SINGULARIS LAB

Подготовка к хакатону. Практика

Катаев Александр
Ведущий инженер-программист, к.т.н.
Алексеев Алексей
Инженер-программист
Singularis Lab, Ltd.

Детектирование лиц

```
image = cv2.imread(image_name)
cascade =
cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = cascade.detectMultiScale(gray)
for x, y, w, h in faces:
    cv2.rectangle(image, (x, y), (x + w, y + h), (255, 255, 0))
```

Детектирование лиц

```
image = cv2.imread(image_name)
cascade =
cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = cascade.detectMultiScale(gray)
for x, y, w, h in faces:
    cv2.rectangle(image, (x, y), (x + w, y + h), (255, 255, 0))
```

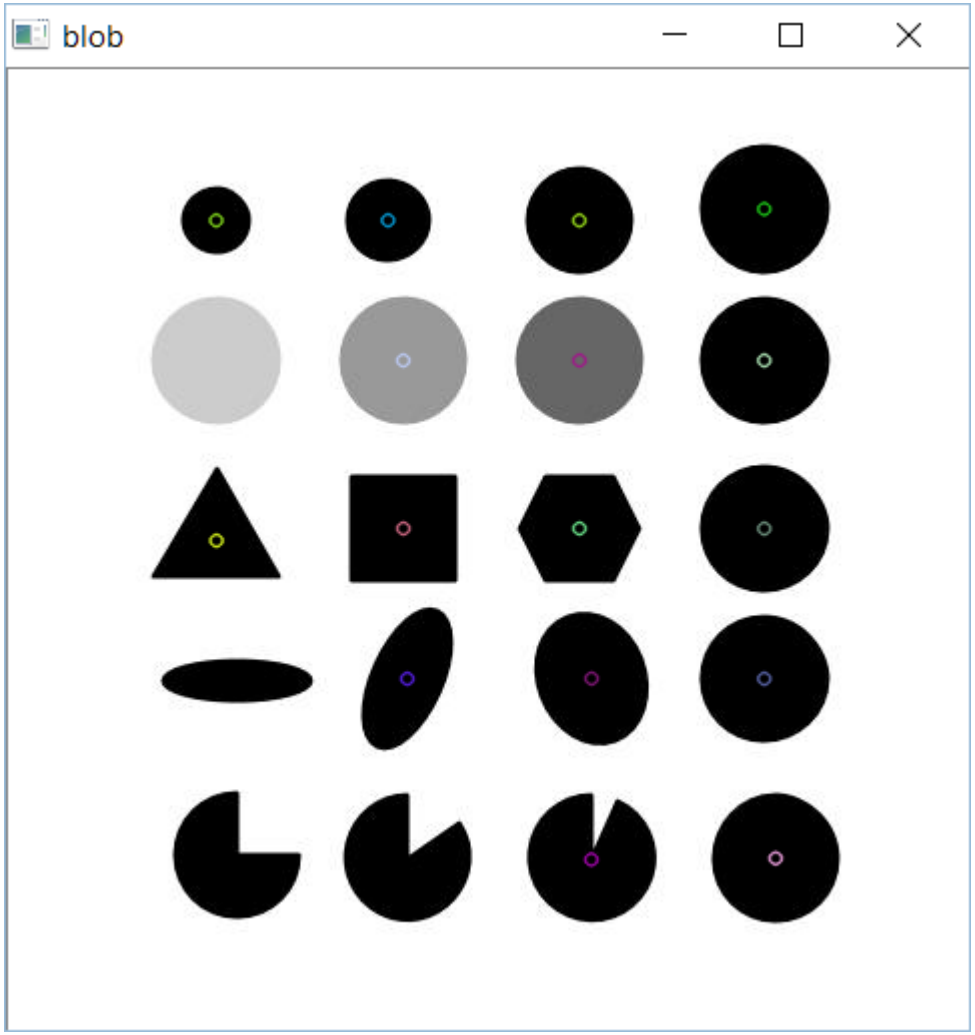
Детектирование блобов

```
image = cv2.imread(image_name)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
params = cv2.SimpleBlobDetector_Params()
detector = cv2.SimpleBlobDetector_create(params)

key_points = detector.detect(gray)
image = cv2.drawKeypoints(image, key_points, np.array([]))
cv2.imshow('blob', image)
cv2.waitKey()
```

