

Семинар 2. Детекция мяча с помощью OpenCV

Егор Давыденко, к.т.н

Moscow Institute of Physics and Technology

20 октября 2020 г.

Overview

- 1 Сущность мяча
- 2 MatchTemplate
- 3 HoughCircles
- 4 Canny -> FindContours
- 5 MSER -> ConvexHull -> Area ratio check
- 6 AdaptiveThreshold -> Skeletonize -> Ransac
- 7 AdaptiveThresholdGaussian -> FindContours -> FitEllipse
- 8 GreenFilter -> DistanceTransform -> ConnectedComponents
- 9 Green filter -> SimpleBlobDetector

Что такое мяч?



- Круглый с любого ракурса
- Белый с черным
- Закрывает собой поле
- Белая часть светлее поля
- Белая часть формирует почти полную окружность
- Достаточно четкая наружная граница
- Размер в каждой точке может быть заранее известен из модели геометрии робота

Сравнение с шаблоном - идея метода



- Мы знаем как выглядит мяч
- Вид мяча слабо меняется от ракурса
- Это всегда светлый круг на зеленом фоне
- Попиксельно сравниваем части изображения с образцом (шаблоном)
- При высокой степени совпадения - мяч найден

matchTemplate обзор



Template

Matching Result



Detected Point



https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html

matchTemplate методы сравнения

TM_SQDIFF

Python: cv.TM_SQDIFF

$$R(x, y) = \sum_{x',y'} (T(x', y') - I(x + x', y + y'))^2$$

TM_SQDIFF_NORMED

Python: cv.TM_SQDIFF_NORMED

$$R(x, y) = \frac{\sum_{x',y'} (T(x', y') - I(x + x', y + y'))^2}{\sqrt{\sum_{x',y'} T(x', y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2}}$$

TM_CCORR

Python: cv.TM_CCORR

$$R(x, y) = \sum_{x',y'} (T(x', y') \cdot I(x + x', y + y'))$$

TM_CCORR_NORMED

Python: cv.TM_CCORR_NORMED

$$R(x, y) = \frac{\sum_{x',y'} (T(x', y') \cdot I(x + x', y + y'))}{\sqrt{\sum_{x',y'} T(x', y')^2 \cdot \sum_{x',y'} I(x + x', y + y')^2}}$$

TM_CCOEFF

Python: cv.TM_CCOEFF

$$R(x, y) = \sum_{x',y'} (T'(x', y') \cdot I'(x + x', y + y'))$$

where

$$T'(x', y') = T(x', y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'', y'')$$

$$I'(x + x', y + y') = I(x + x', y + y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x + x'', y + y'')$$

TM_CCOEFF_NORMED

Python: cv.TM_CCOEFF_NORMED

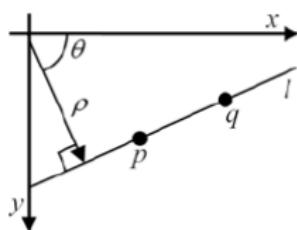
$$R(x, y) = \frac{\sum_{x',y'} (T'(x', y') \cdot I'(x + x', y + y'))}{\sqrt{\sum_{x',y'} T'(x', y')^2 \cdot \sum_{x',y'} I'(x + x', y + y')^2}}$$

Преобразование Хафа - идея метода

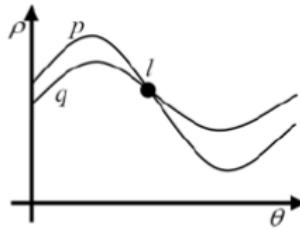


- Мяч круглый
- Имеет четкие края, которые формируют окружность
- Ищем все окружности на изображении, проверяем их размер
- При совпадении размера окружности с искомой - мяч найден

HoughLine метод



(a)



