступні кроки:

1. Знайти зображення миші;
2. Позначити його як “миша” і закодувати його піксель за пікселем використовуючи інформацію про кольори.
3. Повторити це з кількома тисячами зображень.

Система CV аналізує дані та розпізнає патерни. Система застосовує визначення до нового зображення, перевіряючи кадри на наявність патернів та виявляючи мишей в режимі реального часу.

Це, звичайно, спрощений сценарій. Сьогоднішня технологія комп'ютерного бачення дійшла далеко, і вже не потребує ручної роботи. AI використовує машинне навчання (ML) та глибоке.

Класифікація зображень є важливою темою у галузі комп'ютерного бачення. Це процес, який дозволяє комп'ютеру ідентифікувати та класифікувати об'єкти на зображеннях на основі їх характерних ознак. Для досягнення цієї мети, комп'ютерні системи використовуються для навчання інтелектуальних алгоритмів, які навчаються розпізнавати певні зразки на зображеннях.

У процесі класифікації зображень, зображення розбивають на окремі пікселі та аналізують їх характеристики, такі як колір, яскравість, форма та розташування. Після цього система порівнює ці характеристики з тими, які вона навчилася розпізнавати, і визначає, який об'єкт зображений на зображенні.

Класифікація зображень має різноманітні застосування у різних галузях, таких як медицина, автомобільна промисловість, безпека, моніторинг довкілля та інше. Наприклад, системи класифікації зображень можуть використовуватися для виявлення раку на рентгенівських знімках, розпізнавання транспортних засобів на вулицях та використовуватися в системах безпеки для розпізнавання обличчя та ідентифікації злочинців.

Застосування класифікації зображень вимагає значної обробки даних та використання потужних алгоритмів машинного навчання. Але, завдяки розвитку технологій і алгоритмів, класифікація зображень стає все більш точною та широко використовується в різних галузях технологій.

Для поточної задачі лише класифікації картинок не достатньо, тому для пошуку гравців на зображенні використовується більш інноваційна методика – знаходження об'єктів на зображенні з використанням комп'ютерного

бачення. Знаходження об'єктів на зображенні є однією з основних задач комп'ютерного бачення. Ця задача полягає в тому, щоб знайти та локалізувати об'єкти на зображеннях, що важливо для багатьох різних застосувань, включаючи медичну діагностику, автопілоти в автомобілях, відеоспостереження та багато інших.

Для знаходження об'єктів на зображенні застосовуються різні техніки, такі як методи, засновані на виокремленні ознак, або методи, засновані на навчанні з вчителем та без вчителя. Один з найпоширеніших методів полягає в застосуванні глибоких нейронних мереж, які здатні вивчати різноманітні ознаки зображення та виконувати задачу знаходження об'єктів на зображенні.

Інші методи, такі як детектори об'єктів, використовуються для знаходження об'єктів на реальному часі в відеопотоках або в режимі реального часу. Для досягнення високої точності та ефективності використовуються різні техніки, такі як мережі Region-based Convolutional Neural Network (R-CNN), Fast R-CNN та Faster R-CNN та інші.

У загальному, знаходження об'єктів на зображенні є важливою задачею комп'ютерного бачення, яка має широкі застосування та використовує різні техніки для досягнення високої точності та ефективності.

Для знаходження гравців на зображенні потрібно використовувати нейронні мережі, адже вони є найпотужнішим інструментом для цієї задачі на поточний час. Одним із найпопулярніших рішень є конвулюційні нейронні мережі.

Конволюційні нейронні мережі (Convolutional Neural Networks, CNNs) є типом нейронних мереж, який широко використовується у багатьох завданнях машинного навчання, зокрема в обробці зображень та розпізнаванні образів.

Основна ідея застосування конволюційних нейронних мереж полягає в тому, щоб використовувати спеціалізовані шари, які розуміють певні

характеристики зображення, такі як контури, форми та текстури. Ці шари зазвичай складаються з фільтрів, які здійснюють операцію згортки над вхідним зображенням, що дозволяє виділяти важливі ознаки зображення.

У конволюційних нейронних мережах також використовуються інші типи шарів, такі як підвибіркові (pooling) шари, які зменшують розмір вхідного зображення, та повнозв'язні (fully connected) шари, які здійснюють класифікацію зображення на основі отриманих ознак.

Конволюційні нейронні мережі зазвичай навчаються методом зворотного поширення помилки (backpropagation), де ваги шарів оновлюються на основі різниці між передбаченим результатом і фактичним результатом.

## 2.3 Існуючі дослідження та підходи знаходження об’єктів на зображенні

Виявлення об'єктів за допомогою глибокого навчання є найрозумнішим підходом до проблеми. Цей алгоритм можна широко розділити на дві категорії – метод з одним етапом та метод з двома етапами. Метод з двома етапами включає алгоритми, такі як R-CNN, R_FCN та FPN-FRCN, а метод з одним етапом включає алгоритми YOLO, SSD та RetinaNet. Далі будуть розглянуті ключові терміни, які корисні для розуміння цих двох алгоритмів – R-CNN та YOLO. Чому саме ці два? R-CNN дав прорив у галузі виявлення об'єктів, а YOLO є одним з широко використовуваних алгоритмів виявлення об'єктів та є сучасністю.

### 2.3.1 Огляд архітектури нейронної мережі R-CNN

R-CNN (регіони засновані на CNN). Як і саме ім'я підказує, ці алгоритми працюють з регіонами зображення, а не з усім зображенням. Крім того, вони не працюють з великою кількістю регіонів, а працюють тільки з кількома

прямокутниками (боксами) на зображенні. Коли ми надаємо вхідне зображення до R-CNN, він виділяє обмежувальні прямокутники для регіонів, які "ймовірно" є об'єктами.

Обмежувальні прямокутники: Обмежувальний прямокутник – це уявний прямокутник, який використовується для обмеження об'єкта інтересу на зображенні або відео. Обмежувальний прямокутник виступає як точка посилання для розпізнавання об'єктів та використовується для ідентифікації місця та розміру об'єкта в межах зображення. Обмежувальний прямокутник визначається координатами центра прямокутника (x, y) відносно обмежувального прямокутника, а також його шириною та висотою (w, h). Ці координати та розміри надають рамку, яка обмежує об'єкт інтересу. Приклад роботи такої CNN зображено на рисунку 2.1.

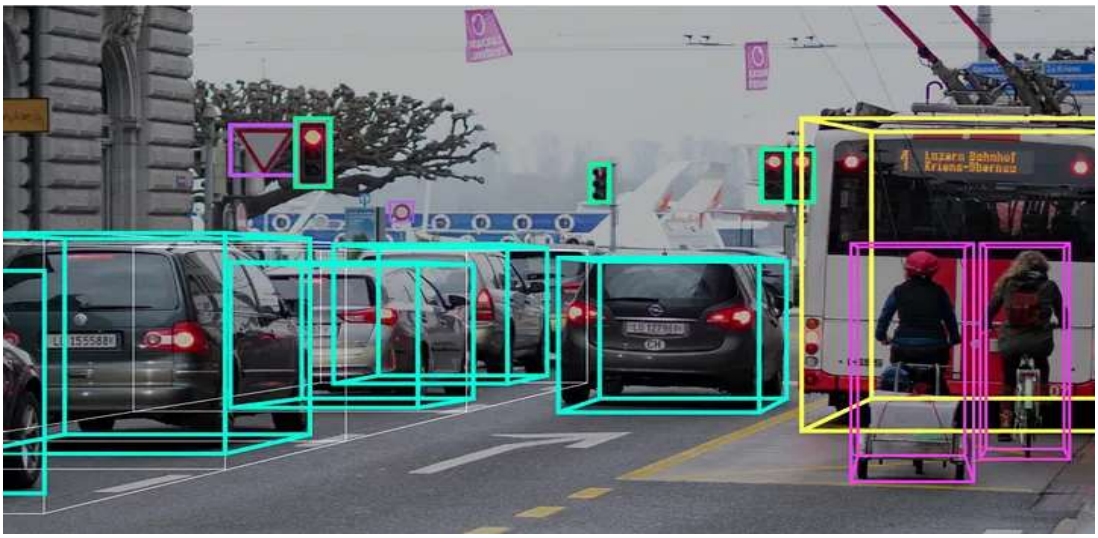

Рисунок 2.1 – Приклад роботи R-CNN

Ці прямокутники створюються на основі алгоритму вибіркового пошуку. Алгоритм вибіркового пошуку намагається розділити зображення на групи, поєднуючи схожі області, такі як кольори / текстури / різноманітні

масштаби і т.д. Він визначає ці патерни та пропонує ці області як "цікаві" межі обмежування. На рисунку 2.2 зображено приклад роботи алгоритму вибіркового пошуку – ліва вхідне зображення, а справа те, як його обробляє алгоритм.

Рисунок 2.2 – Приклад обробки зображення за допомогою алгоритму вибіркового пошуку

Тепер кожен з цих прямокутників передається до CNN для вилучення ознак. Після отримання ознак, вони передаються до SVM для класифікації. CNN видає фіксований вектор ознак для кожної пропозиції регіону, який використовується як вхід до набору лінійних SVM (машина опорних векторів), що класифікують об'єкт в межах регіону. SVM навчені відрізняти пропозиції об'єктів від необ'єктових пропозицій. Далі застосовується лінійна регресія для генерації більш вузьких межових коробок. Нижче наведено зображення, що допоможе нам зрозуміти більше. На рисунку 2.3 зображено приклад архітектури моделі.

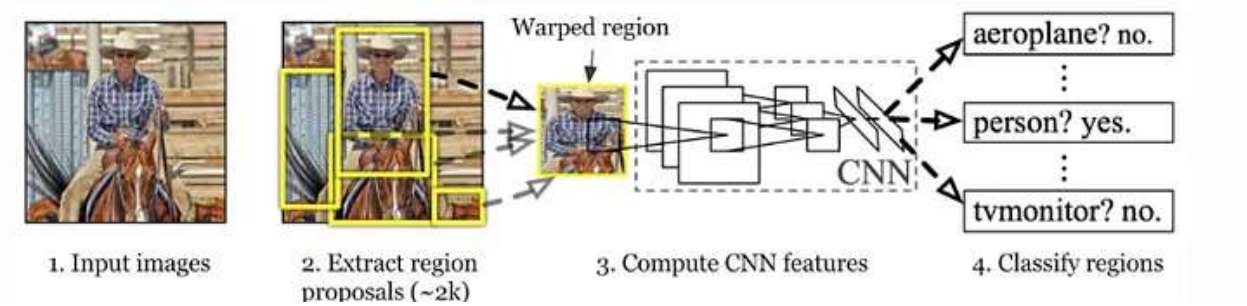


Рисунок 2.3 – Приклад архітектури готової моделі

Щоб підсумувати R-CNN, можна сказати, що він працює у двох етапах – визначення областей та класифікація об'єктів. Етап визначення областей визначає тисячі потенційних областей за допомогою алгоритму вибіркового пошуку. Етап класифікації об'єктів класифікує кожну пропозицію об'єкту та локалізує його у межах області. Для навчання R-CNN потрібно окремо навчити три моделі – CNN (модель вилучення ознак), SVM (модель класифікації об'єктів) та модель лінійної регресії для точнішого обмеження прямокутника-рамки.

R-CNN представляє нову хвилю в області виявлення об'єктів, але має повільний час роботи та високу обчислювальну вартість. Це призвело до розвитку Fast R-CNN та Faster R-CNN.

Детально ці методи на будуть розглянуті, але нижче наведено декілька ключових відмінностей:

- Region proposal: R-CNN використовує алгоритм вибіркового пошуку (selective search) для створення областей пропозицій регіонів, тоді як Fast R-CNN і Faster R-CNN використовують мережу областей пропозицій (RPN) для генерації областей пропозицій регіонів. RPN в Faster R-CNN є більш ефективним, ніж алгоритм вибіркового пошуку, що використовується в R-CNN.

- Виявлення об'єктів: У R-CNN кожна область пропозиції регіону передається через окрему мережу згортки для класифікації та локалізації об'єкта в межах регіону. Fast R-CNN і Faster R-CNN спільно використовують функції мережі згортки для всіх областей пропозицій регіонів, що робить їх більш обчислювально ефективними.
- Тренування: У R-CNN мережу згортки та машини опорних векторів (SVM) навчають окремо, що може призвести до неповноцінної продуктивності. Fast R-CNN і Faster R-CNN використовують спільний підхід до тренування, що дозволяє навчати разом мережу згортки та мережу виявлення об'єктів, що призводить до кращої продуктивності.
- Вихід: У R-CNN і Fast R-CNN вихідним є набір обмежуючих прямокутників та класових міток для виявлених об'єктів на вхідному зображенні. Faster R-CNN покращує це, прямо передбачаючи обмежуючі прямокутники та класові мітки, використовуючи RPN.

### 2.3.2 Огляд архітектури нейронної мережі YOLO

Як було згадано раніше, YOLO широко використовується завдяки своїй великій швидкості та точності – він надзвичайно швидкий та може обробляти 45 кадрів за секунду. Вищезгадані двоетапні методи розв'язують проблему виявлення об'єктів як класифікаційну задачу, а YOLO вирішує її як задачу регресії.

YOLO застосовує одну нейромережу до всього зображення. Спочатку воно розділяє вхідне зображення на сітку розміром S x S. Кожна отримана з сітки комірка відповідає за виявлення об'єктів у межах своєї території. На рисунку 2.3 зображено приклад сітки розміром 7x7.

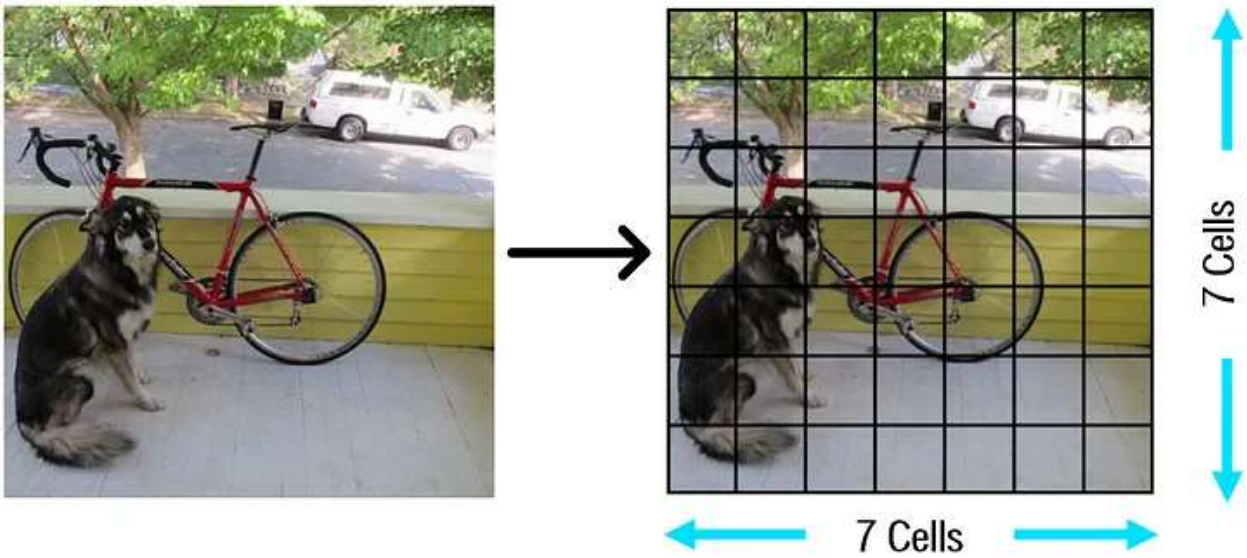


Рисунок 2.3 – Приклад сітки розміром 7x7

Якщо центр об'єкту, який ми отримали з навчальних даних, потрапляє в комірку сітки, то ця комірка сітки відповідальна за виявлення цього об'єкту. Кожна комірка сітки передбачає B прямокутників та оцінку достовірності для цих прямокутників. Тут B є попередньо заданою кількістю прямокутників на кожну комірку сітки. Наприклад, якщо B встановлено на 2, кожна комірка сітки передбачає 2 прямокутника.

Оцінка достовірності – оцінка, яка представляє ймовірність того, що межа рамки містить об'єкт. Оцінка достовірності обчислюється за допомогою функції логістичної регресії на основі перетину та об'єднання (IoU) між передбаченою межею рамки та правдивою межею рамки. Достовірність = ймовірність Pr (Object) * IoU. Якщо немає передбаченого об'єкту, то оцінка достовірності повинна дорівнювати 0.

Перетин над об'єднанням – IOU розраховується шляхом ділення площі перетину (частини, яка перекривається між двома обмежуючими рамками – зразком і передбачуваною рамкою) на площу їх об'єднання (загальну площу, яку охоплюють обидві обмежувальні рамки). IOU коливається від 0 до 1, де 1

вказує на те, що дві обмежувальні рамки досконало перекриваються, а 0 означає відсутність перекриття. На рисунку 2.4 зображено візуальний приклад обчислення метрики.

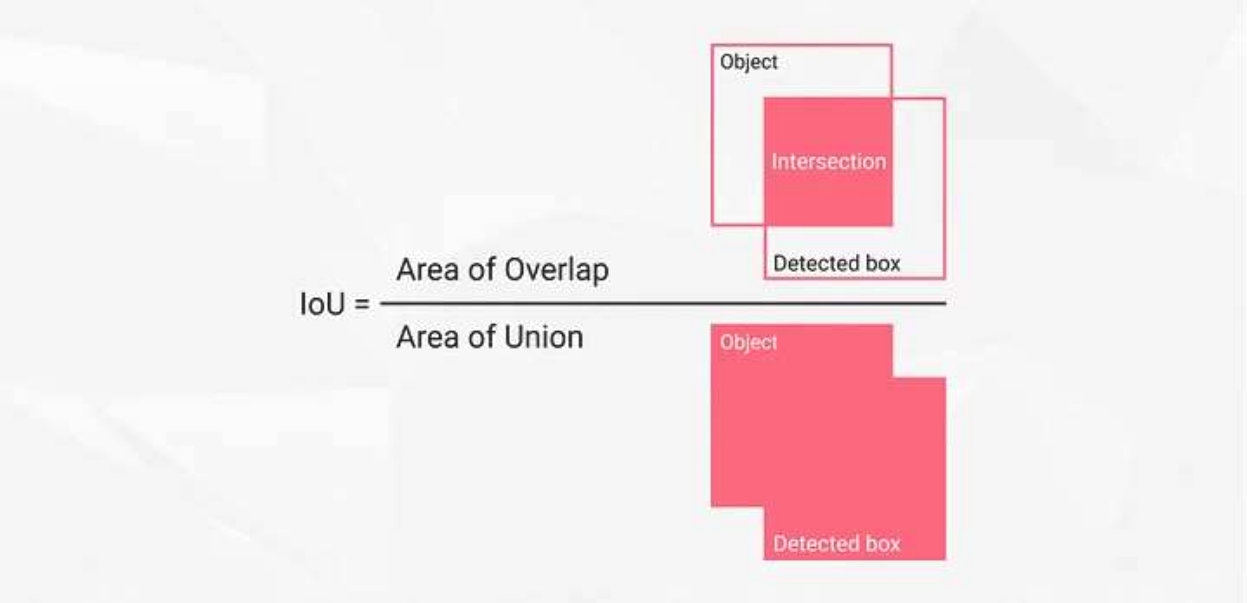


Рисунок 2.4 – Візуальний приклад обрахунку IOU

Кожна комірка сітки також передбачає C умовних ймовірностей класу, Pr(Classi |Object). Вона передбачає лише один набір ймовірностей класу на кожну комірку сітки, незалежно від кількості прямокутників B.