Детектирование блобов

```
image = cv2.imread(image_name)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
params = cv2.SimpleBlobDetector_Params()
detector = cv2.SimpleBlobDetector_create(params)

key_points = detector.detect(gray)
image = cv2.drawKeypoints(image, key_points, np.array([]))
cv2.imshow('blob', image)
cv2.waitKey()
```

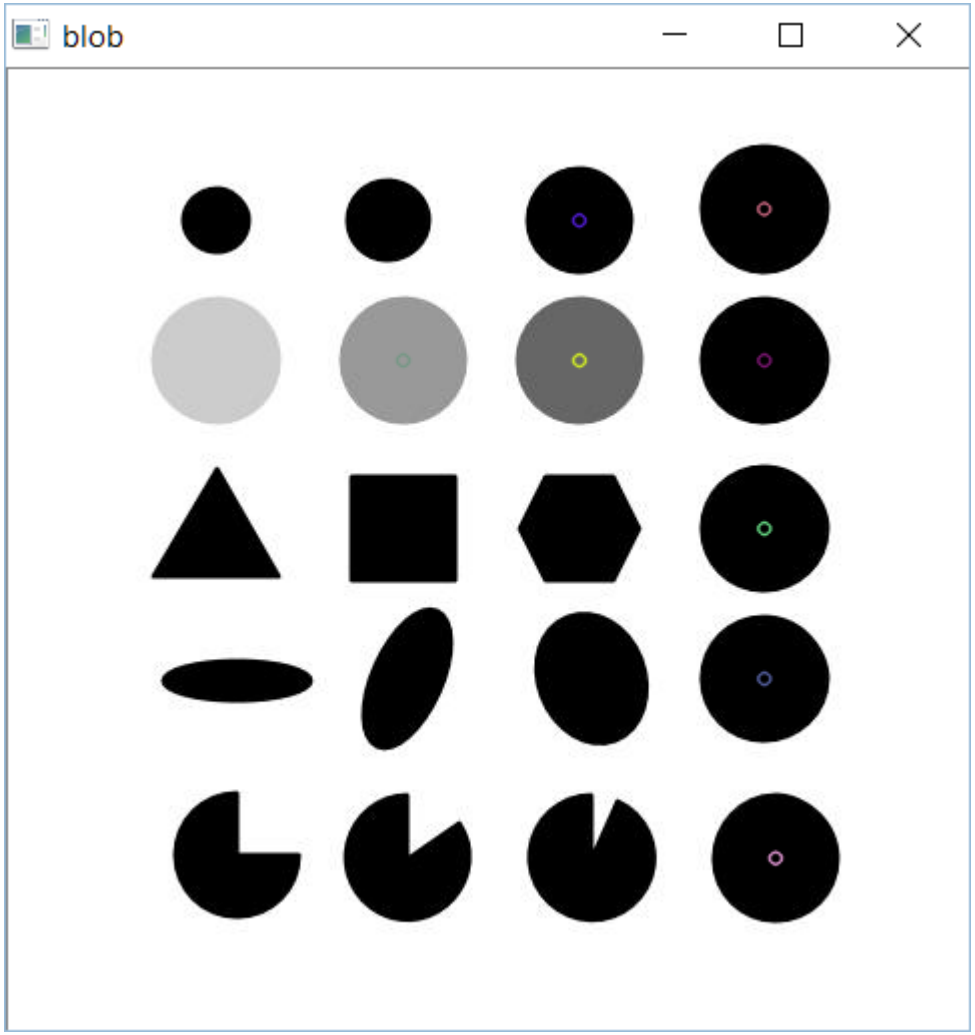
Детектирование блобов



Детектирование блобов. Параметры

```
params = cv2.SimpleBlobDetector_Params()
params.minThreshold = 10
params.maxThreshold = 200
params.filterByArea = True
params.minArea = 1500
params.filterByCircularity = True
params.minCircularity = 0.9
params.filterByConvexity = True
params.minConvexity = 0.87
params.filterByInertia = True
params.minInertiaRatio = 0.9
detector = cv2.SimpleBlobDetector_create(params)
```