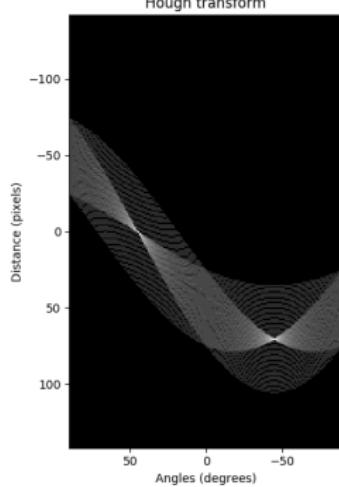
(b)

$$r = x * \cos(\theta) + y * \sin(\theta)$$

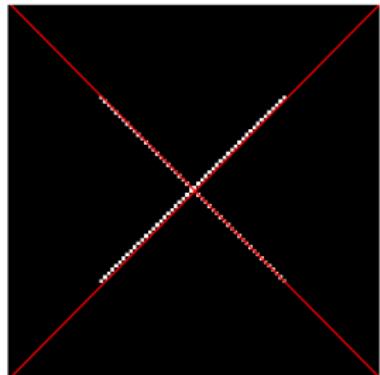
Input image



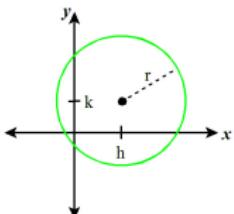
Hough transform



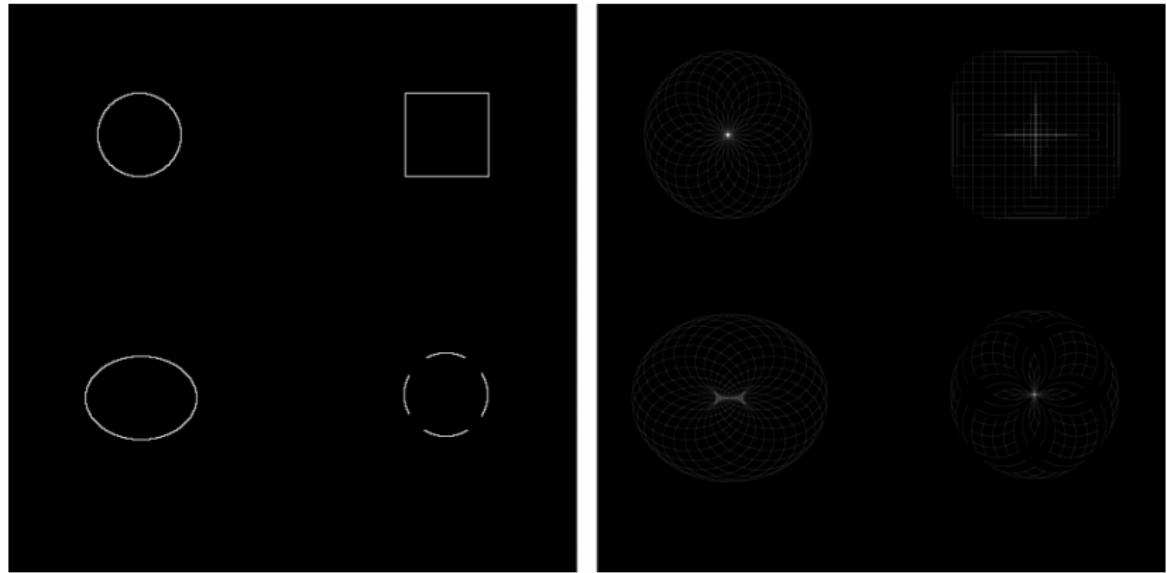
Detected lines



HoughCircle



$$(x - h)^2 + (y - k)^2 = r^2$$



Преобразование Кэнни + трассировка контуров - идея метода



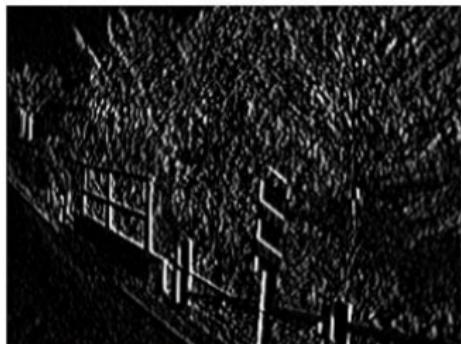
- Мяч имеет четкий круглый контур
- Форма контура слабо зависит от ракурса
- Выделяем края (контуры) на изображении
- Ищем длинные цепочки пикселей контуров, оцениваем насколько они круглые
- При высокой степени "округлости" контура - мяч найден

