**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Старший преподаватель департамента программной инженерии факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. В. Патнюхин  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  профессор департамента программной инженерии, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **КОМПЬЮТЕРНАЯ СТРАТЕГИЧЕСКАЯ ИГРА "NERV" НА UNITY**  **Пояснительная записка**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.04.01-01** **ТЗ-1-ЛУ** | | |
|  |  | |
| Исполнитель  студент группы БПИ175  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /И. О. Балбин/  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г. | |
|  | | |
|  | |  |

**Москва 2019**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| УТВЕРЖДЕН  RU. 17701729.04.01-01 81 01-1-ЛУ |  | |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | | **ПРИЛОЖЕНИЕ ДЛЯ АВТОМАТИЧЕСКОГО ДЕТЕКТИРОВАНИЯ**  **ШАХМАТНЫХ ФИГУР ДЛЯ ЗАПИСИ ХОДОВ**  **Пояснительная записка**  **RU. 17701729.04.01-01 81 01-1**  **Листов 32** | | | | |
|  | |  | | |
|  | | |
|  | | | | |
|  | | | |  |

**Москва 2019**

# ВВЕДЕНИЕ

## Наименование программы

Наименование программы: «Приложение для автоматического детектирования шахматных фигур для записи ходов» («Application for automatic detection of chess pieces to record moves»).

## Краткая характеристика области применения программы

Программа предназначена для игроков в шахматы. С его помощью игроки смогут в автоматическом режиме записать партию в шахматной нотации. Это позволит игрокам анализировать сыгранные партии и не отвлекаться на их запись вручную во время самой игры.

# ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Программа выполнена в рамках темы курсовой работы — «Приложение для автоматического детектирования шахматных фигур для записи ходов», в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия».

Основанием для разработки является приказ от ХХ.ХХ.2017 г. № ХХХ «ХХХХ» декана факультета компьютерных наук НИУ ВШЭ.

# ТЕХНИЧЕСКИЕ ХАРАКТЕТРИСТИКИ

## 3.1. Постановка задача на разработку программы

Программа должна обеспечить возможность выполнение следующих функций:

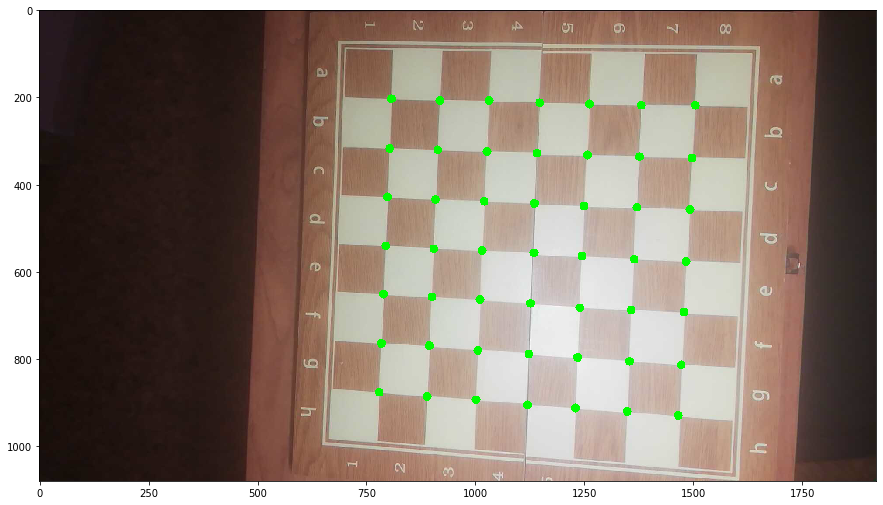
1. Подключение IP-видеокамеры по ее IP адресу
2. Автоматически детектировать доску по первому кадру из видеопотока с подключенной IP камеры
3. Детектировать начало и конец хода на шахматной доску
4. Детектировать изменение шахматной позиции и записывать ход

## 3.2. Описание алгоритмов и функционирование программы

Общий алгоритм детектирование шахматной доски и ходов представлен следующий диаграммой (рис. 1)

## 3.2.1 Детектирование доски

Для детектирования доски используется следующий алгоритм используется алгоритм из OpenCV – findChessBoardCorner. который возвращает внутреннюю углы на шахматной доске (рис. 2)



*Рис. 2 Зеленые точки – найденные внутренние углы на шахматной доске*.