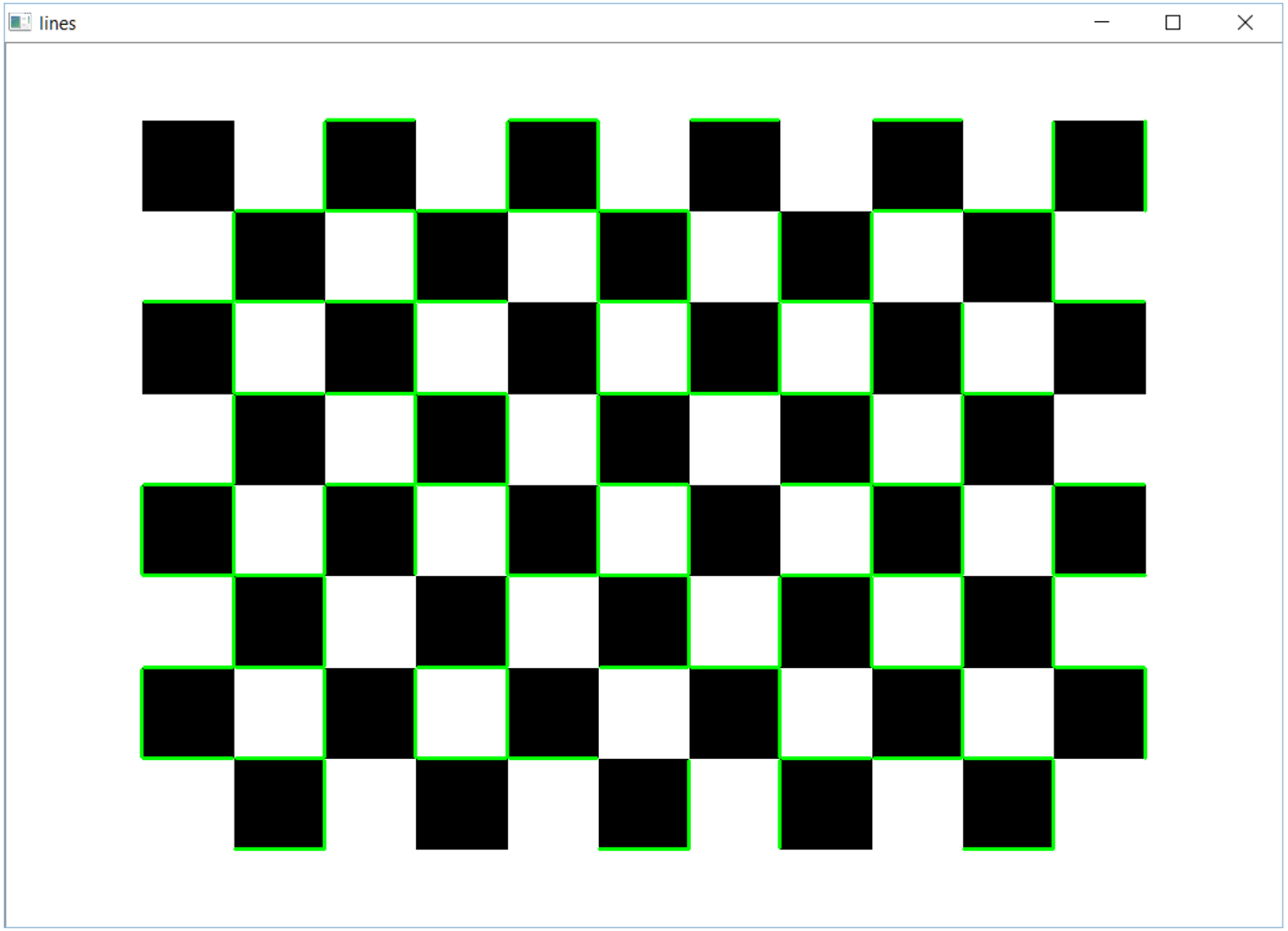
Детектирование блобов. Параметры



Детектирование линий

```
image = cv2.imread(image_name)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 50, 150)

minLineLength = 100
maxLineGap = 10
lines = cv2.HoughLinesP(edges, 1, np.pi / 180, 100, minLineLength, maxLineGap)
lines = [x[0] for x in lines]
for x1, y1, x2, y2 in lines:
    cv2.line(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
cv2.imshow('lines', image)
cv2.waitKey()
```