Sobel Edge Detector

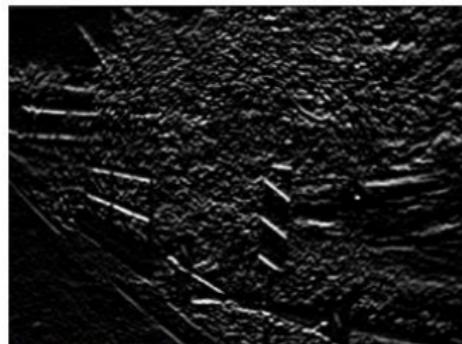


$$\text{Edge_Gradient } (G) = \sqrt{G_x^2 + G_y^2}$$

$$\text{Angle } (\theta) = \tan^{-1} \left(\frac{G_y}{G_x} \right)$$

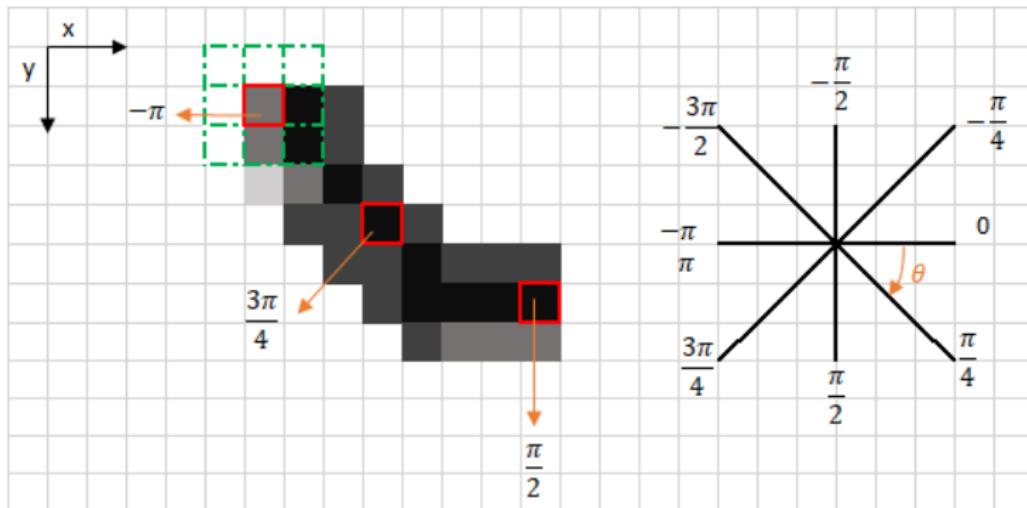


$$\begin{array}{|c|c|c|} \hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \quad G_x$$



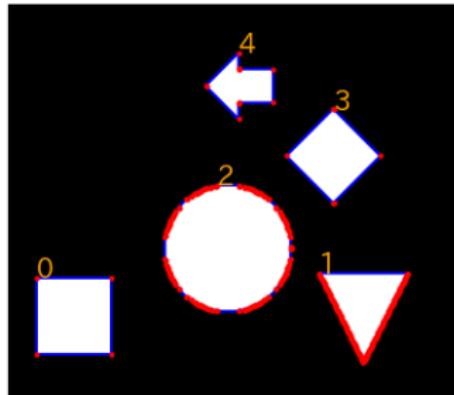
$$\begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \quad G_y$$

Canny Edge Detector



<https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123>

FindContours + Circularity check



$$\text{Circularity} = \frac{4 * \pi * \text{Area}}{(\text{Perimeter} * \text{Perimeter})}$$

Circle: 1.0

Rectangle: 0.698

Triangle: 0.605

Square: 0.785

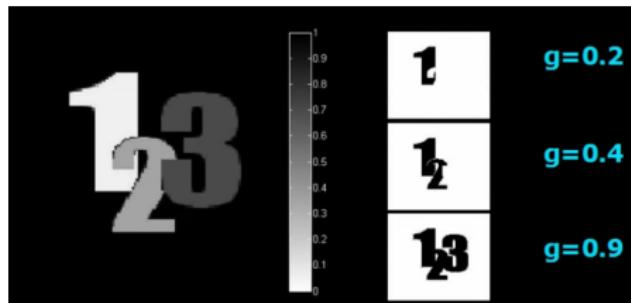
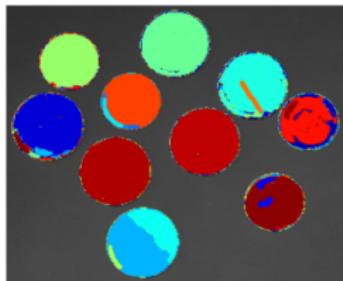
Hexagon: 0.907