Тепер, коли я чітко зазначив, що одна сітка може передбачити тільки один клас, що робити, якщо в одній сітці є декілька центрів декількох об'єктів? Щоб вирішити цю проблему, ми можемо використовувати метод збільшення кількості сіток. Це створить клітинки меншого розміру, тому ми матимемо велику кількість клітинок. Але що, якщо два різних об'єкти мають одну і ту ж точку центру? Для цього була розроблена концепція якорних коробок (Anchor Boxes).

Якірні рамки – це наперед визначені прямокутники з різними розмірами та співвідношеннями сторін, щоб генерувати пропозиції об'єктів.

На рисунку 2.5 показано, що можна створити дві якірні рамки для фігури, показаної як Анкерна рамка 1 та 2. Не обов'язково мати тільки 2 якірні рамки, їх може бути будь-яка кількість, залежно від типу датасету.

Кожна сіткова комірка тепер має дві якірні рамки, де кожна якірна рамка діє як контейнер. Це означає, що тепер кожна сіткова комірка може передбачати до 2 об'єктів. Але чому вибрати дві якірні рамки з двома різними формами? Ідея полягає в тому, щоб під час прийняття рішення про те, який об'єкт поміщається в яку якірну рамку, враховувати їх форми, відзначаючи, наскільки схожа форма рамки на форму рамки-анкер. Для вищезазначеного прикладу людина буде пов'язана з високою якірною рамкою, оскільки їхня форма більш схожа.

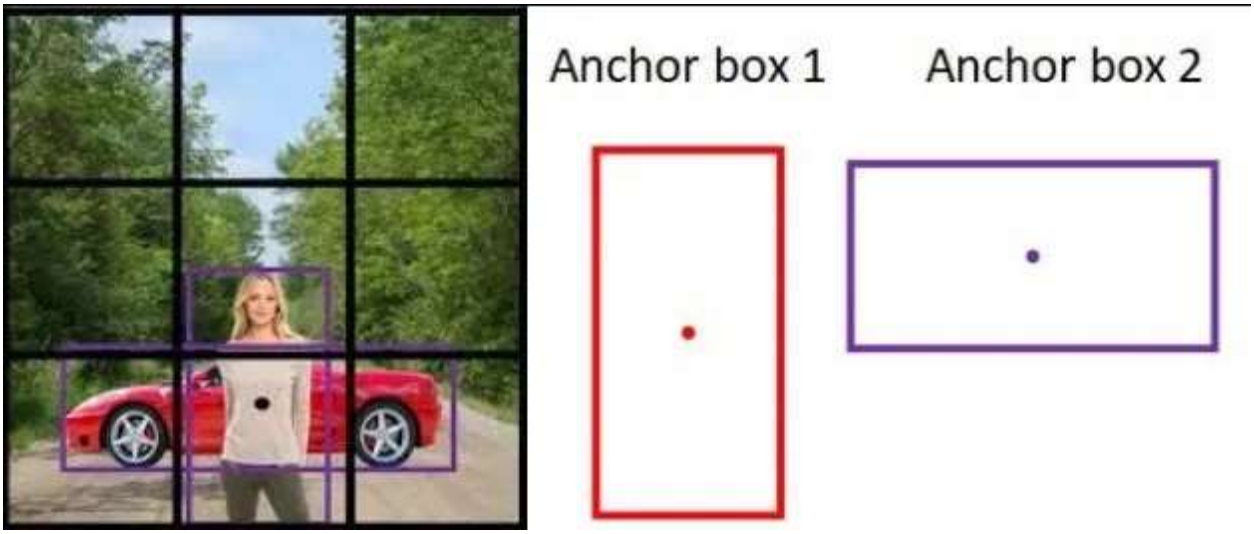


Рисунок 2.5 – Приклад якірних рамок

Щоб уникнути повторного виявлення одного і того ж об’єкту, YOLO використовує техніку після обробки, що називається non-max suppression.

Вона використовується для вилучення дублюючихся рамок-обмежень та вибору найбільш релевантних.

Non-max suppression працює шляхом порівняння оцінок достовірності запропонованих обмежуючих рамок та видалення тих, які перекриваються значно з обмежувальною рамкою з більш високою оцінкою. Кроки для цього можуть бути описані наступним чином:

1. Сортування обмежувальних рамок за їхніми оцінками достовірності.

2. Вибір обмежувальної рамки з найвищою оцінкою та зберігання її як виявлення.

3. Виділення всіх обмежувальних рамок, які мають значний перекриття з вибраною обмежувальною рамкою (це робиться шляхом обчислення IOU). Кількість перекриття зазвичай вимірюється передбачуваним пороговим значенням.

Отже, алгоритм YOLO є швидким порівняно з іншими алгоритмами виявлення об'єктів, оскільки обробляє все зображення за один прохід через мережу, та точним, оскільки враховує контекст зображення та просторові відносини між об'єктами.

## 2.4 Існуючі дослідження та підходи визначення позиції гравців на полі

Задача визначення позиції гравців на полі є однією з найважчих в цій роботі, адже з інформації є лише кадри з відео. Вирішення цієї задачі неможливе без врахування додаткових змінних і є декілька основних шляхів вирішення.

### 2.4.1 Визначення матриції гомографії ручним методом

Ручне налаштування з однією або кількома камерами. Для того, щоб визначити позицію гравців на полі потрібно знати межі поля, відстань від основи камери до поля, відстань від верхівки камери до поля та матрицю гомографії.

Матриця гомографії – це матриця перетворення, яка використовується для зображення тривимірних об'єктів в двовимірному просторі або для виконання інших подібних завдань у графічному процесингу. Гомографія – це математична концепція, яка описує відношення між двома проективними просторами, що можуть бути реалізовані як зображення тривимірних об'єктів на плоскість.

Матриця гомографії зазвичай визначається за допомогою калібрування камери та підгонки до точок, які відображають зображення на площині. Знаючи матрицю гомографії, можна виконувати операції з перетворенням зображень, такі як збільшення, зменшення, поворот і трансляція.

У більш загальному сенсі, матриця гомографії використовується для опису перетворень між просторами будь-якої кількості вимірів, і не обмежується лише областю графічного процесингу.

Тому, в цьому підході налаштування матриці відбувається вручну. Спочатку виставляється одна широкоформатна або декілька камер в сталу позицію, вибираються ключові точки з 3д простору і 2д простору та знаходиться матриця гомографії.

У випадку з однією камерою потрібно її виставляти рівно, адже похибка в пару міліметрів може стати похибкою в пару метрів, через те, що футбольне поле є велике – близько 100 метрів. Після знаходження матриці гомографії для

початкової позиції однієї камери потрібно оновлювати матрицю використовуючи інформацію про кути повороту камери там зум.

### 2.4.2 Визначення матриції гомографії автоматичним методом

Одне з перших досліджень є дослідження з використанням градієнтів країв зображення. Основною ідеєю дослідження є розробка нового методу визначення параметрів камери на основі градієнтного аналізу країв зображення.

У звичайному процесі калібрування камери необхідно використовувати точки на зображенні, що можуть бути важкі для визначення та можуть вимагати додаткової підготовки. В той же час, краї зображення, які можуть бути легко визначені, мають високу чутливість до змін у параметрах камери, що зробило їх об'єктом дослідження.

Автори пропонують використовувати градієнт країв зображення, щоб визначити параметри камери без необхідності використання точок. Метод визначення параметрів камери полягає у знаходженні оптимального співставлення між відповідними краями на двох зображеннях, використовуючи градієнти цих країв.

Дослідження проводилося на різних типах зображень та було продемонстровано, що метод може бути ефективним для визначення параметрів камери, особливо в ситуаціях, де точки на зображенні складно визначити, але краї зображення можуть бути легко знайдені.

Основним результатом дослідження є використання градієнтів країв зображень для визначення параметрів камери, що може бути корисним для розробки більш ефективних та точних методів калібрування камери у майбутньому. Приклад результату цього підходу зображено на рисунку 2.6.

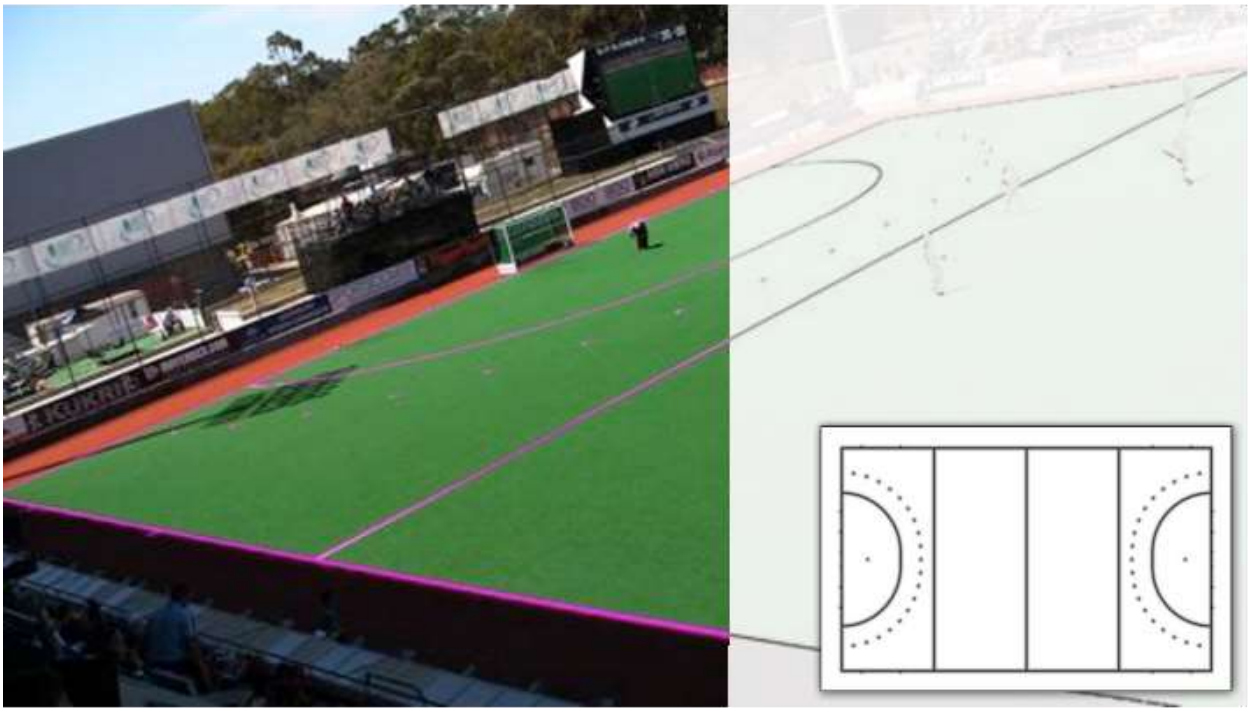

Рисунок 2.6 – Приклад результату градієнтного підходу

Ще одним підходом для автоматичного визначення матриці гомографії є розробка нейронної мережі, яка визначає позиції ключових точок на зображенні для знаходження гомографії.

Для цього використовується похожа архітектура моделі до UNet з використанням енкодера та декодера. U-Net – це глибокий нейронний мережевий архітектурний підхід для семантичної сегментації зображень, тобто для розділення зображення на підрегіони та призначення класу кожному під регіону. Він широко використовується у біомедичному зображенні, зокрема в задачах сегментації клітин, тканин, органів та патологічних змін.

Архітектура U-Net складається з двох основних частин: енкодера (encoder) та декодера (decoder). Енкодер складається з кількох згорткових шарів (convolutional layers) та пулінгових шарів (pooling layers), які зменшують

розмір зображення і збільшують кількість каналів. Ця частина мережі виконує функцію екстракції ознак, які потім використовуються декодером.

Декодер складається з транспонованих згорткових шарів (transposed convolutional layers) та конкатенації (concatenation) з попередніми виходами енкодера. Декодер поступово збільшує розмір зображення і зменшує кількість каналів, щоб отримати мапу сегментації. Конкатенація допомагає передати локальну інформацію з енкодера до декодера, що дозволяє отримати більш деталізовану мапу сегментації.

Узагальнюючи, енкодер забезпечує екстракцію ієрархії ознак, які можуть бути використані для сегментації зображення, а декодер забезпечує відтворення зображення з мапи сегментації, використовуючи інформацію з енкодера.

Далі, з використанням 91 ключової точки були розмічені зображення для тренування.  Також, звичайний обрахунок ваг для фільтрів в конвуляційній моделі був замінений на динамічне генерування фільтрів. Для цього кожна з 91 точок була перекодована в вектор і була використана для тренування декодер моделі.

Для вимірювання точності була використана метрика IoU. Використовуючи World Cup Dataset за допомогою алгоритму було досягнуто точності у 98%. Приклад результату роботи алгоритму зображено на рисунку 2.7.

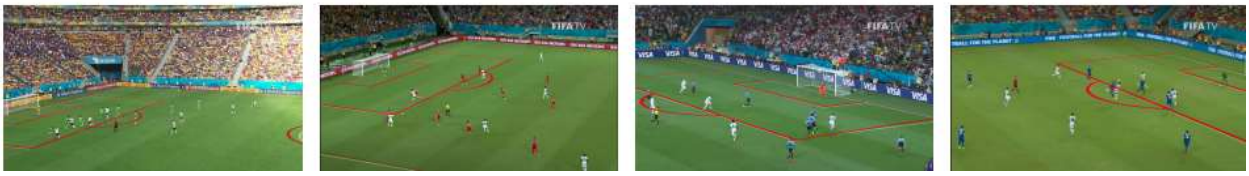

Рисунок 2.7 – Приклад результату роботи алгоритму з використанням нейронної мережі

Ще одним з підходів є визначення матриці гомографії є використання синтетичних даних. У цьому підході використовується подвійна GAN модель щоб отримати картинку з кутами поля.

GAN (Generative Adversarial Networks) – це нейромережеві моделі, які складаються з двох глибоких нейромереж: генератора (Generator) та дискримінатора (Discriminator). Генератор створює нові зображення, поки дискримінатор намагається відрізнити їх від реальних. Модель навчається в процесі протистояння між генератором і дискримінатором, де генератор намагається створити такі зображення, які не можуть бути відрізнені від реальних, а дискримінатор намагається розрізнити між реальними та синтезованими зображеннями. У процесі навчання, генератор та дискримінатор взаємодіють та навчаються один від одного, покращуючи свою якість та здатність створювати реалістичні зображення.

Генератор і дискримінатор – це конкурентні моделі, які працюють разом, щоб забезпечити навчання GAN. Генератор приймає на вхід випадковий шум або вектор, який використовується для створення нових зображень. Дискримінатор приймає на вхід зображення та визначає, чи є воно реальним або згенерованим.

Найбільш важливим елементом GAN є функція втрат (loss function), яка міряє рівень помилки генератора та дискримінатора. Функція втрат має бути налаштована на те, щоб дискримінатор відрізняв реальні зображення від згенерованих, та щоб генератор створював зображення, які були близькі до реальних. У процесі навчання GAN, генератор намагається змінити свої параметри, щоб зменшити помилку дискримінатора та підвищити якість своїх зображень.

Після отримання інформації з GAN моделі за допомогою HOG трансформацій отримується інформація про зображення, яка потім підбирає

найкращого відповідника з бази, в якій містяться схожі зображення та відповідні до них матриці гомографії.

HOG (Histogram of Oriented Gradients) є методом для визначення особливостей (фіч) у зображеннях, який широко використовується у комп'ютерному зорі та обробці зображень. В основі HOG лежить концепція того, що форма об'єктів можна визначити на основі орієнтацій градієнтів пікселів в зображенні.

Алгоритм HOG включає наступні етапи:

1. Нормалізація яскравості зображення: зображення змінюється таким чином, щоб краще виокремлювати особливості, тобто зменшується вплив різних освітлення на зображення.

2. Обчислення градієнтів: зображення перетворюється на відтінки сірого, і для кожного пікселя обчислюється градієнт як вектор, що показує напрям найбільшої зміни яскравості у точці.

3. Створення гістограми орієнтацій градієнтів: зображення поділяється на клітинки, а в кожній клітинці обчислюється гістограма орієнтацій градієнтів.

4. Створення блоків: клітинки групуються в блоки, і гістограми орієнтацій градієнтів для кожного блоку об'єднуються в один вектор.

5. Нормалізація блоків: кожен блок нормалізується таким чином, щоб усі блоки мали однакову вагу при подальшій обробці.

Отримані вектори вважаються особливостями зображення, які можуть використовуватися для розпізнавання об'єктів на зображенні.

Приклад роботи алгоритму зображено на рисунку 2.8.