Детектирование линий



Поиск шахматной доски

```
chess_image = cv2.imread(“chess.png”, 0)
Image = cv2.imread(“panda.jpg)
board_size = (7, 10)
ret, corners =
cv2.findChessboardCorners(chess_image, board_size)
```

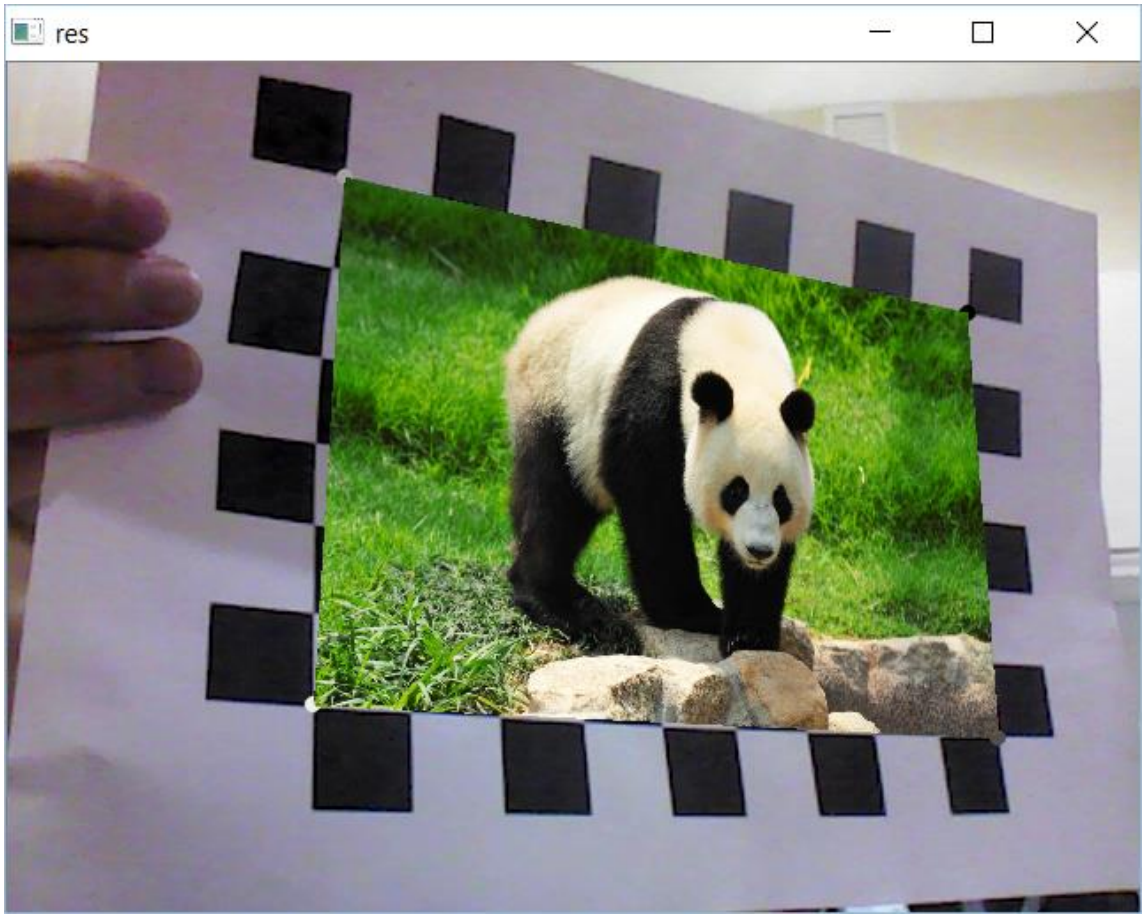
Поиск шахматной доски

```
chess_image = cv2.imread(“chess.png”, 0)
Image = cv2.imread(“panda.jpg)
board_size = (7, 10)
ret, corners =
cv2.findChessboardCorners(chess_image, board_size)
```