При этом необходимо, чтобы доска была достаточно освещена (при условии недостаточной освещенности возникают шумы, которые мешают работе алгоритма).

Далее по полученной сетке возвращается восстанавливаются углы шахматной доски следующим кодом:

min\_x, max\_x = res[:, 0].min(), res[:, 0].max()

min\_y, max\_y = res[:, 1].min(), res[:, 1].max()

dx = (max\_x - min\_x) / 6

dy = (max\_y - min\_y) / 6

board\_corner = np.array([

[min\_x - dx, max\_y + dy],

[min\_x - dx, min\_y - dy],

[max\_x + dx, max\_y + dy],

[max\_x + dx, min\_y - dy],

])

res = np.vstack((res, board\_corner))

Минусы данного алгоритма, следующие:

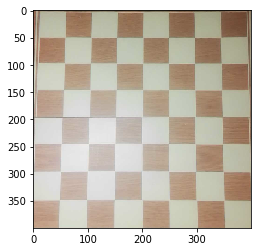
1. Нестабилен к шумам. Способен детектировать только пустую шахматную доску.
2. Способен детектировать только пустую шахматную доску.

Алгоритмы, более стабильные к шумам и способные находить доски с поставленным на нее фигурами описаны здесь <https://github.com/Elucidation/ChessboardDetect> и здесь <https://www.researchgate.net/publication/328461364_Chessboard_and_chess_piece_recognition_with_the_support_of_neural_networks>.

## 3.2.2 Обрезание шахматной доски из видеопотока

С помощью найденных углов шахматной доски getPerspectiveTransform из opencv мы высчитываем матрицу преобразование, и далее к каждому кадру ее применяем.

Пример на рис. 3



*Рис. 3 Обрезанная доска из рис.1*

Далее подразумевается, что при считывании кадра мы его обрезаем по шахматной доске

И преобразовываем в серую школы (RGB->GRAY SCALE).

## 3.2.2 Детектирование ходов

Для детектирование шахматных ходов используется алгоритм поиска руки над шахматной доской.

Мы храним изображение с текущей позицией без руки (frame\_last), и далее итеративно:

1. Считываем текущий кадр (frame)
2. С помощью алгоритма поиска руки, определяем есть ли она над доской
3. Если рука была над доской и пропала – значит совершился ход, иначе ход не был совершен
4. Если ход был совершен, то
   1. мы повторно считываем текущий кадр (frame)
   2. Определяем по frame и frame\_last какой ход был совершен
   3. Обновляем frame\_last – frame\_last = frame

Записываем ход

Алгоритм на python:

frame\_last = next(Camera.frame\_real())

frame\_last = cv2.cvtColor(frame\_last, cv2.COLOR\_BGR2GRAY)

hand\_on\_last = False

**for** frame **in** Camera.frame\_real():

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

hand\_on\_cur = Camera.\_hand\_on(frame, frame\_last)

hand\_on\_change = hand\_on\_cur ^ hand\_on\_last

hand\_on\_last = hand\_on\_cur

# print("hand on" if hand\_on\_cur else "hand off")

**if** hand\_on\_change **and** **not** hand\_on\_cur:

frame = next(Camera.frame\_real())

frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

cur\_move = Camera.\_get\_move(frame, frame\_last)

**print**(cur\_move)

Camera.board.push(cur\_move)

## 3.2.2.1 Определение руки над шахматной доской

Для детектирования руки над шахматной доской:

1. высчитаем абсолютную разницу между frame и frame\_last - diff.
2. Бинаризуем diff по threshold=150
3. С помощью морфологический операций с прямоугольным ядром чистим шумы:
   1. diff = erode(diff, kernel)
   2. diff = dilate(diff, kernel)
4. Опредялем есть ли рука над доской по проценту ненулевых элементов в отчищенной от шумов diff