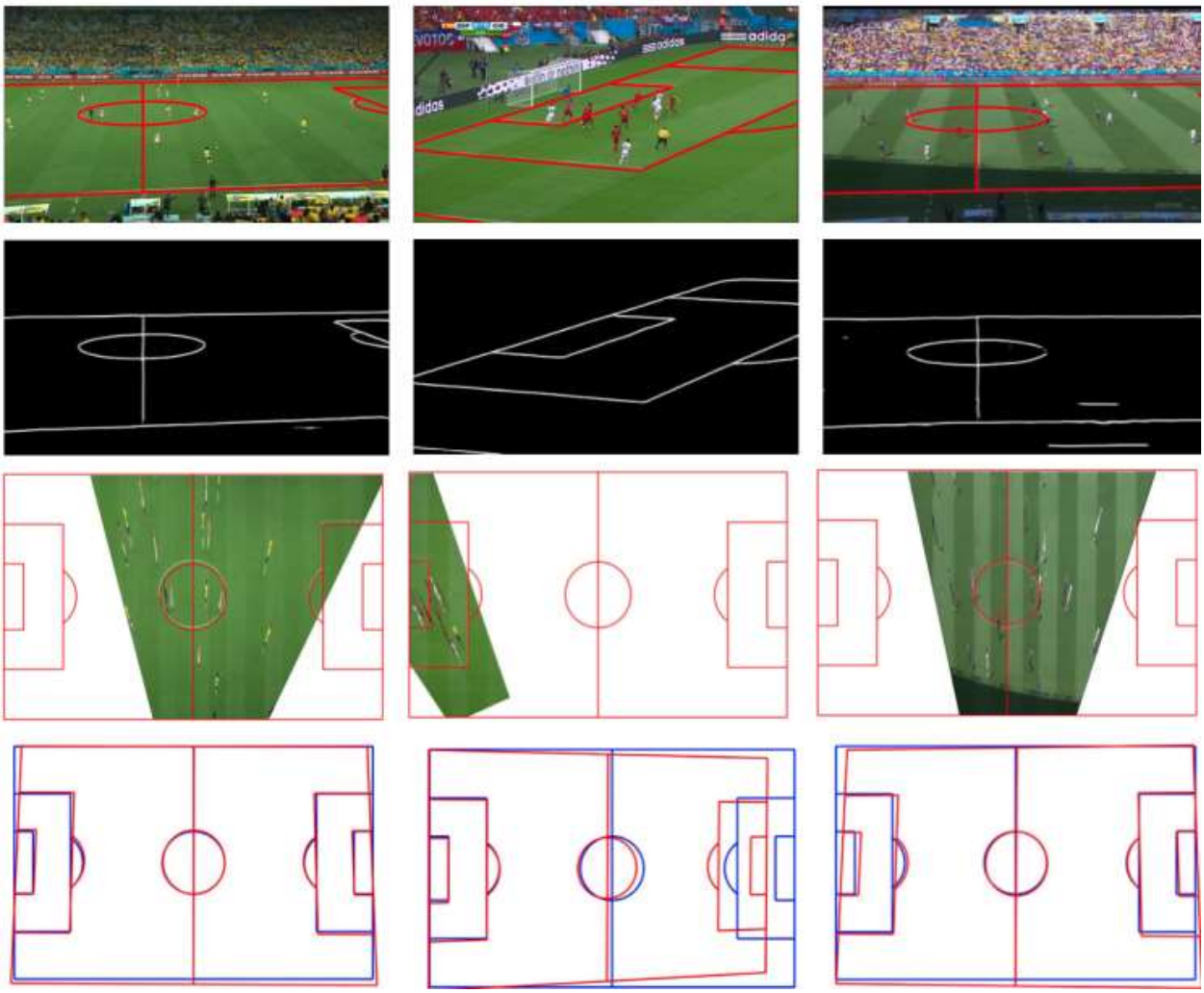

Рисунок 2.8 – Приклад визначення гомографії з використанням GAN моделей

На зображення видно, що модель не завжди максимально точно передбачає позицію поля. Максимальна точність цієї задачі виходить за рамки роботи.

### 2.5 Існуючі дослідження та підходи ре-ідентифікації гравців

Ре-ідентифікація гравців на полі є важливою задачею для аналізу футбольних матчів і розуміння того, як гравці рухаються та взаємодіють на

полі. Комп'ютерне бачення може використовуватись для автоматичної ідентифікації гравців на основі їх вигляду.

Для реіндетифікаї людей зазвичай використовують систему під назвою torchreid.

Torchreid – це відкрите програмне забезпечення (open-source software), яке забезпечує фреймворк для розробки та експериментів з алгоритмами ре-ідентифікації (re-identification) на базі зображень.

Це фреймворк забезпечує підтримку для численних датасетів, зокрема Market1501, DukeMTMC-reID, CUHK03 та інших. Крім того, він містить інструменти для обробки даних, побудови моделей, валідації результатів та візуалізації.

Основні можливості Torchreid:

1. Підтримка різноманітних архітектур нейронних мереж для ре-ідентифікації, включаючи ResNet, DenseNet, NASNet, EfficientNet тощо.
2. Підтримка різноманітних функцій втрат, включаючи cross-entropy loss, triplet loss, quadruplet loss, circle loss тощо.
3. Підтримка різноманітних методів для зменшення розмірності даних, включаючи Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), t-SNE тощо.
4. Можливість побудови змішаних моделей, які використовують зображення та відео одночасно для покращення результатів ре-ідентифікації.
5. Підтримка різноманітних методів для збору даних, включаючи ручну анотацію, автоматичну анотацію за допомогою алгоритмів комп'ютерного бачення тощо.
6. Можливість використання попередньо навчених моделей для досягнення кращих результатів на нових даних.

7. Підтримка різноманітних метрик для оцінювання результатів ре-ідентифікації, включаючи Cumulative Matching Characteristics (CMC) curve та mean average precision (mAP).

На базі цієї технології засновується технологія для реідентифікації гравців у футболі – SportsReID.

SportsReID є системою для ре-ідентифікації гравців у відео матчів з використанням комп'ютерного бачення. Система використовує технології глибокого навчання, зокрема нейронні мережі, для визначення ідентичності гравців у відео.

Для цього спочатку відео проходить обробку, під час якої з нього вилучаються кадри зі зображенням гравців та використовуються алгоритми для визначення ознак, таких як форма тіла, одяг, взуття та інші деталі зовнішності.

Отримані ознаки порівнюються з даними з бази даних, що містить зображення гравців та їхніх ідентифікаційні дані. Пошук відбувається за допомогою алгоритмів порівняння зображень, таких як гістограма орієнтованих градієнтів (HOG), глибинні нейронні мережі та інші.

SportsReID має застосування у великих командних спортивних заходах, де необхідно відстежувати рухи багатьох гравців одночасно. Система може допомогти в тренуванні команд, аналізувати гру та визначати найкращих гравців з метою підвищення результативності.

## 2.6 Висновки до другого розділу

У цьому розділі розглянуто два основні підходи для знаходження об’єктів – YOLO та R-CNN, що застосовуються для виявлення футбольних

гравців на зображеннях та відео. Розглянуті основні принципи робот 3D-и кожного з цих методів та їх переваги та недоліки.

Також було описано способи визначення позиції поля з використанням методу гомографії та інших автоматизованих підходів, що можуть бути корисними для футбольних матчів. Ці методи дозволяють з високою точністю відтворити модель поля з зображень та відео.

Нарешті, було досліджено методи реідентифікації гравців, які дозволяють відслідковувати гравців на полі та ідентифікувати їх у різних кадрах відео. Описані методи використовують різні підходи, такі як використання геометричних ознак та моделей глибокого навчання.

Застосування цих методів дозволяє створювати автоматизовані системи відеоаналізу для футбольних матчів, які можуть бути корисними для тренерів та аналітиків для отримання детальної статистики та аналізу гри.

# 3 РОЗРОБКА СИСТЕМИ ФОРМУВАННЯ СТАТИСТИЧНОЇ ІНФОРМАЦІЇ ПРО ПРОВЕДЕНІ СПОРТИВНІ МАТЧІ НА ОСНОВІ АНАЛІЗУ ВІДЕО ЗАПИСІВ

## 3.1 Основні вимоги до системи формування статистичної інформації про проведені спортивні матчі на основі аналізу відео записів

Система для формування статистичної інформації повинна вміти знаходити гравців та м’яч на відео, визначати їх позицію на полі та обраховувати певні метрики, які можуть бути використані тренерами в подальшому.

Система також має мати можливість бути інтегрованою в любий застосунок. Для успішної інтеграції системи у застосунки було обрано створення API доступу до системи, за яким можна отримати інформацію про переміщення гравців та обраховані статистичні дані.

Статистичні дані, які повинні бути в системі для кожного гравця:

- Час володіння м’ячем;
- Пройдена відстань;
- Кількість пасів;
- Кількість перехватів;
- Час проведений на своїй стороні;
- Час проведений на стороні противника;
- Середня швидкість.

Завдяки цим статистичним даним можна скласти повну картину про роботу гравця під час гри, а також, зрозуміти ключові переваги та недоліки тих чи інших стратегій.

## 3.2 Знаходження гравців та м’яча

Для тренування знаходження м’ячів було використано датасет з розміщеними гравцями та м’ячем. Гравці та м’яч виділялись прямокутниками, а координати прямокутників записувались у спеціальному форматі для тренування моделі. В результаті було відібрано 1000 картинок для тренування моделі, які містили, в середньому, 14.5 гравців в кадрі. Приклад датасету зображено на рисунку 3.1.


Рисунок 3.1 – Приклад датасету для знаходження гравців та м’яча

Для того, щоб обрати найкращу модель для задачі знаходження було проведено порівняння двох моделей – YOLO та R-CNN за такими параметрами:

- Архітектура: YOLO є одноступеневою архітектурою, тобто вона визначає об'єкти одразу на всьому зображенні, а R-CNN – багатоступеневою, тобто спочатку виділяє області, що містять об'єкти, а потім визначає їх.

- Швидкість: YOLO зазвичай працює швидше за R-CNN, тому що вона виконує обчислення на всьому зображенні одночасно, не потребуючи додаткових обчислень на областях, що не містять об'єкти.
- Точність: R-CNN зазвичай має вищу точність визначення об'єктів, тому що вона може використовувати більш складні методи визначення областей зображень, що містять об'єкти. Однак, за рахунок більш складної архітектури, вона працює повільніше за YOLO.
- Вимоги до обчислювальних ресурсів: YOLO зазвичай вимагає менше ресурсів, ніж R-CNN, оскільки вона має меншу кількість шарів та операцій. Це робить її популярнішою для використання на пристроях з обмеженими ресурсами, таких як мобільні телефони.
- Обробка відео: YOLO зазвичай краще працює з відео, оскільки вона може визначати об'єкти на кожному кадрі одночасно. R-CNN, натомість, потребує додаткового часу на обробку кожного кадру.

До того ж, для тренування YOLO моделей існує безліч готових рішень, які спрощують це завдання, тож основною моделлю для знаходження об’єктів було обрано саме її. На момент розробки було дві найпопулярніші моделі YOLO – v4 та v5. Обидві версії моделі мають переваги та недоліки:

- Архітектура: YOLOv4 має більш складну архітектуру за порівняння з YOLOv5, яка має більш просту та ефективну структуру.
- Швидкість: YOLOv5 зазвичай працює швидше за YOLOv4, тому що вона має меншу кількість обчислювальних операцій та шарів.
- Точність: YOLOv4 зазвичай має вищу точність визначення об'єктів, оскільки вона використовує більш складні методи навчання, такі як Scaled-YOLOv4 та YOLOv4-P5. Однак, за рахунок більш складної архітектури, вона працює повільніше за YOLOv5.

- Обробка зображень: YOLOv5 має більшу точність визначення об'єктів на зображеннях невеликого розміру, оскільки вона використовує методи обробки зображень з високою роздільною здатністю. YOLOv4, натомість, має тенденцію до перенавчання на невеликих зображеннях.
- Вимоги до обчислювальних ресурсів: YOLOv5 зазвичай вимагає менше ресурсів, ніж YOLOv4, оскільки вона має меншу кількість шарів та операцій. Це робить її популярнішою для використання на пристроях з обмеженими ресурсами, таких як мобільні телефони.

Тренування YOLOv5 потребує спеціальної структури даних. Приклад такої структури зображено на рисунку 3.2.

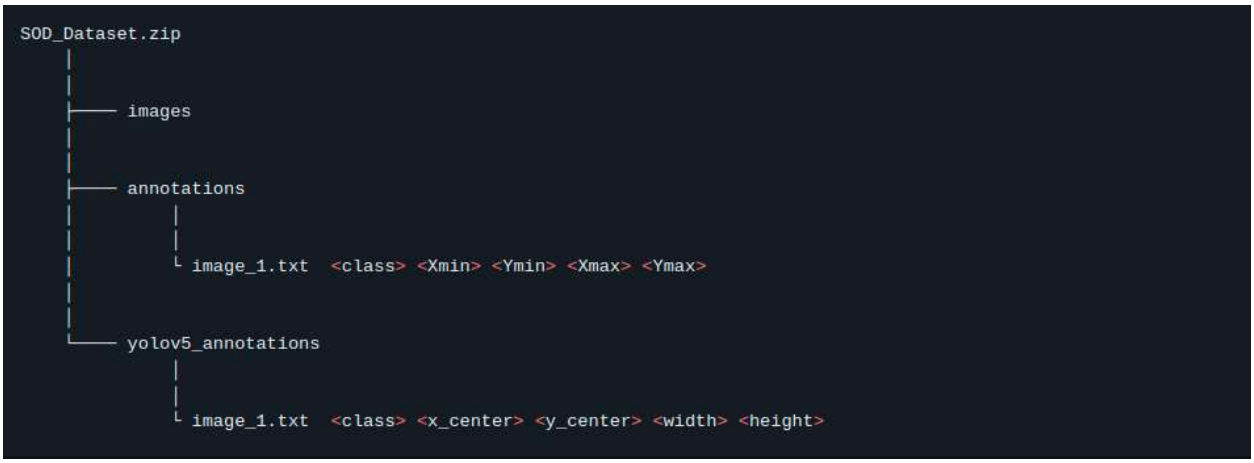


Рисунок 3.2 – Приклад структури даних для тренування моделі

Порівняльний результат тренування моделі зображено на рисунку 3.3.

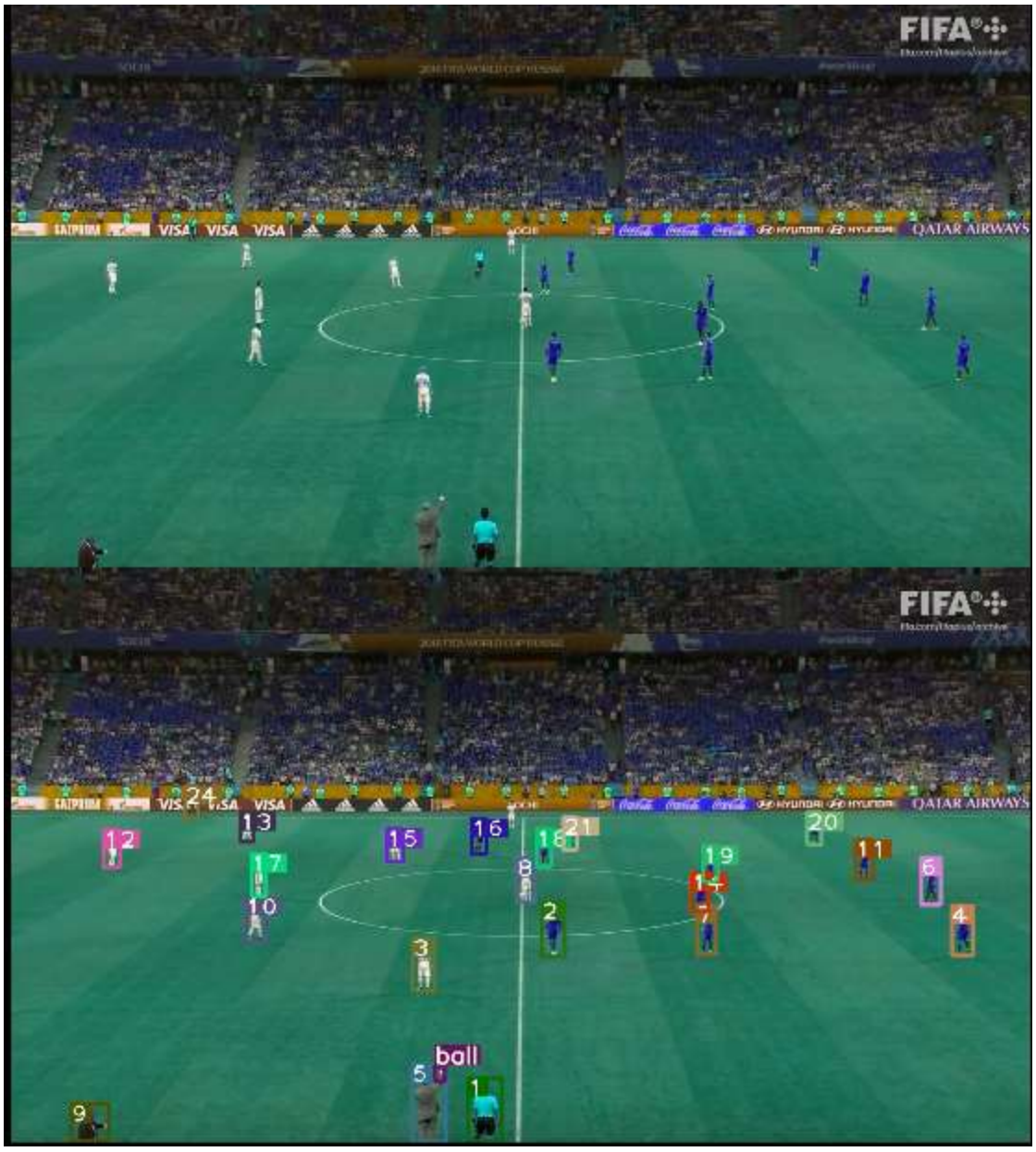


Рисунок 3.3 – Результат роботи моделі знаходження гравців та м’яча

Зображення зверху показує як виглядає відео до обробки моделлю, а зображеня внизу – після.

### 3.3 Визначення позиції гравців і м’яча на полі

Для визначення позиції гравців на полі, спершу, потрібно знайти матрицю гомографії. Для цього було обрано алгоритм, який використовує синтетичні дані для тренування, адже його точності достатньо для поставленої задачі. Приклад роботи алгоритму зображено на рисунку 3.4.