Поиск шахматной доски

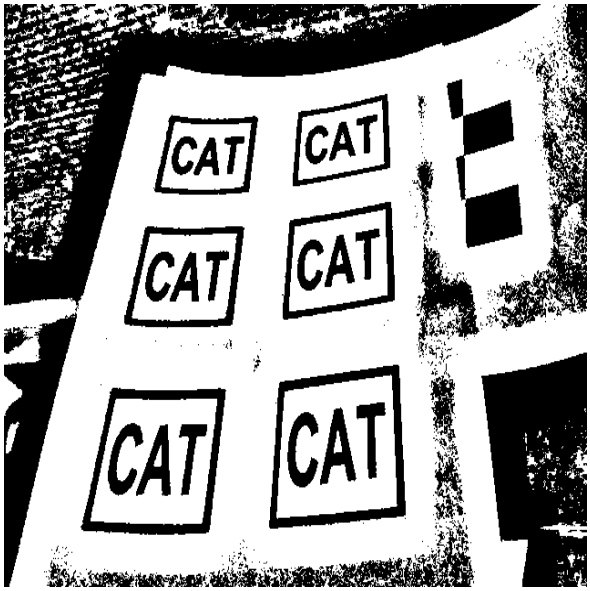
```
if ret:
    chess_corners = np.float32([tuple(corners[0][0]),
                                tuple(corners[board_size[0] - 1][0]),
                                tuple(corners[len(corners) - board_size[0]][0]),
                                tuple(corners[len(corners) - 1][0])])
    w,h = image.shape[:2]
    image_corners = np.float32([(0, 0), (0, w), (h, 0), (h, w )])
    transform = cv2.getPerspectiveTransform(image_corners,
                                            chess_corners)
    chess_image = cv2.warpPerspective(image, transform, (h,w),
                  chess_image, borderMode=cv2.BORDER_TRANSPARENT)
```

Поиск шахматной доски



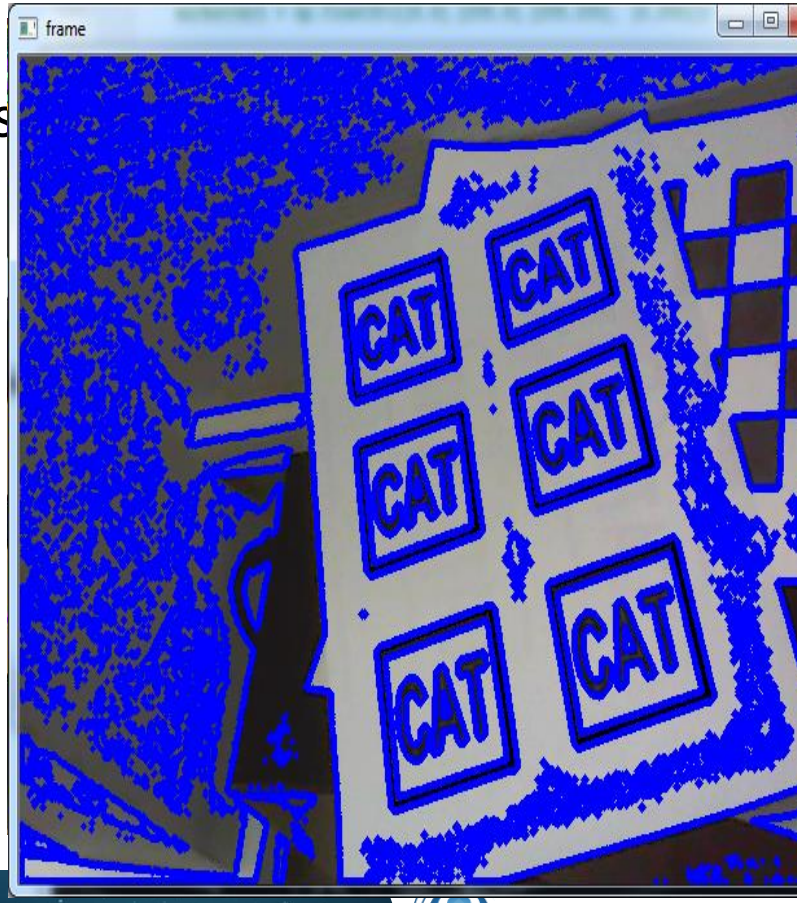
Чтение, бинаризация изображения

```
cap = cv2.VideoCapture(0)
while True:
    _,frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    binary = cv2.adaptiveThreshold(gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,

cv2.THRESH_BINARY, 71, 0)
    cv2.imshow('frame', frame)
    cv2.imshow('binary', binary)

    if 27 == cv2.waitKey(30):
        break
```