Алгоритм на python:

**def** getFrameDiff(frame1, frame2):

diff = cv2.absdiff(frame1, frame2)

diff = cv2.threshold(diff, 10, 150, cv2.THRESH\_BINARY)[1]

kernel = np.ones((5, 5,))

diff = cv2.erode(diff, kernel)

diff = cv2.dilate(diff, kernel)

**return** diff

**def** handOn(frame\_cur, frame\_last, thr = 0.1):

diff = getFrameDiff(frame\_cur, frame\_last)

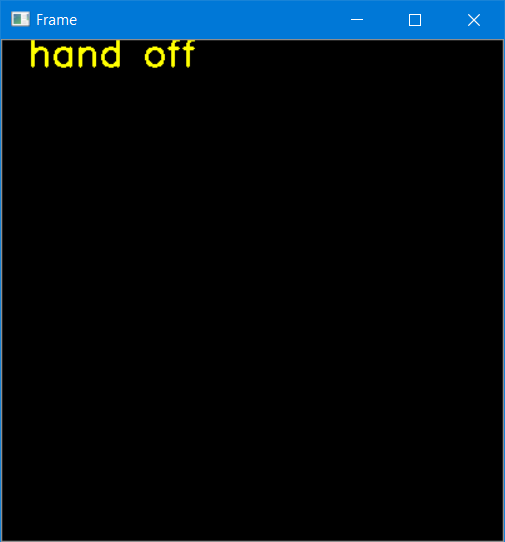
size = diff.shape[0]

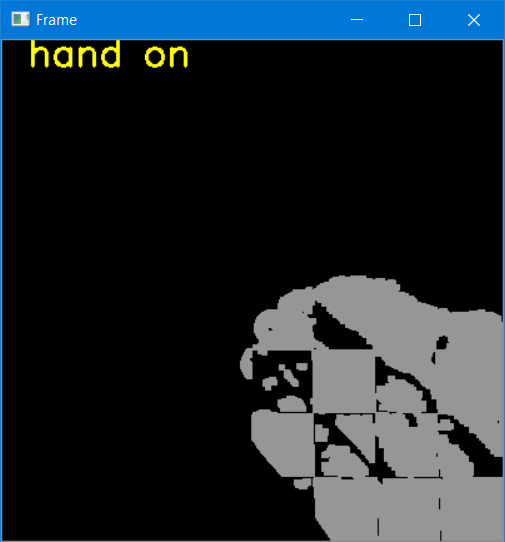
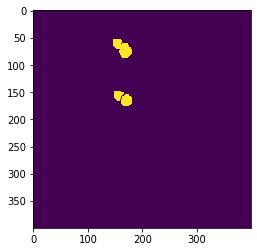
**return** (diff != 0).sum() / (size\*size) > thr

## 3.2.2.2 Определение шахматного хода

Для определения хода мы высчитаем отчищенную от шумов разницу между frame и last\_frame.

Далее по полученной разницу определяем в каких клетках произошли изменения (diff[i][j] > 0). Если в клетке произошли значительные изменения ( >0.1), значит эта клетка задействован в ходе.



  q  


Алгоритм на python:

**def** getMove(frame\_cur, frame\_last, board, thr = 0.1):

diff = getFrameDiff(frame\_cur, frame\_last)

step = diff.shape[0] // 8

change\_square = []

**for** i **in** range(8):

**for** j **in** range(8):

sub\_diff = diff[step\*i:step\*(i+1),step\*j:step\*(j+1)]

**if** (sub\_diff != 0).sum()/(step\*step) > thr:

change\_square.append([i, j])

moves = []

**for** j, i **in** change\_square:

**for** l, k **in** change\_square:

moves.append(chess.Move(i + j\*8, k + l\*8))

res\_move = None

**for** move **in** moves:

**for** beaut\_move **in** board.legal\_moves:

# if move in board.legal\_moves:

**if** beaut\_move == move:

res\_move = beaut\_move

**break**

**if** res\_move **is** **not** None:

**break**

**return** res\_move