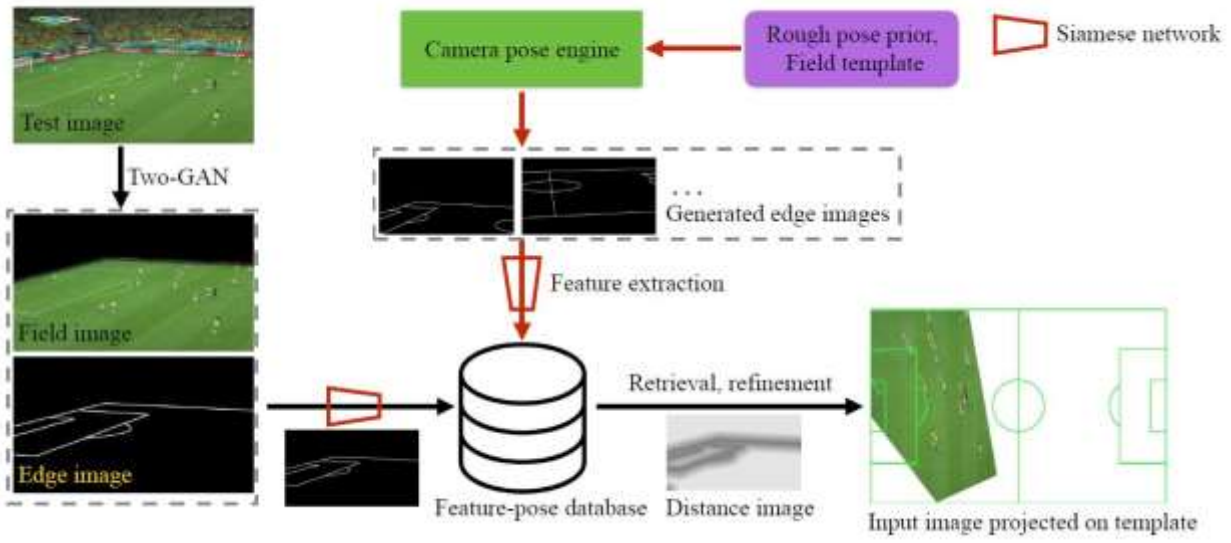


Рисунок 3.4 – Приклад роботи алгоритму визначення гомографії

Two-GAN модель складається з двох pix2pix моделей – одна з яких відповідає за сегментацію трави на полі, а інша за пошук ліній на полі.

Pix2pix є глибокою навчальною моделлю, яка використовується для генерації зображень з використанням умовних GAN (generative adversarial network). Вона здатна генерувати високоякісні зображення, які відповідають вхідним даним.

Модель pix2pix використовує пару зображень – вхідне та вихідне – для навчання. Наприклад, вхідним може бути чорно-біле зображення, а вихідним

– кольорове зображення. Модель навчається шукати залежність між вхідним та вихідним зображеннями з використанням умовних GAN. Приклад роботи такої моделі зображено на рисунку 3.5.

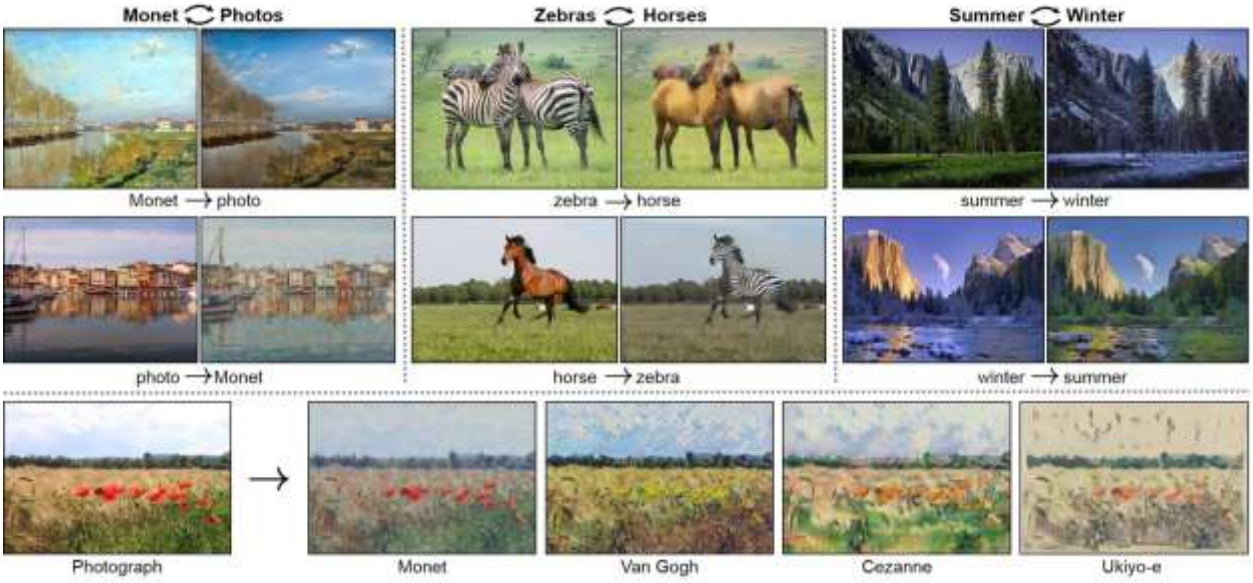


Рисунок 3.5 – Приклад роботи pix2pix моделі

В процесі навчання, pix2pix генерує зображення зі стохастичного шуму, які потім передаються на вхід дискримінатору, який розпізнає, чи є зображення правдоподібним. Окремо, зображення передається на вхід генератору, який створює нове вихідне зображення на основі вхідного зображення та вивчених залежностей.

Модель pix2pix є дуже гнучкою, оскільки вона може використовуватися для багатьох задач генерації зображень, таких як створення фотореалістичних зображень зі штучних описів, перетворення стилів зображень, генерація карти міста з зображення супутникової зйомки та інше.

Проте, для успішної роботи моделі pix2pix необхідно велику кількість даних для навчання та високопотужні обчислювальні ресурси, особливо, якщо використовувати великі зображення.

Приклад натренованої моделі поєднаної з моделлю знаходження гравців та м’яча зображено на рисунку 3.6.

Рисунок 3.6 – Результат об’єднання моделі знаходження гравців та моделі визначення матриці гомографії

Як видно з зображень алгоритм справляється доволі точно, враховуючи те, що використовується лише одна камера.

### 3.4 Ре-ідентифікація гравців

Для реідентифікації гравців використовується технологія Sportsreid яка тренована саме для реідентифікації гравців на різних кадрах. Sportsreid містить декілька різних видів моделей. Порівняльні характеристики моделей зображено в таблиці 3.1.

Для порівняння використовуються такі метрики як mAP та rank-1. mAP (mean average precision) та rank-1 – це метрики, які використовуються для оцінки ефективності алгоритмів комп'ютерного зору в задачах розпізнавання об'єктів або людей на зображеннях або відео.

mAP є метрикою, що вимірює середню точність розпізнавання об'єктів на зображенні. Вона враховує як точність знайдених об'єктів, так і їх кількість. Зазвичай, для обчислення mAP, спочатку визначається поріг для позначення об'єкту як знайденого або не знайденого, потім обчислюється точність розпізнавання (precision) та перевіряється, чи було знайдено правильну кількість об'єктів. Остаточний результат обчислюється як середнє значення точності для кожного порогу.

Rank-1 є метрикою, що вимірює точність розпізнавання людей на зображенні або відео. Вона вказує на те, який процент людей був розпізнаний правильно при порівнянні їх із базою даних. Ранжування відбувається за допомогою різних алгоритмів, наприклад, з використанням методу евклідової відстані або методу косинусної подібності між векторами ознак обличчя.

Як правило, ці метрики використовуються для порівняння ефективності різних алгоритмів та моделей розпізнавання об'єктів або людей, допомагаючи дослідникам з'ясувати, який з алгоритмів є найбільш ефективним для конкретної задачі.

Таблиця 3.1 – Порівняльна характеристика моделей

| Ім’я | Розмір | Розширення | mAP | rank-1 |
|---|---|---|---|---|
| ResNet50-fc512 | 24.6M | 256x128 | 81.8 | 76.1 |
| OSNet_x1_0 | 2.2M | 256x128 | 83.4 | 78.0 |
| DeiT-Tiny/16 | 5.5M | 224x224 | 82.2 | 76.2 |
| DeiT-S/16 | 21.7M | 224x224 | 84.3 | 79.4 |
| ViT-B/16 | 57.7M | 224x224 | 86.0 | 81.5 |
| ViT-L/16* | 303.6M | 224x224 | 89.8 | 86.7 |

Згідно з таблицею найоптимальнішим варіантом є модель з 2.2м параметрами та розміром зображення 256x128.

Для перевірки роботи реідентифікації було обрано два кадри з різницею в 100 кадрів.

Для кращої точності було обрано 10 сусідніх кадрів та використано знаходження об’єктів з трекінгом за допомогою DeepSort. Після визначених унікальних особистостей з  DeepSort. Далі, всі знайдені футболісти були обрізані та переведені у вектори за допомогою Sportsreid. Після цього відстань між всіма векторами була порахована за допомогою формули косинусоїдної відстані. Приклад роботи формули косинусоїдної відстані зображено на рисунку 3.8.

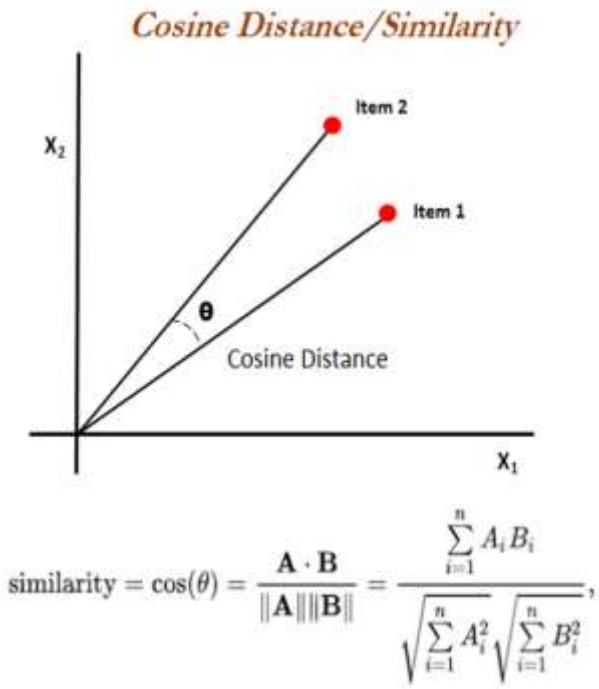


Рисунок 3.9 – Приклад роботи косинусоїдної відстані

Після знаходження відстані вибирається найкращий кандидат серед усіх та перевіряється чи відстань менше певного трешхолду. Якщо так, то кандидат залишається і процес продовжується. Результат знаходження кандидатів зображено на рисунках 3.10 та 3.11. Зверху зображені відповідники, а знизу – їх кандидати.

Рисунок 3.9 – Кандидати та їх відповідники

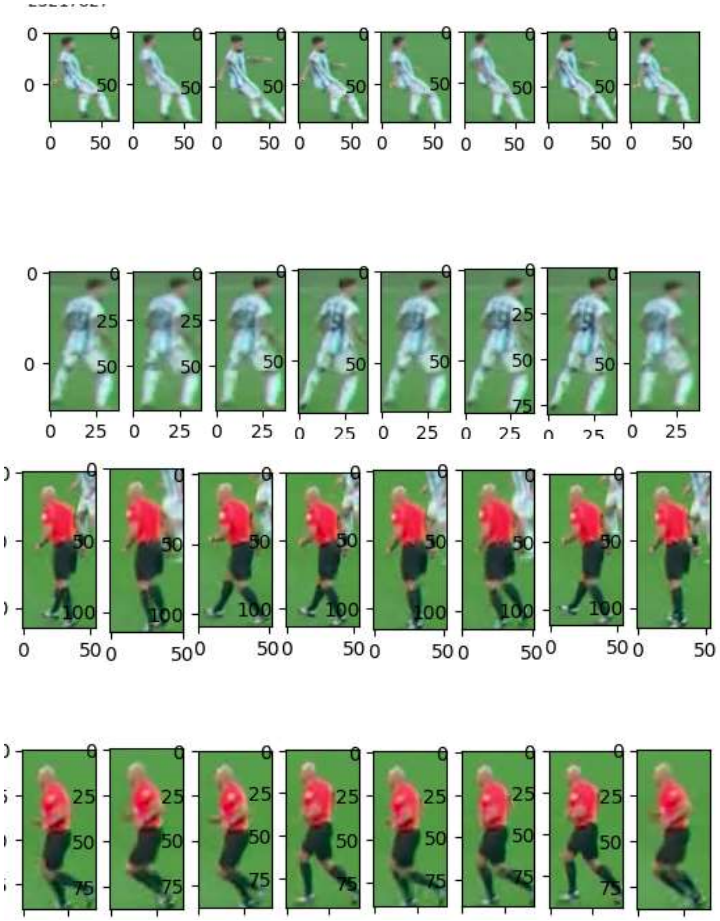

Рисунок 3.10 – Кандидати з їх відповідниками

В загальному між кадрами знайшов 9 відповідностей. Загалом точність системи достатня для початкових задач, але потребує доопрацювання в майбутньому.

### 3.5 Обрахунок статистичних даних

Для футбольного гравця, важливість різних метрик може залежати від ролі, яку він виконує на полі.

Час володіння м'ячем. Ця метрика вказує на те, скільки часу гравець утримує м'яч у своїх ногах. Це важливо для гравців, які відповідають за

розгортання атаки команди. Чим довше гравець утримує м'яч, тим більше часу він має на прийняття правильного рішення та передачу м'яча своїм партнерам. Щоб порахувати цю метрику використовується інформація про позицію м’яча в кадрі та гравця. Якщо м’яч знаходиться близько до гравця (30 піселів) або ж останні координати м’яча це координати ніг гравця то тоді час володіння м’ячем зараховується до гравця. Також враховується частота кадрів у відео для більш точного обрахування метрики.

Пройдена відстань. Ця метрика вказує на те, скільки метрів пройшов гравець протягом матчу. Це важливо для гравців, які відповідають за охоплення великої території на полі. Такі гравці, як нападники, захисники та півзахисники, повинні бути відповідальними за переміщення від своєї до противникової сторони та назад, щоб допомогти своїй команді у нападі та захисті. Дані для обрахунку цієї метрика збираються лише тоді, коли гравця видно у кадрі. Під час відео обраховується матриця гомографії, співставляється з переміщенням гравця та обраховує орієнтовну кількість метрів, які він пройшов.

Кількість пасів. Ця метрика вказує на те, скільки разів гравець віддав передачу своїм партнерам. Це важливо для гравців, які відповідають за організацію атак команди. Гравці пів захисту та нападу, зазвичай, повинні бути добрими пасистами, оскільки їхні передачі можуть вести до голів або інших гольових нагод. Щоб порахувати цю метрику використовується позиція гравців та м’яча. Якщо м’яч був близько до гравця (30 пікселів), або ж траєкторія м’яча розпочалась з ніг одного гравця команди і перейшла в близьку зону або ноги іншого гравця з тієї ж команди, то така подія зараховується як пас.

Кількість перехватів. Ця метрика вказує на те, скільки разів гравець зупинив атаку противника, перехопивши м'яч. Це важливо для гравців, які

відповідають за захист команди. Гравці захисту та півзахисту, зазвичай, повинні бути добрими перехоплювачами, оскільки їхня здатність зупинити атаки противника може бути критичною для успіху команди. Якщо м’яч був близько до гравця (30 пікселів), або ж траєкторія м’яча розпочалась з ніг одного гравця команди і перейшла в близьку зону або ноги іншого гравця з протилежної команди, то така подія зараховується як перехват.

Час проведений на своїй стороні та час проведений на стороні противника: Ці метрики вказують на те, скільки часу гравець провів на своїй стороні та на стороні противника. Це важливо для всіх гравців, оскільки воно допомагає розуміти, де та як гравець проводить свій час на полі та як його можна використати для досягнення успіху команди. Для обрахунку цих метрик використовується інформація про позицію на полі яка отримується за допомогою матриці гомографії.

Загалом, ці метрики допомагають гравцям і тренерам аналізувати та поліпшувати ефективність гравців та команди в цілому. І, якщо гравець може підвищити свої результати в одній з цих метрик, це може відразу ж позначитися на успішності команди.

### 3.6 Прикладний програмний інтерфейс (API)

API (Application Programming Interface) – це набір протоколів, інструментів та стандартів, які використовуються для розробки програмного забезпечення та забезпечення взаємодії між різними програмними компонентами.

API визначає, як різні компоненти програмного забезпечення повинні взаємодіяти між собою та які дії та операції можна виконувати з цих компонентів. Приклад повернення інформації API зображено на рисунку 3.11.


Рисунок 3.12 – Приклад інформації, що повертається з API

API програми реалізовано на Python з використанням Flask. Flask – це легкий фреймворк для створення веб-додатків на мові Python. API в цьому випадку реалізує наступні функції:

- Прийом відео: API може приймати відео, яке необхідно декодувати для подальшої обробки.
- Обробка відео: Закодоване відео обробляється з використанням моделей машинного навчання для отримання метрик та даних про гравців. Ці метрики включають час володіння м'ячем, пройдену відстань, кількість пасів, кількість перехоплень та інші характеристики, які є важливими.
- Повернення результатів: Останнім кроком API є повернення результатів обробки відео у вигляді JSON-об'єкту.

Загальна архітектура API включає в себе такі компоненти, як маршрутизатор, модель машинного навчання, базу даних та інші.

### 3.7 Висновки до третього розділу

У цьому розділі було описано використання різних технологій та моделей для розв'язання завдання аналізу футбольного матчу.

Модель Yolov5 була використана для знаходження гравців та м'яча на зображенні. Вона є швидкою та точною, тому є ідеальним варіантом для такого завдання.

Для визначення позиції гравців на полі були використані pix2pix нейронні мережі. Вони були навчені на зображеннях гравців та їхніх позицій на полі, що дозволило їм точно визначати позиції гравців на нових зображеннях.

Технологія soccerreid була використана для реідентифікації гравців. Вона дозволяє відрізнити одного гравця від іншого на основі його вигляду та рухів.

Також, було описано створення API на Python з використанням Flask для отримання статистичної інформації про гравців та їхні дії на полі. Це дозволяє тренерам та аналітикам отримувати цінну інформацію для покращення стратегії та тактики команди. В цілому, використання цих технологій та моделей може допомогти підвищити ефективність та результативність команди на полі.