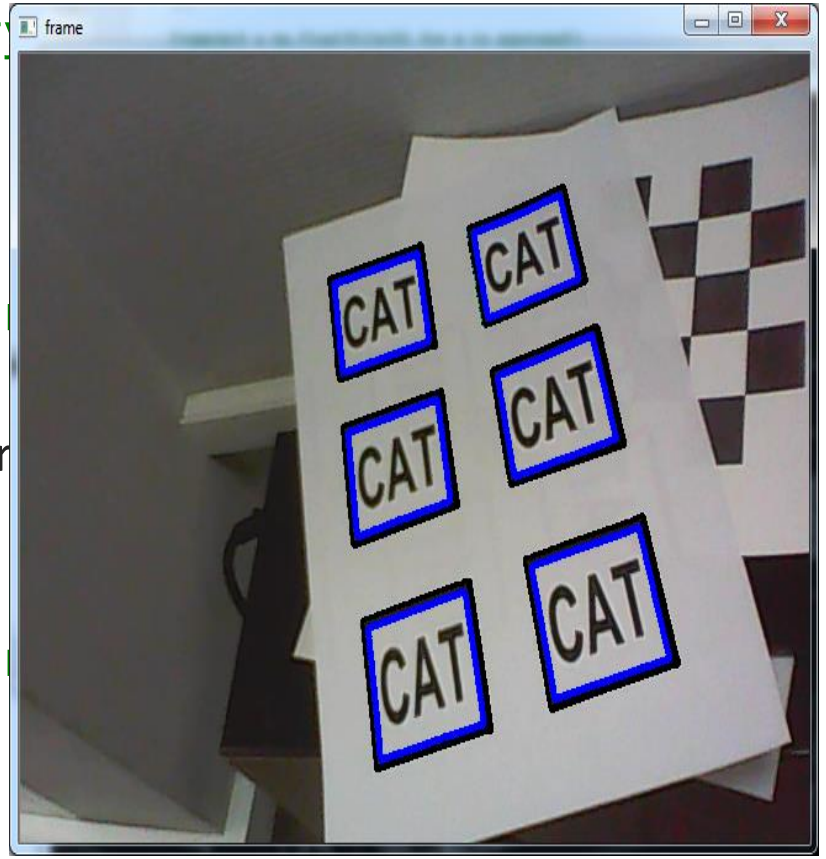
Поиск контуров

```
im2, contours, hierarhy = cv2.findContours(binary, cv2.RETR_TREE,
                                           cv2.CHAIN_APPROX_SIMPLE)

for index, contour in enumerate(contours
    cv2.drawContours(frame, [contour], 0,
```



Выделение рамок

```
# Отбрасываем самые маленькие конту
area = cv2.contourArea(contour)
if area < 2500:
    continue
# отбрасываем контуры не 4 угольни
contourLen = cv2.arcLength(contour,True)
approxed = cv2.approxPolyDP(contour,contour
if len(approxed) != 4:
    continue
# отбрасываем контуры без родителе
parent = hierarhy[0][index][3]
if parent == -1:
    continue
# рисуем аппроксимированный контур
cv2.drawContours(frame, [approxed], 0, (255,0,0),3)
```