Нахождение " пятен" на изображении - идея метода



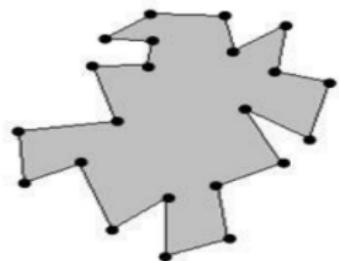
- Мяч яркий и преимущественно белый
- Он выглядит как белое пятно на более темном фоне
- Ищем все " пятна" (непрерывные светлые области) на изображении
- Строим выпуклый контур вокруг краев найденных пятен
- Если площадь этого контура близка к площади круга - мяч найден

MSER: Maximally Stable Extremal Region

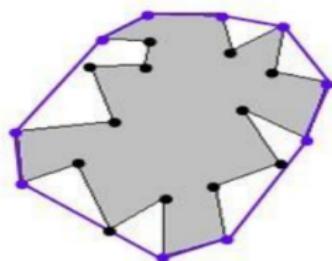


<https://www.youtube.com/watch?v=b2pLUxB9Vaw>

Convex hull



Region from Image



Convex hull of Region



Convex Area of Region

$$areaRatio = \frac{areaFromConvexHull}{areaFromCircle}$$

0 - тонкая линия, 1 - круг

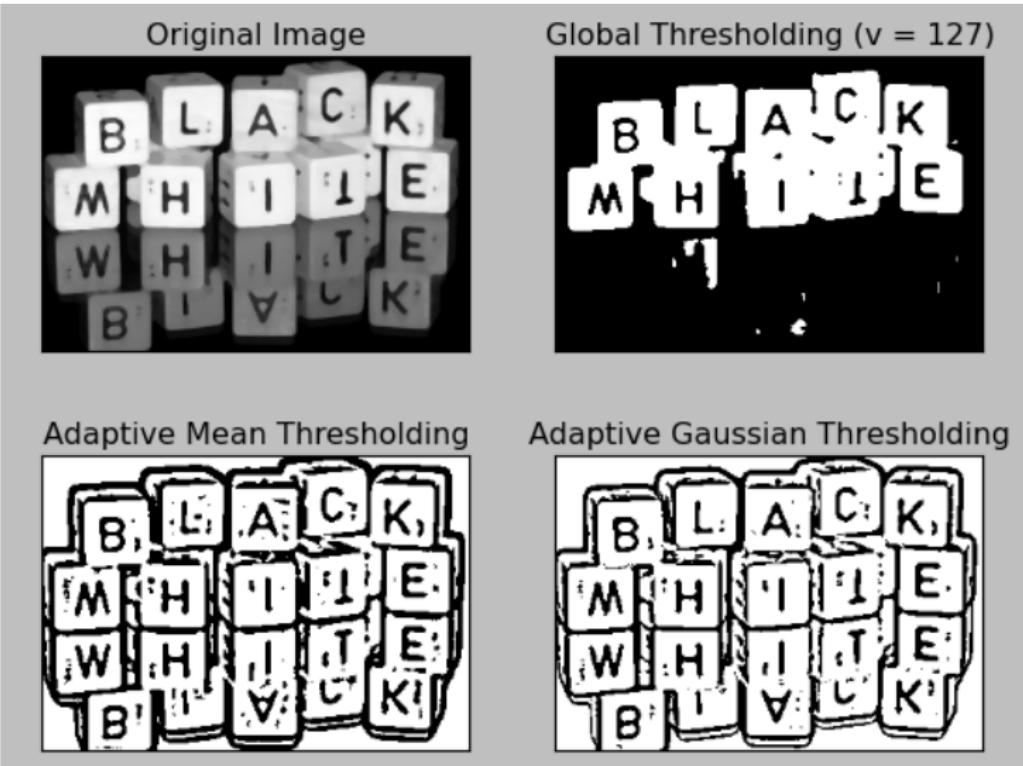
https://shodhganga.inflibnet.ac.in/bitstream/10603/36188/16/16_chapter 5.pdf

Адаптивная пороговая обработка + скелетонизация + RANSAC - идея метода



- Мяч круглый, имеет четкие края
- Края мяча точно светлые, а вокруг либо темная тень от мяча, либо темное поле, либо темные пятна на самом мяче
- Ищем на изображении точки которые заметно светлее своего окружения, скорее всего это будут точки белого края мяча
- Ищем группы этих точек, условные середины которых формируют окружность
- Если окружность почти непрерывна и подходящего радиуса - мяч найден