# 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

## 4.1 Вимоги електробезпеки у приміщеннях, де встановлені персональні комп'ютери та відеодисплейні термінали

Приміщення із робочими місцями користувачів комп'ютерів для забезпечення електробезпеки обладнання, а також для захисту від ураження електричним струмом самих користувачів ПК повинні мати достатні технічні засоби захисту відповідно до ГОСТ 12.1.009-76, НПАОП 40.1-1.07-01 "Правила експлуатації електрозахисних засобів", НПАОП 40.1-1.21-98 "Правила безпечної експлуатації електроустановок споживачів", НПАОП 40.1-1.32-01 "Правила будови електроустановок. Електрообладнання спеціальних установок"

З метою запобігання ушкодженням, що можуть статися через ураження електричним струмом, загоряння, коротке замикання тощо, розроблено загальний стандарт безпеки IEC 950. Загальним стандартом електробезпечності для країн Європейської співдружності є Cemark.

Під час проектування систем електропостачання, монтажу силового електрообладнання та електричного освітлення будівель та приміщень для ПЕОМ необхідно дотримуватись вимог вищеназваних нормативно-правових актів, а також СН 357-77 "Инструкция по проектированию силового осветительного оборудования промышленных предприятий", затверджених Держбудом СРСР, ГОСТу 12.1.006, ГОСТу 12.1.030 "ССБТ. Электробезопасность. Защитное заземление, зануление", ГОСТу 12.1.019 "ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты", ГОСТу 12.1.045, ВСН 59-88 Держкомархітектури СРСР "Электрооборудование жилых и общественных зданий. Нормы проектирования", Правил пожежної безпеки в Україні, ДСанПіН 3.3.2.007-98,

розділів СНиП, що стосуються штучного освітлення і електротехнічних пристроїв, та вимог нормативно-технічної і експлуатаційної документації заводу-виробника ПЕОМ.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники тощо), електропроводи та кабелі за виконанням та ступенем захисту мають відповідати класу зони за ПУЕ, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів.

Під час монтажу та експлуатації ліній електромережі необхідно повністю унеможливити виникнення електричного джерела загоряння внаслідок короткого замикання та перевантаження проводів, обмежувати застосування проводів з легкозаймистою ізоляцією і, за можливості, перейти на негорючу ізоляцію.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

Використання нульового робочого провідника як нульового захисного провідника забороняється. Нульовий захисний провід прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення. Не допускається підключення на щиті до одного контактного затискача нульового робочого та нульового захисного провідників. Площа перерізу нульового робочого та нульового захисного провідника в груповій

трипровідній мережі повинна бути не менше площі перерізу фазового провідника.

Усі провідники повинні відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам ПУЕ.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

ПЕОМ, периферійні пристрої ПЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ повинні підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Штепсельні з'єднання та електророзетки крім контактів фазового та нульового робочого провідників повинні мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового та нульового робочого провідників. Порядок роз'єднання при відключенні має бути зворотним. Необхідно унеможливити з'єднання контактів фазових провідників з контактами нульового захисного провідника.

Неприпустимим є підключення ПЕОМ та периферійних пристроїв ПЕОМ до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення ПЕОМ, периферійних пристроїв слід виконувати за магістральною схемою, по 3…6 з’єднань або електророзеток в одному колі. Штепсельні з'єднання та

електророзетки для напруги 12 В та 36 В за своєю конструкцією повинні відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В і мають бути пофарбовані в колір, який візуально значно відрізняється від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах з урахуванням вимог ПУЕ та Правил пожежної безпеки в Україні.

Електромережу штепсельних розеток для живлення ПЕОМ, периферійних пристроїв ПЕОМ при розташуванні їх уздовж стін приміщення прокладають по підлозі поряд зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

При розташуванні в приміщенні за його периметром до 5 ПЕОМ, використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволяється прокладання їх без металевих труб та гнучких металевих рукавів.

Електромережу штепсельних розеток для живлення ПЕОМ при розташуванні їх у центрі приміщення, прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не дозволяється застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, що містять сірку. Відкрита прокладка кабелів під підлогою забороняється. Металеві труби та гнучкі металеві рукави повинні бути заземлені. Заземлення повинно відповідати вимогам НПАОП 40.1-1.21-98.

Для підключення переносної електроапаратури застосовують гнучкі проводи в надійній ізоляції.

Тимчасова електропроводка від переносних приладів до джерел живлення виконується найкоротшим шляхом без заплутування проводів у

конструкціях машин, приладів та меблях. Доточувати проводи можна тільки шляхом паяння з наступним старанним ізолюванням місць з'єднання.

Є неприпустимими:

- експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізольованими провідниками;
- застосування саморобних подовжувачів, які не відповідають вимогам ПВЕ до переносних електропроводок;
- застосування для опалення приміщення нестандартного (саморобного) електронагрівального обладнання або ламп розжарювання;
- користування пошкодженими розетками, розгалужувальними та з'єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;
- підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканиною та іншими горючими матеріалами, експлуатація їх зі знятими ковпаками (розсіювачами);
- використання електроапаратури та приладів в умовах, що не відповідають вказівкам (рекомендаціям) підприємств-виготовлювачів.

## 4.2 Підвищення стійкості роботи підприємств приладобудівної галузі у воєнний час

В умовах воєнних конфліктів стійкість та продуктивність підприємств приладобудівної галузі мають вирішальне значення для забезпечення потреб військових структур. З метою забезпечення неперервного функціонування цих підприємств у воєнний час виникає необхідність впровадження ефективних стратегій та заходів, спрямованих на підвищення їх стійкості. У цьому тексті

розглядаються різноманітні заходи, які можуть бути використані для забезпечення сталої роботи підприємств приладобудівної галузі в умовах воєнних дій. Ці заходи включають резервування виробничих потужностей, диверсифікацію постачальників, гнучкість виробництва, забезпечення безпеки та захисту, співпрацю з військовими структурами, навчання персоналу та розробку планів надзвичайних ситуацій. Ці заходи спрямовані на забезпечення продовження виробництва важливих приладів та обладнання, які є необхідними в умовах військових дій.

Приладобудівна галузь включає в себе широкий спектр приладів та обладнання, які відіграють важливу роль у воєнних операціях. Це можуть бути радіоелектронні пристрої, радари, оптичні прилади, системи зв'язку, військові датчики, навігаційні системи, аерокосмічна техніка, системи контролю та керування, медичне обладнання та багато іншого.

У воєнний час ці прилади використовуються для спостереження, розвідки, виявлення ворожих об'єктів, наведення озброєння, комунікації, медичного догляду та багатьох інших військових завдань. Вони допомагають підвищити ефективність та точність операцій, забезпечують безпеку військових сил, підвищують можливості збору та обробки інформації, та забезпечують швидкий обмін даними.

Завдяки впровадженню заходів, описаних нище, підприємства приладобудівної галузі зможуть підтримувати продуктивність та стійкість своєї роботи навіть у воєнний період. Резервування виробничих потужностей та диверсифікація постачальників забезпечать постачання необхідних матеріалів та компонентів для виробництва приладів. Гнучкість виробництва дозволить швидко переключатись на виробництво нових продуктів або модифікацію існуючих. Забезпечення безпеки та захисту зменшить ризик втрат та недоступності обладнання. Співпраця з військовими структурами

дозволить налагодити ефективний обмін інформацією та ресурсами. Навчання персоналу та розробка планів надзвичайних ситуацій підготують працівників до виконання завдань у воєнних умовах.

Забезпечення сталої роботи підприємств приладобудівної галузі у воєнний час відіграє критичну роль у забезпеченні обороноздатності та безпеки країни. Розвиток та вдосконалення цих заходів є постійним процесом, що дозволяє забезпечити високу готовність та ефективність приладобудівних підприємств у непередбачуваних умовах воєнного конфлікту.

Підвищення стійкості роботи підприємств приладобудівної галузі у воєнний час можна здійснити за допомогою наступних заходів:

1. Резервування виробничих потужностей: Одним з ключових заходів для забезпечення стійкості роботи підприємств приладобудівної галузі у воєнний час є створення резервних копій виробничого обладнання та запасів сировини. Це означає, що підприємства повинні мати додаткове обладнання та матеріали, які можуть бути швидко введені в експлуатацію у разі знищення або пошкодження основних активів. Такий підхід дозволяє забезпечити продовження виробництва навіть після можливих втрат.

2. Диверсифікація постачальників: Залежність від одного постачальника може становити великий ризик для підприємств у воєнний період, коли постачання може бути порушене або припинене. Тому важливо розширити базу постачальників та встановити додаткові джерела постачання матеріалів і компонентів. Це зменшить ризик простою виробництва і дозволить підприємствам швидко реагувати на зміни у виробничому ланцюжку.

3. Розвиток гнучкості виробництва: Гнучкість виробництва є ключовим чинником стійкості підприємств приладобудівної галузі у воєнний час. Це означає, що підприємства повинні бути здатні швидко переключатись

на виробництво інших продуктів або модифікувати існуючі, відповідаючи потребам воєнних структур. Впровадження гнучких систем виробництва, таких як "Lean production" або "Just-in-Time", дозволить забезпечити ефективну реорганізацію виробництва та мінімізувати затримки чи перерви.

4. Забезпечення безпеки та захисту: У воєнний час підприємства приладобудівної галузі стають потенційною мішенню для ворожих дій. Тому важливо звернути особливу увагу на фізичну та кібербезпеку виробничих об'єктів, інформаційних систем та даних. Забезпечення високого рівня безпеки та захисту допоможе уникнути можливих вторгнень, розголошення конфіденційної інформації та пошкоджень виробничих засобів.

5. Співпраця з військовими структурами: Підприємства приладобудівної галузі повинні підтримувати тісні зв'язки з військовими організаціями та урядовими установами. Це дозволить їм отримати необхідну підтримку та захист у воєнний час, а також активно співпрацювати замовниками військової техніки та обладнання. Така співпраця може включати спільний розвиток проектів, обмін інформацією та планування заходів відповідно до потреб воєнних структур.

6. Навчання та підготовка персоналу: У воєнний період персонал підприємств приладобудівної галузі повинен бути готовий ефективно діяти в умовах загострених ситуацій. Регулярне навчання та підготовка персоналу забезпечують необхідні знання і навички для ефективного виконання завдань та реагування на зміни у воєнному середовищі.

7. Розробка планів надзвичайних ситуацій: У воєнний період персонал підприємств приладобудівної галузі повинен бути готовий ефективно діяти в умовах загострених ситуацій. Регулярне навчання та підготовка персоналу забезпечують необхідні знання і навички для

ефективного виконання завдань та реагування на зміни у воєнному середовищі.

Для підвищення стійкості роботи підприємств приладобудівної галузі у воєнний час можна реалізувати наступні шляхи:

Забезпечення резервних копій виробничих потужностей є важливим аспектом. Це можна досягти шляхом проведення аудиту та інвентаризації обладнання та запасів, щоб визначити, які резервні копії необхідно мати. Також потрібно формувати фінансові резерви для придбання додаткового обладнання та запасів і встановлювати контрактні відносини з надійними постачальниками, які забезпечать оперативне постачання необхідних матеріалів та компонентів.

Диверсифікація постачальників також відіграє важливу роль у забезпеченні стійкості. Це означає, що підприємства повинні мати різні джерела постачання матеріалів і компонентів. Для досягнення цього можна провести маркетингове дослідження для виявлення альтернативних постачальників і укласти договори з різними постачальниками, щоб забезпечити різноманітність джерел постачання.

Розвиток гнучкості виробництва є ще одним важливим аспектом. Підприємства повинні мати здатність швидко переключатись на виробництво нових продуктів або модифікувати існуючі. Це можна досягти впровадженням гнучких систем виробництва, таких як "Lean production" або "Just-in-Time", підвищенням кваліфікації персоналу та використанням гнучких машин та обладнання.

Забезпечення безпеки та захисту є критичним аспектом у воєнний період. Підприємства повинні встановлювати системи відеоспостереження, контролю доступу та тривожних сигналізацій, щоб забезпечити безпеку

приміщень та обладнання. Також слід враховувати заходи щодо кібербезпеки, так як цифрові атаки можуть бути серйозною загрозою для підприємств.

Проведення навчань та тренувань персоналу є важливим елементом підвищення стійкості роботи підприємств. Працівники повинні бути підготовлені до дій у воєнних умовах та надзвичайних ситуаціях. Такі тренування можуть включати евакуаційні плани, навчання з технік безпеки та виробничих процесів у кризових ситуаціях.

Розробка планів надзвичайних ситуацій є ще одним кроком до підвищення стійкості підприємств. Ці плани повинні включати стратегії врегулювання проблем, швидку реакцію на кризові ситуації та координацію дій з військовими та урядовими структурами. Ретельно пророблені плани надзвичайних ситуацій забезпечують здатність підприємств до оперативного реагування і зменшують негативні наслідки в разі конфлікту.

Всі ці заходи сприятимуть забезпеченню стійкості роботи підприємств приладобудівної галузі під час воєнних дій та допоможуть забезпечити неперервність виробництва важливої військової техніки та обладнання.

# ВИСНОВКИ

У результаті виконання кваліфікаційної роботи магістра було досягнуто поставленої мети дослідження, а саме досліджено та розроблено систему автоматичного формування статистики про гравців з відео записів футбольних матчів, що забезпечуватиме ефективний та швидкий процес отримування статистичних даних про граців, що посприяє швидшому розвитку команди.

У ході виконання даного дослідження отримано наступні результати:

- Здійснено аналіз конкурентних рішень. Розглянуто такі конкуретні рішення як track160, spiideo та інші. На основі провденого огляду вдалося встановити, що ці рішення є дорогими та потребують додаткового обладнання для роботи, а рішень, які працюють лише з відео записів немає.
- На основі дослідження про існуючі підходи до вирішення поставлених задач визначено популярні моделі для знаходження обєктів – YOLO та R-CNN. Також розглянуто основні методи трекінгу об’єктів, такі як DeepSort. Також було розглянуто, що таке матриця гомографії, які є способи її знаходження та як, за допомогою неї отримати інформацію про гравців на полі.
- Здійснено порівняння існуючих підходів до вирішення поставлених задач, а саме порівняння моделей для знаходження об’єктів та обрано модель YOLO. Також, визначенно найкращий спосіб знаходження матриці гомографії для кожного кадру – а саме, з використовуючи технологію twoGAN моделі.
- Натреновано моделі машинного навчання необхідні для отримання статистики, такі як YOLO, DeepSort, Sportsreid та twoGAN.
- Обґрунтовано вибір технологій, для виконання проектованої системи.

- Розроблено автоматизоване формування статистики зі спортивних мачів базуючись на відео записах.

Подальше впровадження і розвиток відповідної автоматизованої системи оптимізує та вдосконалить наявні процеси у роботі тренера зменшить кількість помилок та пришвидшить розвиток команди. Результати можуть бути використані для подальших досліджень в області автоматизації формування статистики. Отримані результати також можуть бути використані для покращення роботи в інших сферах, пов'язаних з спортом.

# ПЕРЕЛІК ДЖЕРЕЛ

1. SSD: Single Shot MultiBox Detector [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1512.02325v5.pdf – Дата доступу: 14.04.2023.

2. YOLO vs SSD: Which One is a Superior Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://algoscale.com/blog/yolo-vs-ssd-which-one-is-a-superior-algorithm – Дата доступу: 14.04.2023.

3. YOLOv4: Optimal Speed and Accuracy of Object Detection [Науково-технічний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2004.10934 – Дата доступу: 14.04.2023. Дата публікації: 21.04.2020.

4. YOLOv5 Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.ultralytics.com/ – Дата доступу: 14.04.2023.

5. Computer Vision: Algorithms and Applications [Електронний ресурс] – Режим доступу до ресурсу: http://szeliski.org/Book/ – Дата доступу: 15.04.2023.

6. Homography Estimation: A Comprehensive Review [Електронний ресурс] – Режим доступу до ресурсу: https://ieeexplore.ieee.org/abstract/document/6909767/ – Дата доступу: 15.04.2023.

7. Object Detection using Deep Learning: A Comprehensive Survey [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1807.05511 – Дата доступу: 14.04.2023.

8. Soccereid: A Dataset for Soccer Player Re-Identification [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/2012.08395 – Дата доступу: 15.04.2023.

9. Torchreid: A Library for Deep Learning-based Person Re-Identification [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/KaiyangZhou/deep-person-reid – Дата доступу: 15.04.2023.

10. Building Web APIs with Flask [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.0.x/ – Дата доступу: 15.04.2023.

11. API Design: Crafting Interfaces that Developers Love [Електронний ресурс] – Режим доступу до ресурсу: https://www.amazon.com/API-Design-Crafting-Interfaces-Developers/dp/1492026921 – Дата доступу: 15.04.2023.

12. Soccernet: Soccer Statistics API [Електронний ресурс] – Режим доступу до ресурсу: https://developer.sportradar.com/docs/read/football/Soccer_v3 – Дата доступу: 15.04.2023.

13. Football Statistics and Analysis: A Comprehensive Overview [Електронний ресурс] – Режим доступу до ресурсу: https://link.springer.com/book/10.1007/978-3-319-98651-2 – Дата доступу: 15.04.2023.

14. Spiideo: Sports Video Analysis Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.spiideo.com/ – Дата доступу: 15.04.2023.

15. Track160 Dataset: A Benchmark for Multiple Object Tracking [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1905.01689 – Дата доступу: 15.04.2023.

16. Python [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/ – Дата доступу: 16.04.2023.

17. Визначення гомографії PapersWithCode [Електронний ресурс] – Режим доступу до ресурсу: https://paperswithcode.com/task/homography-

estimation#:~:text=Homography%20estimation%20is%20a%20technique,distortions%2C%20or%20perform%20image%20stitching. – Дата доступу: 16.04.2023.

18. SuperPoint: Self-Supervised Interest Point Detection and Description [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1712.07629v4.pdf – Дата доступу: 16.04.2023.

19. pytorch-two-GAN [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/lood339/pytorch-two-GAN – Дата доступу: 16.04.2023.

20. Sports Camera Calibration via Synthetic Data [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/abs/1810.10658 – Дата доступу: 16.04.2023.

21. Image-to-Image Translation with Conditional Adversarial Networks [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1611.07004.pdf – Дата доступу: 16.04.2023.

22. Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html – Дата доступу: 16.04.2023.

23. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e – Дата доступу: 16.04.2023.

24. Rich feature hierarchies for accurate object detection and semantic segmentation [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1311.2524.pdf – Дата доступу: 16.04.2023.

25. Fast R-CNN [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1504.08083.pdf – Дата доступу: 16.04.2023.

26. You Only Look Once: Unified, Real-Time Object Detection [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1506.02640v5.pdf – Дата доступу: 16.04.2023.