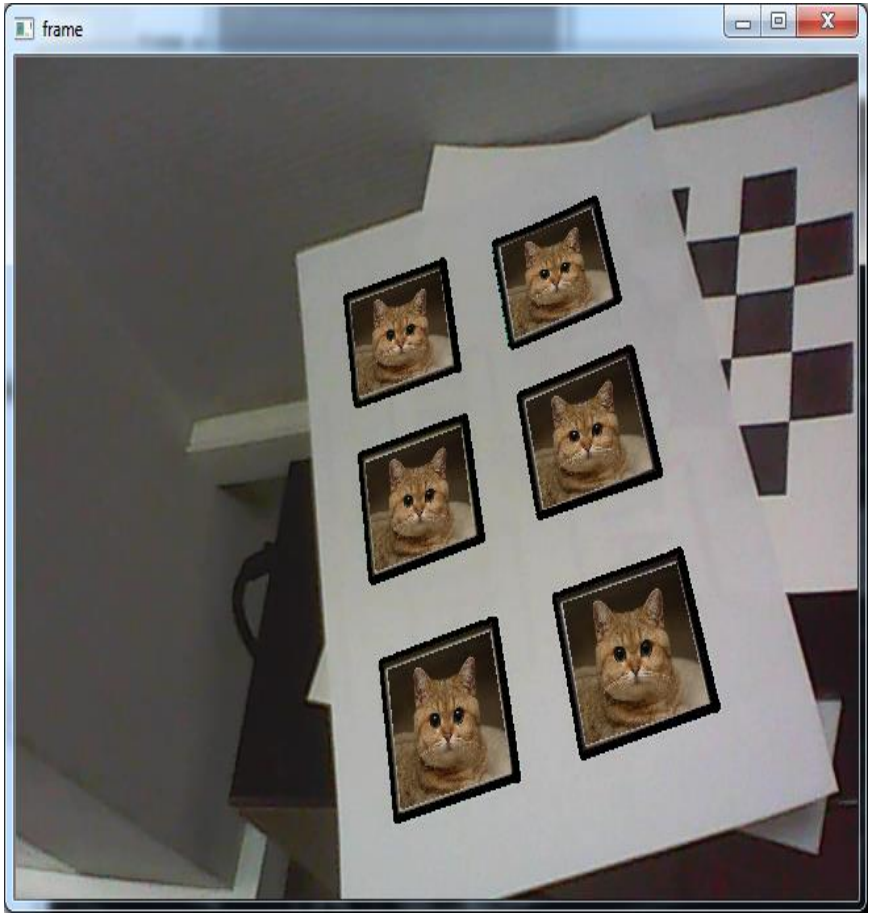
Получение изображения маркера

```
# Координаты углов маркера на картинке
framerect = np.float32([x[0] for x in approxed])

# Целевые углов трансформированного изображ
markerrect = np.float32([[0,0],[255,0],[255,255], [0,255]])

# Трансформация изображения
transform = cv2.getPerspectiveTransform(framerect, mark
marker = cv2.warpPerspective(gray,transform, (256,256))
cv2.imshow('marker',marker)
```

Выводим выбранное изображение

```
cat = cv2.imread('c:\\images\\cat.jpg')
…

// Координаты углов выводимого изображения
h, w = cat.shape[:2]
catrect = np.float32([[0,0],[w,0],[w,h], [0,h]])

// Трансформация
transform = cv2.getPerspectiveTransform(catrect, framerect)
frame = cv2.warpPerspective(cat, transform, frame.shape[:2][::-1],
frame, borderMode=cv2.BORDER_TRANSPARENT)
```

Результат работы



Спасибо за внимание

Александр Катаев

- alexander.kataev@singularis-lab.com

Алексей Алексеев

- aleksey.alekseev@singularis-lab.com

https://www.singularis-lab.com/

https://www.linkedin.com/company/singularis-lab-llc

http://habrahabr.ru/company/singularis

http://vk.com/singularis_lab