27. Lecture 11: Detection and Segmentation [Електронний ресурс] – Режим доступу до ресурсу: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf – Дата доступу: 16.04.2023.

28. Homography estimation of football field [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/code/s903124/homography-estimation-of-football-field – Дата доступу: 16.04.2023.

29. Homography from football (soccer) field lines [Електронний ресурс] – Режим доступу до ресурсу: https://stackoverflow.com/questions/60352448/homography-from-football-soccer-field-lines – Дата доступу: 16.04.2023.

30. Narya — Tracking and Evaluating Soccer Players [Електронний ресурс] – Режим доступу до ресурсу: https://paulgrn.medium.com/narya-tracking-and-evaluating-soccer-players-bbed888a9494 – Дата доступу: 16.04.2023.

31. A Robust and Efficient Framework for Sports-Field Registration [Електронний ресурс] – Режим доступу до ресурсу: https://openaccess.thecvf.com/content/WACV2021/papers/Nie_A_Robust_and_Efficient_Framework_for_Sports-Field_Registration_WACV_2021_paper.pdf – Дата доступу: 17.04.2023.

32. Homography-Estimation [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/sglbl/Homography-Estimation – Дата доступу: 17.04.2023.

33. Tracking soccer players based on homography among multiple views [Електронний ресурс] – Режим доступу до ресурсу:

https://www.researchgate.net/publication/221458111_Tracking_soccer_players_based_on_homography_among_multiple_views – Дата доступу: 17.04.2023.

34. Track160 [Електронний ресурс] – Режим доступу до ресурсу: https://www.track160.com/ – Дата доступу: 17.04.2023.

35. A Practical Guide to Person Re-Identification Using AlignedReID [Електронний ресурс] – Режим доступу до ресурсу: https://niruhan.medium.com/a-practical-guide-to-person-re-identification-using-alignedreid-7683222da644 – Дата доступу: 17.04.2023.

36. Torchreid: Deep Learning Person Re-identification in PyTorch [Електронний ресурс] – Режим доступу до ресурсу: https://morioh.com/p/3c548c4b1190 – Дата доступу: 17.04.2023.

37. sn-reid [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/SoccerNet/sn-reid – Дата доступу: 17.04.2023.

38. Top 5 Object Tracking Methods [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/augmented-startups/top-5-object-tracking-methods-92f1643f8435 – Дата доступу: 17.04.2023.

39. What is Object Tracking in Computer Vision? [Електронний ресурс] – Режим доступу до ресурсу: https://blog.roboflow.com/what-is-object-tracking-computer-vision/ – Дата доступу: 17.04.2023.

40. The Complete Guide to Object Tracking [Електронний ресурс] – Режим доступу до ресурсу: https://www.v7labs.com/blog/object-tracking-guide – Дата доступу: 17.04.2023.

41. DeepSORT — Deep Learning applied to Object Tracking [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/augmented-startups/deepsort-deep-learning-applied-to-object-tracking-924f59f99104 – Дата доступу: 18.04.2023.

42. deep_sort [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/nwojke/deep_sort – Дата доступу: 18.04.2023.

43. SIMPLE ONLINE AND REALTIME TRACKING WITH A DEEP ASSOCIATION METRIC [Електронний ресурс] – Режим доступу до ресурсу: https://arxiv.org/pdf/1703.07402.pdf – Дата доступу: 18.04.2023.

44. Object Tracking Using DeepSORT and YOLOv5 | Multi Object Tracking [Електронний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=NhCQBQqTAhE&ab_channel=CodeWithAarohi – Дата доступу: 18.04.2023.

45. What Is An API (Application Programming Interface)? [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses. – Дата доступу: 18.04.2023.

46. Best Python Framework to Create REST APIs [Електронний ресурс] – Режим доступу до ресурсу: https://python.plainenglish.io/best-python-framework-to-create-rest-apis-9f687e1261e1?gi=49724d8285c0 – Дата доступу: 18.04.2023.

47. Build Your Python Flask Application [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/bhavaniravi/build-your-1st-python-web-app-with-flask-b039d11f101c – Дата доступу: 18.04.2023.

48. Pyenv, manage your Python environment as an expert! [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/thedevproject/pyenv-manage-your-python-environment-as-an-expert-20ccfe0839ae – Дата доступу: 18.04.2023.

49. Global GPU Market [Електронний ресурс] – Режим доступу до ресурсу: https://vast.ai/ – Дата доступу: 18.04.2023.

50. Google Cloud Platform [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/ – Дата доступу: 18.04.2023.

# ДОДАТКИ

**Додаток А**

**Тези на наукову конференцію XI Міжнародної науково-практичної конференції молодих учених та студентів «Актуальні задачі сучасних технологій»**

**УДК 004.6**

**В. Лісовський, А. Зелінський, О. Сороківський**

Тернопільський національний технічний університет імені Івана Пулюя, Україна

## АНАЛІЗ ЗАДАЧ МАШИННОГО АНАЛІЗУ ЗОБРАЖЕННЯ ТА СУЧАСНИХ МЕТОДІВ ЇХ ВИРІШЕННЯ

**V. Lisovskyi, A. Zeliniskyi, O. Sorokivskyi**

## ANALYSIS OF MACHINE IMAGE ANALYSIS TASKS AND THEIR MODERN SOULTIONS

З кожним роком машинний аналіз покриває все більшу кількість сфер та аспектів буденного життя людей. Одним з значних проривів у сфері інформатики стала розробка моделей машинного навчання та штучного інтелекту, які дозволяють автоматизувати велику кількість задач. Комп’ютерне бачення, яке полягає у аналізі візуальної інформації машиною засобами штучного інтелекту, є яскравим прикладом сучасних методів аналізу зображень.

Задачі машинний аналізу зображень поділяються на наступні категорії:

- обробка зображень;
- розпізнавання об’єктів (також відома як класифікація об’єктів);
- сегментація зображень;
- оцінка пози.

Обробка зображень, в порівнянні з іншими, є простішою задачею аналізу, оскільки вона полягає в проведенні статистичного аналізу характеристик зображення, та застосуванні простих алгоритмічних змін на основі отриманої інформації. До цієї категорії належать такі техніки як вирівнювання гістограм тональностей (використовуючи гістограми розподілу зображення), зниження шуму (використовуючи або морфологічні фільтри, або лінійні-нелінійні фільтри, або нейронні мережі), фільтри зглаження, фільтри різкості та фільтри виявлення контурів. Методи з категорії обробки зображень переважно використовуються як початковий крок з покращення зображення в підготовці його до більш глибокого машинного аналізу.

Розпізнавання об’єктів – це складна система, яка полягає у зборі сукупності статистичних та візуальних характеристик зображення та їх подальшому аналізі на основі схожості набору характеристик до певних шаблонів (цільових об’єктів). Сучасним методом у розпізнаванні об’єктів є використання глибинних нейронних мереж, що використовують згорткові шари (фільтри). Їх використовують у таких задачах як:

- класифікація зображень, де проводиться огляд зображення в цілому та його класифікація до цільових класів (сучасні підходи до вирішення задачі – архітектури ResNet, EfficientNet, MobileNet, Inception та ін.);
- розпізнавання об’єктів, де проводить аналіз елементів зображення, знаходження окремих складових (цільових об’єктів) та їх класифікація до цільових класів (сучасні підходи до вирішення задачі – архітектури SSD, R-CNN, RetinaNet, YOLO та ін.).

Сегментація зображень полягає у розділенні зображення на полігональні сегменти, кожен з яких представляє собою певний об’єкт, який візуально відрізняється від інших сегментів. До методів, які проводять подібний аналіз зображень, належать пороговий аналіз, кластерний аналіз, контурний аналіз,

аналіз гістограм розподілу та використання глибнних нейронних мереж. Сегментація поділяється на три групи задач – семантична (кожен піксель класифікується до цільового класу), об’єктна (instance) (кожен піксель класифікується до цільового класу з розподілом на різні об’єкти) та комбінована (panoptic) (включає у себе особливості обох груп). До нейронних мереж, що розв’язують першу групу здача, входять U-Net, FCN, SegNet, DeepLab, EfficientDet та інші. До нейронних мереж другої групи входять Mask R-CNN, HTC, ConInst та інші.

Оцінка пози – це задача, направлена на аналіз позиції та положення об’єкта у просторі. Переважно це виконується завдяки розпізнаванню складових (орієнтирів) об’єкта на зображенні, після чого проводиться їх об’єднання у цільовий об’єкт, для якого вираховується агрегована позиція та положення у просторі. До моделей, які вирішують цю проблему, належать OpenPose, MoveNet, PoseNet, AlphaPose та інші.

Отже, великий спектр задач пов’язаних з аналізом зображення сьогодні доволі успішно розв’язуються машиною з використанням глибинних нейронних мереж. Кожна категорія задач має свій широкий набір методів для розв’язку, які мають свої недоліки, проте значно полегшують процес автоматизації виконання задач та знижують постійну потребу у людському нагляді.

**Перелік використаної літератури:**

1. Yağmur Çiğdem Aktaş. A Comprehensive Guide to Image Processing: Fundamentals, 2021. https://towardsdatascience.com/image-processing-4391c5bcef78

2. Neetu Rani. Image Processing Techniques: A Review, 2017. https://www.researchgate.net/publication/345364858_Image_Processing_Techniques_A_Review

3. Salwa Khalid Abdulateef, Mohanad Dawood Salman. A Comprehensive Review of Image Segmentation Techniques, 2021. https://www.researchgate.net/publication/354846947_A_Comprehensive_Review_of_Image_Segmentation_Techniques

4. Rohit Josyula, Sarah Ostadabbas. A Review on Human Pose Estimation, 2021. https://arxiv.org/abs/2110.06877

**УДК 004.6**

**Сороківський Олександр, Я.В. Литвиненко, дотк. техн. наук., проф**

Тернопільський національний технічний університет імені Івана Пулюя, Україна

## ПОРІВНЯННЯ ПРЕТРЕНОВАНИХ МОДЕЛЕЙ ДЛЯ ДЕТЕКЦІЇ ОБ'ЄКТІВ

**Sorokivskyi Oleksandr, I.V. Lytvynenko, Doctor of Sc., Professor**

**Ternopil Ivan Pul'uj National Technical University, Ukraine**

## COMPARATION OF THE PRETRAINED MODELS FOR OBJECT DETECTION

Детекція об'єктів – це одне з найважливіших завдань комп'ютерного бачення, завдання якого знаходження об'єктів певних класів на заданих зображеннях. Для вирішення цієї не простої задачі існує два підходи:

1. Використання власноруч налаштованих фільтрів для виділення певних об'єктів на зображеннях. Цей спосіб не є ефективним, адже, зазвичай, об'єкти візуально відрізняються як і кольорами так і формую. Налаштовування фільтрів під кожен випадок може тривати дуже довго і не завжди успішно;
2. Використання моделей машинного навчання, які автоматично налаштовують безліч фільтрів використовуючи для цього промарковані зображення. Цей підхід є більш універсальним, адже він враховує всі можливі варіанти об'єктів, а також, володіє властивістю генералізації, що допомагає справлятись з раніше небаченими об'єктами.

Враховуючи те, що перший метод займає багато часу та потребує багато зусиль – для вирішення задачі детекції об'єктів буде використовуватись, в основному, другий спосіб.

Ця доповідь стосується теми «Порівняння претернованих моделей для детекції об'єтів». Для того, щоб отримати модель, яка здатна знаходити потрібні об'єкти на зображенні потрібно багато промаркованих даних, а також потужні обрахункові ресурси. Для економії ресурсів та для покращення

точності моделей використовується техніка під назвою «Передача навчання». Згідно з дослідженням Proper Reuse of Image Classification Features Improves Object Detection (1, Cristina Vasconcelos, Vighnesh Birodkar, Vincent Dumoulin, 2022) з правильним використанням попередньо тренованих моделей можна досягнути кращих результатів швидше чим без них.

Під час дослідження готових імплементацій моделей для детекції об’єктів було знайдено багато рішень які відрізняються як і датою публікації, точністю і швидкодією. Для вирішення поточної задачі не потрібно враховувати швидкодію, адже швидкість роботи не залежить від реального часу. Для порівняння точності моделей машинного навчання потрібно використовувати одні й ті ж дані. Одним з найпопулярніших відкритих наборів даних з готовими промаркованими об’єктами є Microsoft COCO (2, Tsung-Yi Lin Michael Maire Serge Belongie Lubomir Bourdev Ross Girshick, James Hays Pietro Perona Deva Ramanan C. Lawrence Zitnick Piotr Dollar, 2015) – це набір даних, який містить 328 тисяч картинок повсякденних об’єктів та людей.

Для порівняння було обрано такі моделі:

- InternImage;
- EVA;
- Co-DETR;
- FD-SwinV2-G.

Для кращого визначення моделі також буде враховуватись набір даних для семантичної сегментації ADE20K (3, Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso and Antonio Torralba. Computer Vision and Pattern Recognition (CVPR), 2017).

Для порівняння на COCO наборі даних використовувалась метрика box AP, а для порівняння на ADE20K – Validation mIoU.

На рисунку 1 зображена порівняльна таблиця моделей на обох наборах даних.

| Model name | COCO test-dev | ADE20K |
|---|---|---|
| InternImage | 65.0 | 62.9 |
| EVA | 64.7 | 62.3 |
| Co-DETR | 64.5 | - |
| FD-SwinV2-G | 64.2 | 61.4 |
| YOLOR-P6 | 55.4 | - |

Рисунок 1 – Порівняльна таблиця

Як видно з таблиці, більшість порівнюваних моделей дуже близькі по значеннях. Модель YOLOR-P6 має найгірший результат, але моделі YOLO є найпопулярнішими моделями, які використовуються для детекції об’єктів через те, що вони мають декілька конфігурацій з різною швидкодією, що дозволяє налаштувати їх під потрібну задачу.

Отже, є багато моделей та наборів даних для їх порівнянь. Попри те, що є безліч моделей для знаходження об’єктів потрібно використовувати ресурси саме на ті, які дають хороші результати на схожих задачах. Всі моделі, які розглянуті вартують подальшого дослідження вже в рамках поточної задачі знаходження спортивних об’єктів на кадрах відеозапису.

**Перелік використаної літератури:**

1. Cristina Vasconcelos, Vighnesh Birodkar, Vincent Dumoulin. Proper Reuse of Image Classification Features Improves Object Detection, 2022 https://arxiv.org/abs/2204.00484.

2. Microsoft COCO: Common Objects in Context, Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár, 2015, https://arxiv.org/abs/1405.0312.

3. Semantic Understanding of Scenes through the ADE20K Dataset, Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, Antonio Torralba, 2018, https://arxiv.org/abs/1608.05442.

4. InternImage: Exploring Large-Scale Vision Foundation Models with Deformable Convolutions, Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, Yu Qiao, 2022, https://arxiv.org/abs/2211.05778.

5. EVA: Exploring the Limits of Masked Visual Representation Learning at Scale, Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, Yue Cao, 2022, https://arxiv.org/abs/2211.07636v1.

6. DETRs with Collaborative Hybrid Assignments Training, Zhuofan Zong, Guanglu Song, Yu Liu, 2022, https://arxiv.org/abs/2211.12860v1.

7. Contrastive Learning Rivals Masked Image Modeling in Fine-tuning via Feature Distillation, Yixuan Wei, Han Hu, Zhenda Xie, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, Baining Guo, 2022, https://arxiv.org/abs/2205.14141v3.

8. You Only Learn One Representation: Unified Network for Multiple Tasks, Chien-Yao Wang, I-Hau Yeh, Hong-Yuan Mark Liao, 2021, https://arxiv.org/abs/2105.04206v1.

**Додаток Б**

## Програмний код системи

Лістинг файлу api.py

```
from flask import Flask, request, jsonify
from analyzer import Analyzer

app = Flask(__name__)

@app.route('/analyze_video', methods=['POST'])
def analyze_video():
    # Retrieve the video file from the request
    video_file = request.files['video']

    # Create an instance of the Analyzer class
    analyzer = Analyzer()

    # Analyze the video and calculate statistics
    statistics = analyzer.analyze_video(video_file)

    return jsonify(statistics)

if __name__ == '__main__':
    app.run()
```

Лістинг файлу analyzer.py

```
from pathlib import Path
import sys
import joblib

from models_opt import models_opt
import cv2

sys.path.append('coords_extraction')

from elements.yolo import YOLO
from elements.deep_sort import DEEPSORT
from elements.perspective_transform import Perspective_Transform
from elements.assets import transform_matrix, detect_color

from tqdm.auto import tqdm

import cv2
import numpy as np
```

```
class CoordsExtractor:
    def __init__(self) -> None:
        self.detector = YOLO(models_opt.yolov5_model,
models_opt.conf_thresh, models_opt.iou_thresh)
        self.deep_sort = DEEPSORT(models_opt.deepsort_config)
        self.perspective_transform =
Perspective_Transform(models_opt)

    def __process_frame(self, frame, M, additional_data,
find_homography=False):
        w = additional_data['w']
        h = additional_data['h']
        gt_h = additional_data['gt_h']
        gt_w = additional_data['gt_w']


        main_frame = frame.copy()
        yolo_output = self.detector.detect(frame)

        if find_homography or M is None:
            M, warped_image =
self.perspective_transform.homography_matrix(main_frame)

        deepsort_data = None
        field_coordinates = None

        if yolo_output:
            deepsort_data =
self.deep_sort.detection_to_deepsort(yolo_output, frame)
            field_coordinates = []

            for obj in deepsort_data:
                xyxy = [obj[0], obj[1], obj[2], obj[3]]
                x_center = (xyxy[0] + xyxy[2])/2
                y_center = xyxy[3]

                coords = transform_matrix(M, (x_center,
y_center), (h, w), (gt_h, gt_w))

                field_coordinates.append(coords)

            ball_detection = [detection for detection in
yolo_output if 'ball' in detection['label']]
```

```
        return deepsort_data, field_coordinates, ball_detection

    def __extract_frames(self, video_path):
        cap = cv2.VideoCapture(video_path)
        frames_count = cap.get(cv2.CAP_PROP_FRAME_COUNT)
        frames = [cv2.cvtColor(cap.read()[1], cv2.COLOR_BGR2RGB)
for i in range(int(frames_count))]
        cap.release()

        return frames

    def __extract_additional_data(self, frame):
        additional_data = {}
        additional_data['w'] = frame.shape[1]
        additional_data['h'] = frame.shape[0]

        additional_data['bg_ratio'] =
int(np.ceil(additional_data['w']/(3*115)))
        gt_img =
cv2.imread('coords_extraction/inference/black.jpg')
        additional_data['gt_img'] =
cv2.resize(gt_img,(115*additional_data['bg_ratio'],
74*additional_data['bg_ratio']))
        additional_data['gt_h'], additional_data['gt_w'], _ =
gt_img.shape

        return additional_data

    def __crop_by_bbox(self, frame, bbox):
        x_min = bbox[0]
        x_max = bbox[2]

        y_min = bbox[1]
        y_max = bbox[3]

        return frame[y_min: y_max, x_min: x_max]

    def process_video(self, video_path, frames_amount=None):
        frames = self.__extract_frames(video_path)

        if frames_amount is not None:
            frames = frames[:frames_amount]

        additional_data =
self.__extract_additional_data(frames[0])

        M = None
```

```
        find_homography = False

        for frame_idx, frame in tqdm(enumerate(frames),
total=len(frames)):
            frame_path = Path(f'bbox_frames_data/{frame_idx}')
            frame_path.mkdir(exist_ok=True, parents=True)

            if frame_idx == 0 or frame_idx // 5 == 0:
                find_homography = True
            else:
                find_homography = False

            deepsort_outputs, coords, ball_detection =
self.__process_frame(frame.copy(), M, additional_data,
find_homography)
            joblib.dump({'deepsort_outputs': deepsort_outputs,
'coords': coords, 'ball': ball_detection}, frame_path /
'frame_data.joblib')

            for bbox in deepsort_outputs:
                if len(bbox) < 5:
                    continue

                bbox_img = self.__crop_by_bbox(frame.copy(),
bbox[:4])
                bbox_img_path = frame_path / f'{bbox[4]}.png'
                cv2.imwrite(str(bbox_img_path),
cv2.cvtColor(bbox_img, cv2.COLOR_BGR2RGB))
```

## Лістинг файлу models_opt.py

```
from argparse import Namespace
from numpy import Inf as inf
from pathlib import Path

weights_path = Path('coords_extraction/weights')

models_opt = Namespace(
    aspect_ratio=1.0,
    batchSize=1,
    checkpoints_dir= "coords_extraction/weights",
    conf_thresh=0.5,
    dataset_mode='single',
```

```
deepsort_config='coords_extraction/deep_sort_pytorch/configs/dee
p_sort.yaml',
    display_id=1,
    display_port=8097,
    display_winsize=256,
    fineSize=1024,
    gpu_ids=[],
    how_many=1,
    init_type='normal',
    input_nc=3,
    iou_thresh=0.5,
    isTrain=False,
    loadSize=1024,
    max_dataset_size=inf,
    model='two_pix2pix',
    nThreads=2,
    n_layers_D=3,
    name="pytorch-two-GAN-models/soccer_seg_detection_pix2pix",
    ndf=64,
    ngf=64,
    no_dropout=False,
    no_flip=False,
    norm='batch',
    ntest=inf,
    output='inference/output',
    output_nc=1,
    outputfps=20,
    phase='test',
    resize_or_crop='resize_and_crop',
    save=True, serial_batches=False,
    source='00028.jpg',
    view=False,
    which_direction='AtoB',
    which_epoch='latest',
    which_model_netD='basic',
    which_model_netG='unet_256',
    yolov5_model='coords_extraction/weights/yolov5s.pt'
)
```

Лістинг файлу models_opt.py

```
from pathlib import Path
import joblib
import sys
import random
import io
```

```
sys.path.append('soccer_reid')
sys.path.append('coords_extraction')

from elements.assets import transform_matrix, detect_color

import cv2
from torchreid.utils import FeatureExtractor
from torchreid import metrics
import matplotlib.pyplot as plt
from matplotlib import patches
import numpy as np

from tqdm.auto import tqdm

import warnings
warnings.filterwarnings("ignore")


# In[2]:


extractor = FeatureExtractor(
    model_name='deit_t_16',
    model_path='soccer_reid/dwn_data/model.deit_tiny.pth.tar-
22',
    device='cuda',
    image_size = [224, 224]
)


# In[3]:


processed_frames_folder = Path('bbox_frames_data')
frames_foders = list(processed_frames_folder.glob('*'))
print(len(frames_foders))


# In[4]:


class Detection:
    def __init__(self, uid):
        self.uid = uid
        self.bbox_pathes = []
        self.framers_ids = []
```

```
    def add_bbox_path(self, bbox_path: Path):
        self.bbox_pathes.append(bbox_path)

    def add_framers(self, framers_ids):
        self.framers_ids.extend(framers_ids)
        self.framers_ids = list(set(self.framers_ids))

    def load_bboxes(self):
        random.shuffle(self.bbox_pathes)

        loaded_frames = []

        for i in range(100):
            if i >= len(self.bbox_pathes):
                break


loaded_frames.append(cv2.cvtColor(cv2.imread(str(self.bbox_pathe
s[i])), cv2.COLOR_BGR2RGB))

        return loaded_frames


# In[5]:


detections = {}

for frame in frames_foders:
    detected_objects_pathes = list(frame.glob('*.png'))
    detected_ids = [int(object_path.stem) for object_path in
detected_objects_pathes]

    for detected_object_path in detected_objects_pathes:
        uid = int(detected_object_path.stem)
        if uid in detections:
            detection = detections[uid]
        else:
            detection = Detection(uid)

        detection.add_bbox_path(detected_object_path)
        detection.add_framers(detected_ids)
        detections[uid] = detection


# In[6]:
```

```
def get_bboxes_dist(a_bboxes, b_bboxes, dist_func):
    a_features = extractor(a_bboxes).cpu()
    b_features = extractor(b_bboxes).cpu()

    distmat = metrics.compute_distance_matrix(a_features,
b_features, dist_func)
    distmat = distmat.numpy()

    return distmat.min()


# In[7]:


connected_detections = {}
alt_connected_detections = {}
used_detections = []
for detection_uid, detection in tqdm(detections.items()):
    if detection_uid in used_detections:
        continue
    detection_bboxes = detection.load_bboxes()

    for comp_detection_uid, comp_detection in
detections.items():
        if comp_detection_uid in detection.framers_ids:
            continue

        comp_detection_bboxes = comp_detection.load_bboxes()

        dist = get_bboxes_dist(detection_bboxes,
comp_detection_bboxes, 'cosine')
        if dist < 0.20:
            if detection_uid not in connected_detections:
                connected_detections[detection_uid] = list()
                alt_connected_detections[comp_detection_uid] =
detection_uid


connected_detections[detection_uid].append(comp_detection)
            used_detections.append(comp_detection_uid)

    used_detections.append(detection_uid)


# In[8]:
```

```
connected_detections


# In[9]:


alt_connected_detections


# In[10]:


player_id = 34

test_bboxes = connected_detections[player_id][0].load_bboxes()
test2_bboxes = detections[player_id].load_bboxes()

test_fig, test_axes = plt.subplots(2, min(len(test_bboxes),
len(test2_bboxes)), figsize=(30, 10))

for i in range(len(test_bboxes)):
    if i >= len(test_axes[0]):
        break

    test_axes[0][i].imshow(test_bboxes[i])

for i in range(len(test_bboxes)):
    if i >= len(test_axes[1]):
        break

    test_axes[1][i].imshow(test2_bboxes[i])

plt.show()


# In[11]:


cap = cv2.VideoCapture('test_data/arg_fra_2022_small.mp4')
frames_count = cap.get(cv2.CAP_PROP_FRAME_COUNT)
frames = [cv2.cvtColor(cap.read()[1], cv2.COLOR_BGR2RGB) for i
in range(int(frames_count))]
cap.release()
```

```
# In[12]:


def cluster_and_get_largest_non_green_color(image_rgb,
num_clusters=3, green_threshold=50):
    pixels = image_rgb.reshape(-1, 3)

    # Perform k-means clustering
    _, labels, centers = cv2.kmeans(
        pixels.astype(np.float32),
        num_clusters,
        None,
        criteria=(
            cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER,
            10,
            1.0
        ),
        attempts=10,
        flags=cv2.KMEANS_RANDOM_CENTERS
    )

    # Get the labels of the pixels belonging to the green
cluster
    green_cluster_label = np.argmax(
        centers[:, 1] - centers[:, 0] - centers[:, 2]
    )

    # Retrieve the non-green colors
    non_green_colors = [
        tuple(center.astype(int))
        for idx, center in enumerate(centers)
        if idx != green_cluster_label and center[1] >
green_threshold
    ]

    # Count the occurrences of each color
    color_counts = {}
    for color in non_green_colors:
        if color not in color_counts:
            color_counts[color] = 0
        color_counts[color] += 1

    # Find the color with the most pixels
    largest_color = max(color_counts, key=color_counts.get)

    return largest_color
```

```
# In[13]:


def figure_to_array(fig, frame_size):
    plt.savefig('temp.png', format='png', dpi=300,
bbox_inches='tight')

    img = cv2.imread('temp.png')
    return img


# In[14]:


video_size = (1548, 897)
out = cv2.VideoWriter('demo.mp4',
cv2.VideoWriter_fourcc(*'mp4v'), 15, video_size)


# In[16]:


get_ipython().run_line_magic('matplotlib', 'inline')

colors_by_uids = {}
frames_players_coords_data = []

for i in tqdm(range(int(frames_count))):
    frame = frames[i]
    frame_folder = [frame_folder for frame_folder in
frames_foders if i == int(frame_folder.stem)]

    if len(frame_folder) > 0:
        frame_data =
joblib.load(next(frame_folder[0].glob('*.joblib')))
        frames_players_coords_data.append(frame_data['coords'])
        if len(frame_data['deepsort_outputs']) > 0:

            fig, ax = plt.subplots(1)
            ax.imshow(frame)
            ax.spines['top'].set_visible(False)
            ax.spines['right'].set_visible(False)
            ax.spines['bottom'].set_visible(False)
            ax.spines['left'].set_visible(False)
            ax.get_xaxis().set_ticks([])
            ax.get_yaxis().set_ticks([])
```

```
            for bbox in frame_data['deepsort_outputs']:
                x_min, y_min, x_max, y_max, uid = bbox
                w = x_max - x_min
                h = y_max - y_min

                if uid in alt_connected_detections:
                    uid = alt_connected_detections[uid]

                if uid not in colors_by_uids:
                    bbox_img = frame[y_min:y_max, x_min:x_max]
                    try:
                        bbox_color =
np.array(cluster_and_get_largest_non_green_color(bbox_img)) /
255
                        colors_by_uids[uid] = bbox_color
                    except Exception as e:
                        print(e)
                        print(i)
                        bbox_color = [0, 0, 0]
                else:
                    bbox_color = colors_by_uids[uid]

                box = patches.Rectangle(
                    (x_min, y_min), w, h, edgecolor=bbox_color,
facecolor="none"
                )
                ax.add_patch(box)
                t = ax.text(x_min, y_min, uid, color='white',
fontsize=8)
                t.set_bbox(dict(facecolor=bbox_color, alpha=0.5,
edgecolor=bbox_color, pad=0.1))


            res_frame = figure_to_array(fig, video_size)
            out.write(res_frame)
            plt.clf()

out.release()
cv2.destroyAllWindows()
```

Лістинг файлів coords_extraction/elemnts/assets.py

```
import cv2
from sklearn.cluster import KMeans
import numpy as np
```

```
pallete = {'b': (0, 0, 128),
        'g': (0, 128, 0),
        'r': (255, 0, 0),
        'c': (0, 192, 192),
        'm': (192, 0, 192),
        'y': (192, 192, 0),
        'k': (0, 0, 0),
        'w': (255, 255, 255)}

color_for_labels = (2 ** 11 - 1, 2 ** 15 - 1, 2 ** 20 - 1)

def xyxy_to_xywh(*xyxy):
    """" Calculates the relative bounding box from absolute
pixel values. """
    bbox_left = min([xyxy[0].item(), xyxy[2].item()])
    bbox_top = min([xyxy[1].item(), xyxy[3].item()])
    bbox_w = abs(xyxy[0].item() - xyxy[2].item())
    bbox_h = abs(xyxy[1].item() - xyxy[3].item())
    x_c = (bbox_left + bbox_w / 2)
    y_c = (bbox_top + bbox_h / 2)
    w = bbox_w
    h = bbox_h
    return x_c, y_c, w, h


def xyxy_to_tlwh(bbox_xyxy):
    tlwh_bboxs = []
    for i, box in enumerate(bbox_xyxy):
        x1, y1, x2, y2 = [int(i) for i in box]
        top = x1
        left = y1
        w = int(x2 - x1)
        h = int(y2 - y1)
        tlwh_obj = [top, left, w, h]
        tlwh_bboxs.append(tlwh_obj)
    return tlwh_bboxs


def compute_color_for_labels(label):
    """
    Simple function that adds fixed color depending on the class
    """
    color = [int((p * (label ** 2 - label + 1)) % 255) for p in
color_for_labels]
    return tuple(color)
```

```
def draw_boxes(img, bbox, identities=None, offset=(0, 0)):
    for i, box in enumerate(bbox):
        x1, y1, x2, y2 = [int(i) for i in box]
        x1 += offset[0]
        x2 += offset[0]
        y1 += offset[1]
        y2 += offset[1]
        # box text and bar
        id = int(identities[i]) if identities is not None else 0
        color = compute_color_for_labels(id)
        label = '{}{:d}'.format("", id)
        t_size = cv2.getTextSize(label, cv2.FONT_HERSHEY_PLAIN,
2, 2)[0]
        cv2.rectangle(img, (x1, y1), (x2, y2), color, 3)
        cv2.rectangle(
            img, (x1, y1), (x1 + t_size[0] + 3, y1 + t_size[1] +
4), color, -1)
        cv2.putText(img, label, (x1, y1 +
                                 t_size[1] + 4),
cv2.FONT_HERSHEY_PLAIN, 2, [255, 255, 255], 2)
    return img


def transform_matrix(matrix, p, vid_shape, gt_shape):
    p = (p[0]*1280/vid_shape[1], p[1]*720/vid_shape[0])
    px = (matrix[0][0]*p[0] + matrix[0][1]*p[1] + matrix[0][2])
/ ((matrix[2][0]*p[0] + matrix[2][1]*p[1] + matrix[2][2]))
    py = (matrix[1][0]*p[0] + matrix[1][1]*p[1] + matrix[1][2])
/ ((matrix[2][0]*p[0] + matrix[2][1]*p[1] + matrix[2][2]))

    p_after = (int(px*gt_shape[1]/115) , int(py*gt_shape[0]/74))

    return p_after


# Color Detection with K-means
def detect_color(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    img = img.reshape((img.shape[1]*img.shape[0],3))

    kmeans = KMeans(n_clusters=2)
    s = kmeans.fit(img)

    labels = kmeans.labels_
    centroid = kmeans.cluster_centers_
    labels = list(labels)
```

```
    percent=[]

    for i in range(len(centroid)):
        j=labels.count(i)
        j=j/(len(labels))
        percent.append(j)

    detected_color = centroid[np.argmin(percent)]

    list_of_colors = list(pallete.values())
    assigned_color = closest_color(list_of_colors,
detected_color)[0]
    assigned_color = (int(assigned_color[2]),
int(assigned_color[1]), int(assigned_color[0]))

    if assigned_color == (0, 0, 0):
        assigned_color = (128, 128, 128)

    return assigned_color


# Find the closest color to the detected one based on the
predefined palette
def closest_color(list_of_colors, color):
    colors = np.array(list_of_colors)
    color = np.array(color)
    distances = np.sqrt(np.sum((colors-color)**2,axis=1))
    index_of_shortest = np.where(distances==np.amin(distances))
    shortest_distance = colors[index_of_shortest]

    return shortest_distance
```

Лістинг perspective_transform.py

```
import pyflann
import scipy.io as sio
import numpy as np
import cv2

from perspective_transform.util.synthetic_util import
SyntheticUtil
from perspective_transform.util.iou_util import IouUtil
from perspective_transform.util.projective_camera import
ProjectiveCamera
from arguments import Arguments
from perspective_transform.models.models import create_model
```

```
from PIL import Image
import os
import torch
import torchvision.transforms as transforms
import torch.backends.cudnn as cudnn

from perspective_transform.deep.siamese import BranchNetwork,
SiameseNetwork
from perspective_transform.deep.camera_dataset import
CameraDataset

device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")


class Perspective_Transform():
    def __init__(self, opt):
        self.query_index = 0
        self.current_directory = os.path.join(os.getcwd(),
'coords_extraction/weights')

        # Deep features
        deep_database_directory =
os.path.join(self.current_directory,
"data/deep/feature_camera_91k.mat")
        data=sio.loadmat(deep_database_directory)
        self.database_features = data['features']
        self.database_cameras = data['cameras']

        self.deep_model_directory =
os.path.join(self.current_directory, "deep_network.pth")
        self.net,self.data_transform =
self.initialize_deep_feature()

        self.model =
self.initialize_two_GAN(self.current_directory, opt)
        print('Perspective Transform model loaded!')


    def homography_matrix(self, image):
        image = cv2.resize(image,(1280,720)) # it shouldn't be
changed
        edge_map ,seg_map = self.testing_two_GAN(image)

        # generate deep features
        test_features = self.generate_deep_feature(edge_map)
```

```
        # World Cup soccer template
        data = sio.loadmat(os.path.join(self.current_directory,
"data/worldcup2014.mat"))
        model_points = data['points']
        model_line_index = data['line_segment_index']

        template_h = 74  # yard, soccer template
        template_w = 115

        # retrieve a camera using deep features
        flann = pyflann.FLANN()
        result, _ = flann.nn(self.database_features,
test_features[self.query_index], 1, algorithm="kdtree", trees=8,
checks=64)
        retrieved_index = result[0]

        """
        Retrieval camera: get the nearest-neighbor camera from
database
        """
        retrieved_camera_data =
self.database_cameras[retrieved_index]

        u, v, fl = retrieved_camera_data[0:3]
        rod_rot = retrieved_camera_data[3:6]
        cc = retrieved_camera_data[6:9]

        retrieved_camera = ProjectiveCamera(fl, u, v, cc,
rod_rot)

        retrieved_h =
IouUtil.template_to_image_homography_uot(retrieved_camera,
template_h, template_w)

        retrieved_image =
SyntheticUtil.camera_to_edge_image(retrieved_camera_data,
model_points, model_line_index,
                                                    im_h=720,
im_w=1280, line_width=2)


        """
        Refine camera: refine camera pose using Lucas-Kanade
algorithm
        """
        dist_threshold = 50
        query_dist = SyntheticUtil.distance_transform(edge_map)
```

```
        retrieved_dist =
SyntheticUtil.distance_transform(retrieved_image)

        query_dist[query_dist > dist_threshold] = dist_threshold
        retrieved_dist[retrieved_dist > dist_threshold] =
dist_threshold

        h_retrieved_to_query =
SyntheticUtil.find_transform(retrieved_dist, query_dist)

        refined_h = h_retrieved_to_query@retrieved_h
        Warp_img = cv2.warpPerspective(seg_map,
np.linalg.inv(refined_h), (115,74),
borderMode=cv2.BORDER_CONSTANT)

        return np.linalg.inv(refined_h), Warp_img


    def initialize_deep_feature(self):

        # 2: load network
        branch = BranchNetwork()
        net = SiameseNetwork(branch)

        if os.path.isfile(self.deep_model_directory):
            checkpoint = torch.load(self.deep_model_directory,
map_location=lambda storage, loc: storage)
            net.load_state_dict(checkpoint['state_dict'])

            # 3: setup computation device
        if torch.cuda.is_available():
            net = net.to(device)
            cudnn.benchmark = True


        normalize = transforms.Normalize(mean=[0.0188],
                                         std=[0.128])

        data_transform = transforms.Compose(
            [  transforms.ToTensor(),
            normalize,
            ]
        )

        return net,data_transform
```

```
    def generate_deep_feature(self, edge_map):
        """
        Extract feature from a siamese network
        input: network and edge images
        output: feature and camera
        """
        #parameters
        batch_size = 1
        model_name = self.deep_model_directory

        normalize = transforms.Normalize(mean=[0.0188],
                                         std=[0.128])

        data_transform = transforms.Compose(
            [transforms.ToTensor(),
            normalize,
            ]
        )

        #resize image
        pivot_image = edge_map
        pivot_image = cv2.resize(pivot_image ,(320,180))

        pivot_image = cv2.cvtColor(pivot_image,
cv2.COLOR_RGB2GRAY)

pivot_image=np.reshape(pivot_image,(1,pivot_image.shape[0],pivot
_image.shape[1]))
        # Note: assume input image resolution is 180 x 320 (h x
w)

        data_loader = CameraDataset(pivot_image,
                                    pivot_image,
                                    batch_size,
                                    -1,
                                    data_transform,
                                    is_train=False)

        # 2: load network
        branch = BranchNetwork()
        net = SiameseNetwork(branch)

        if os.path.isfile(model_name):
            checkpoint = torch.load(model_name,
map_location=lambda storage, loc: storage)
            net.load_state_dict(checkpoint['state_dict'])
        else:
```

```
            print('Error: file not found at
{}'.format(model_name))

        # 3: setup computation device
        if torch.cuda.is_available():
            net = net.to(device)
            cudnn.benchmark = True

        features = []

        with torch.no_grad():
            for i in range(len(data_loader)):
                x, _ = data_loader[i]
                x = x.to(device)
                feat = net.feature_numpy(x) # N x C
                features.append(feat)
                # append to the feature list


        features = np.vstack((features))

        return features


    def testing_two_GAN(self, image):

        image=Image.fromarray(image)
        osize = [256,256]
        cropsize = osize
        image=transforms.Compose([transforms.Resize(osize,
transforms.InterpolationMode.BICUBIC),

transforms.RandomCrop(cropsize),transforms.ToTensor(),
                                  transforms.Normalize((0.5, 0.5,
0.5),(0.5, 0.5, 0.5))])(image)
        image=image.unsqueeze(0)

        self.model.set_input(image)
        self.model.test()

        visuals = self.model.get_current_visuals()

        edge_map=visuals['fake_D']
        seg_map = visuals['fake_C']

        edge_map= cv2.resize(edge_map,(1280,720))
        seg_map= cv2.resize(seg_map,(1280,720))
```

```
        return edge_map , seg_map

    def initialize_two_GAN(self, directory, opt):
        # opt = Arguments().parse()
        opt.nThreads = 1   # test code only supports nThreads =
1
        opt.batchSize = 1  # test code only supports batchSize =
1
        opt.serial_batches = True  # no shuffle
        opt.no_flip = True  # no flip
        opt.continue_train = False

        self.model = create_model(opt)
        return self.model
```

Лістинг файлу yolo.py

```
import torch
import cv2
import numpy as np
from yolov5.models.experimental import attempt_load
from yolov5.utils.general import non_max_suppression

device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
classes = {0: 'player', 1: 'ball'}

class YOLO():
    def __init__(self,model_path, conf_thres, iou_thres):
        self.yolo_model = attempt_load(weights=model_path,
map_location=device)
        print("Yolo model loaded!")
        self.conf_thres = conf_thres
        self.iou_thres = iou_thres

    def detect(self,frame):
        """
            Input :
                    BGR image

            Output:
            yolo return list of dict in format:
                {   label   :  str
                    bbox    :  [(xmin,ymin),(xmax,ymax)]
                    score   :  float
                    cls     :  int
```

```
                }
        """
        img = cv2.resize(frame, (640,384))

        # Convert
        img = img.transpose((2, 0, 1))[::-1]  # HWC to CHW, BGR
to RGB
        img = np.ascontiguousarray(img)

        img = torch.from_numpy(img).to(device)
        img = img.float()/255.0  # 0 - 255 to 0.0 - 1.0
        if img.ndimension() == 3:
            img = img.unsqueeze(0)

        pred = self.yolo_model(img, augment=False)[0]
        pred = non_max_suppression(pred,
conf_thres=self.conf_thres, iou_thres=self.iou_thres,
classes=None)
        items = []

        if pred[0] is not None and len(pred):
            for p in pred[0]:
                if int(p[5]) in list(classes.keys()):
                    score =
np.round(p[4].cpu().detach().numpy(),2)
                    label = classes[int(p[5])]
                    xmin = int(p[0] * frame.shape[1] /640)
                    ymin = int(p[1] * frame.shape[0] /384)
                    xmax = int(p[2] * frame.shape[1] /640)
                    ymax = int(p[3] * frame.shape[0] /384)

                    item = {'label': label,
                            'bbox' : [(xmin,ymin),(xmax,ymax)],
                            'score': score,
                            'cls' : int(p[5])}

                    items.append(item)

        return(items)
```

Лістинг файлу coords_extraction/soccer_reid/models/deit.py

```
from __future__ import division, absolute_import
import warnings
import torch
from torch import nn
from torch.nn import functional as F
```

```
import torchvision
from .models_v2 import deit_huge_patch14_LS,
deit_large_patch16_LS, deit_base_patch16_LS

__all__ = ['deit_b_16', 'deit_t_16', 'deit_s_16', 'deit_bd_16',
'deit_td_16', 'deit_sd_16',
           'deit_bd_16_384', 'deit_h_14_ls', 'deit_l_16_ls',
'deit_b_16_ls']

pretrained_urls = {
}

class Identity(nn.Module):
    def __init__(self):
        super(Identity, self).__init__()

    def forward(self, x):
        return x

##########
# Network architecture
##########
class DeiT(nn.Module):
    """DeiT Network.
    """

    def __init__(
        self,
        num_classes,
        name,
        pretrained=True,
        feature_dim=768,
        loss='softmax',
        pretrained_21k=True,
        img_size=None,
        **kwargs
    ):
        super().__init__()

        # Create a pretrained deit model
        if name == 'deit_b_16':
            model = torch.hub.load('facebookresearch/deit:main',
'deit_base_patch16_224', pretrained=True)
        if name == 'deit_bd_16':
            model = torch.hub.load('facebookresearch/deit:main',
'deit_base_distilled_patch16_224', pretrained=True)
        if name == 'deit_s_16':
```

```
            model = torch.hub.load('facebookresearch/deit:main',
'deit_small_patch16_224', pretrained=True)
        if name == 'deit_sd_16':
            model = torch.hub.load('facebookresearch/deit:main',
'deit_small_distilled_patch16_224', pretrained=True)
        if name == 'deit_t_16':
            model = torch.hub.load('facebookresearch/deit:main',
'deit_tiny_patch16_224', pretrained=True)
        if name == 'deit_td_16':
            model = torch.hub.load('facebookresearch/deit:main',
'deit_tiny_distilled_patch16_224', pretrained=True)
        if name == 'deit_bd_16_384':
            model = torch.hub.load('facebookresearch/deit:main',
'deit_base_distilled_patch16_384', pretrained=True)
        if name == 'deit_h_14_ls':
            assert img_size[0] == img_size[1]
            img_size = img_size[0]
            assert img_size == 224
            model = deit_huge_patch14_LS(pretrained=True,
pretrained_21k=pretrained_21k, img_size=img_size)
        if name == 'deit_l_16_ls':
            assert img_size[0] == img_size[1]
            img_size = img_size[0]
            assert img_size == 224 or img_size == 384
            model = deit_large_patch16_LS(pretrained=True,
pretrained_21k=pretrained_21k, img_size=img_size)
        if name == 'deit_b_16_ls':
            assert img_size[0] == img_size[1]
            img_size = img_size[0]
            assert img_size == 224 or img_size == 384
            model = deit_base_patch16_LS(pretrained=True,
pretrained_21k=pretrained_21k, img_size=img_size)

        self.loss = loss
        self.feature_dim = model.head.weight.shape[1]

        # Replace the classification layer in the model with an
Identity layer
        model.head = Identity()
        # Distilled model has two heads
        if '_td_' in name or '_bd_' in name or '_sd_' in name:
            model.head_dist = Identity()
            print("Using distilled model")
            self.distilled = True
        else:
            self.distilled = False
```

```
        self.model = model

        # self.global_avgpool = nn.AdaptiveAvgPool2d(1)
        # # fully connected layer
        self.fc = self._construct_fc_layer(
            int(min(512, self.feature_dim)), self.feature_dim,
dropout_p=None
        )
        # self.fc = None

        # Add our own classification layer
        self.classifier = nn.Linear(self.feature_dim,
num_classes)

        # self._init_params()

    # def _make_layer(
    #     self,
    #     block,
    #     layer,
    #     in_channels,
    #     out_channels,
    #     reduce_spatial_size,
    #     IN=False
    # ):
    #     layers = []
    #
    #     layers.append(block(in_channels, out_channels, IN=IN))
    #     for i in range(1, layer):
    #         layers.append(block(out_channels, out_channels,
IN=IN))
    #
    #     if reduce_spatial_size:
    #         layers.append(
    #             nn.Sequential(
    #                 Conv1x1(out_channels, out_channels),
    #                 nn.AvgPool2d(2, stride=2)
    #             )
    #         )
    #
    #     return nn.Sequential(*layers)

    def _construct_fc_layer(self, fc_dims, input_dim,
dropout_p=None):
        if fc_dims is None or fc_dims < 0:
            self.feature_dim = input_dim
            return None
```

```
        if isinstance(fc_dims, int):
            fc_dims = [fc_dims]

        layers = []
        for dim in fc_dims:
            layers.append(nn.Linear(input_dim, dim))
            layers.append(nn.BatchNorm1d(dim))
            #layers.append(nn.ReLU(inplace=True))
            if dropout_p is not None:
                layers.append(nn.Dropout(p=dropout_p))
            input_dim = dim

        self.feature_dim = fc_dims[-1]

        return nn.Sequential(*layers)

    # def _init_params(self):
    #     for m in self.modules():
    #         if isinstance(m, nn.Conv2d):
    #             nn.init.kaiming_normal_(
    #                 m.weight, mode='fan_out',
nonlinearity='relu'
    #             )
    #             if m.bias is not None:
    #                 nn.init.constant_(m.bias, 0)
    #
    #         elif isinstance(m, nn.BatchNorm2d):
    #             nn.init.constant_(m.weight, 1)
    #             nn.init.constant_(m.bias, 0)
    #
    #         elif isinstance(m, nn.BatchNorm1d):
    #             nn.init.constant_(m.weight, 1)
    #             nn.init.constant_(m.bias, 0)
    #
    #         elif isinstance(m, nn.Linear):
    #             nn.init.normal_(m.weight, 0, 0.01)
    #             if m.bias is not None:
    #                 nn.init.constant_(m.bias, 0)

    def featuremaps(self, x):
        if not self.distilled:
            x = self.model(x)
        else:
            # In case of distilled, there will be two feature
vectors.
            # We just take average for now
```

```
            x, y = self.forward_features(x)
            x = (x + y)/2
        return x

    def forward(self, x, return_featuremaps=False):
        x = self.featuremaps(x)
        if return_featuremaps:
            return x
        # v = self.global_avgpool(x)
        v = x
        v = v.view(v.size(0), -1)
        if self.fc is not None:
            v = self.fc(v)
        if not self.training:
            return v

        y = self.classifier(v)
        if self.loss == 'softmax':
            return y
        elif self.loss == 'triplet':
            return y, v
        else:
            raise KeyError("Unsupported loss:
{}".format(self.loss))

#
# def init_pretrained_weights(model, key=''):
#     """Initializes model with pretrained weights.
#
#     Layers that don't match with pretrained layers in name or
size are kept unchanged.
#     """
#     import os
#     import errno
#     import gdown
#     from collections import OrderedDict
#
#     def _get_torch_home():
#         ENV_TORCH_HOME = 'TORCH_HOME'
#         ENV_XDG_CACHE_HOME = 'XDG_CACHE_HOME'
#         DEFAULT_CACHE_DIR = '~/.cache'
#         torch_home = os.path.expanduser(
#             os.getenv(
#                 ENV_TORCH_HOME,
#                 os.path.join(
#                     os.getenv(ENV_XDG_CACHE_HOME,
DEFAULT_CACHE_DIR), 'torch'
```

```
#                 )
#             )
#         )
#         return torch_home
#
#     torch_home = _get_torch_home()
#     model_dir = os.path.join(torch_home, 'checkpoints')
#     try:
#         os.makedirs(model_dir)
#     except OSError as e:
#         if e.errno == errno.EEXIST:
#             # Directory already exists, ignore.
#             pass
#         else:
#             # Unexpected OSError, re-raise.
#             raise
#     filename = key + '_imagenet.pth'
#     cached_file = os.path.join(model_dir, filename)
#
#     if not os.path.exists(cached_file):
#         gdown.download(pretrained_urls[key], cached_file,
quiet=False)
#
#     state_dict = torch.load(cached_file)
#     model_dict = model.state_dict()
#     new_state_dict = OrderedDict()
#     matched_layers, discarded_layers = [], []
#
#     for k, v in state_dict.items():
#         if k.startswith('module.'):
#             k = k[7:] # discard module.
#
#         if k in model_dict and model_dict[k].size() ==
v.size():
#             new_state_dict[k] = v
#             matched_layers.append(k)
#         else:
#             discarded_layers.append(k)
#
#     model_dict.update(new_state_dict)
#     model.load_state_dict(model_dict)
#
#     if len(matched_layers) == 0:
#         warnings.warn(
#             'The pretrained weights from "{}" cannot be
loaded, '
#             'please check the key names manually '
```

```
#             '(** ignored and continue **)'.format(cached_file)
#         )
#     else:
#         print(
#             'Successfully loaded imagenet pretrained weights
from "{}"'.
#             format(cached_file)
#         )
#         if len(discarded_layers) > 0:
#             print(
#                 '** The following layers are discarded '
#                 'due to unmatched keys or layer size: {}'.
#                 format(discarded_layers)
#             )


##########
# Instantiation
##########

def deit_b_16(num_classes=1000, pretrained=True, loss='softmax',
**kwargs):
    model = DeiT(
        num_classes,
        name='deit_b_16',
        pretrained=pretrained,
        feature_dim=768,
        loss=loss,
        **kwargs
    )
    return model

def deit_bd_16(num_classes=1000, pretrained=True,
loss='softmax', **kwargs):
    model = DeiT(
        num_classes,
        name='deit_bd_16',
        pretrained=pretrained,
        feature_dim=768,
        loss=loss,
        **kwargs
    )
    return model

def deit_bd_16_384(num_classes=1000, pretrained=True,
loss='softmax', **kwargs):
    model = DeiT(
```

```
        num_classes,
        name='deit_bd_16_384',
        pretrained=pretrained,
        feature_dim=768,
        loss=loss,
        **kwargs
    )
    return model

def deit_s_16(num_classes=1000, pretrained=True, loss='softmax',
**kwargs):
    model = DeiT(
        num_classes,
        name='deit_s_16',
        pretrained=pretrained,
        feature_dim=384,
        loss=loss,
        **kwargs
    )
    return model

def deit_sd_16(num_classes=1000, pretrained=True,
loss='softmax', **kwargs):
    model = DeiT(
        num_classes,
        name='deit_sd_16',
        pretrained=pretrained,
        feature_dim=384,
        loss=loss,
        **kwargs
    )
    return model

def deit_t_16(num_classes=1000, pretrained=True, loss='softmax',
**kwargs):
    model = DeiT(
        num_classes,
        name='deit_t_16',
        pretrained=pretrained,
        feature_dim=192,
        loss=loss,
        **kwargs
    )
    return model
def deit_h_14_ls(num_classes=1000, pretrained=True,
loss='softmax', pretrained_21k=True, img_size=None, **kwargs):
    model = DeiT(
```

```
        num_classes,
        name='deit_h_14_ls',
        pretrained=pretrained,
        feature_dim=192,
        loss=loss,
        pretrained_21k=pretrained_21k,
        img_size=img_size,
        **kwargs
    )
    return model
def deit_l_16_ls(num_classes=1000, pretrained=True,
loss='softmax', pretrained_21k=True, img_size=None, **kwargs):
    model = DeiT(
        num_classes,
        name='deit_l_16_ls',
        pretrained=pretrained,
        feature_dim=192,
        loss=loss,
        pretrained_21k=pretrained_21k,
        img_size=img_size,
        **kwargs
    )
    return model
def deit_b_16_ls(num_classes=1000, pretrained=True,
loss='softmax', pretrained_21k=True, img_size=None, **kwargs):
    model = DeiT(
        num_classes,
        name='deit_b_16_ls',
        pretrained=pretrained,
        feature_dim=192,
        loss=loss,
        pretrained_21k=pretrained_21k,
        img_size=img_size,
        **kwargs
    )
    return model
def deit_td_16(num_classes=1000, pretrained=True,
loss='softmax', **kwargs):
    model = DeiT(
        num_classes,
        name='deit_td_16',
        pretrained=pretrained,
        feature_dim=192,
        loss=loss,
        **kwargs
    )
    return model
```