AdaptiveThreshold для выделения краев



https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Global_Thresholding_Adaptive_Thresholding.php

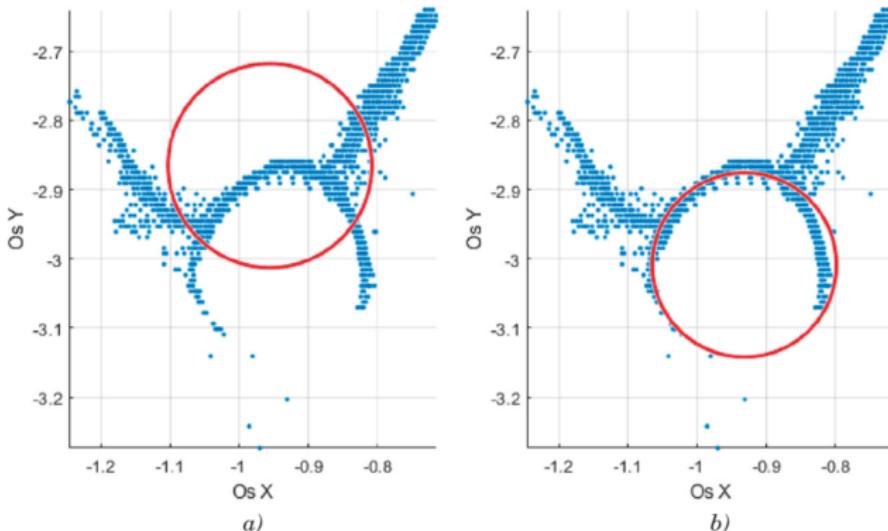
Skeletonize



<https://www.youtube.com/watch?v=iOtodmhfMfU>

RANSAC: RANdom SAmple Consensus

- Случайный выбор 3 точек
- Построение гипотезы (окружность по 3 точкам)
- Проверка гипотезы, сохранение лучшей гипотезы



https://www.researchgate.net/figure/Interpolated-circle-obtained-a-by-the-least-squares-method-b-the-RANSAC-algorithm_fig4_329104411

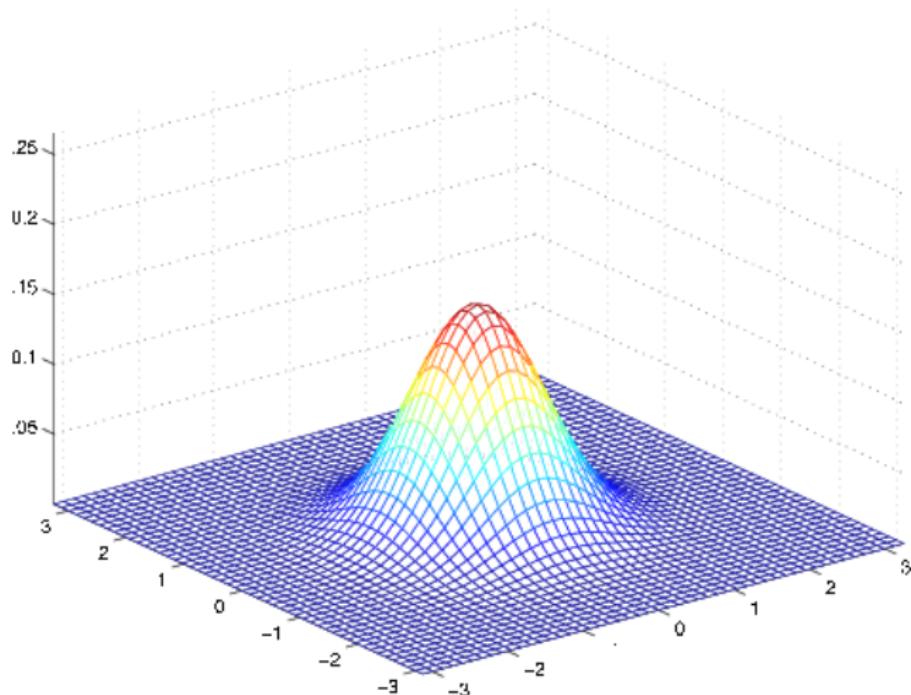
Адаптивная пороговая обработка по соразмерному объекту гауссовому ядру + аппроксимация эллипсом - идея метода



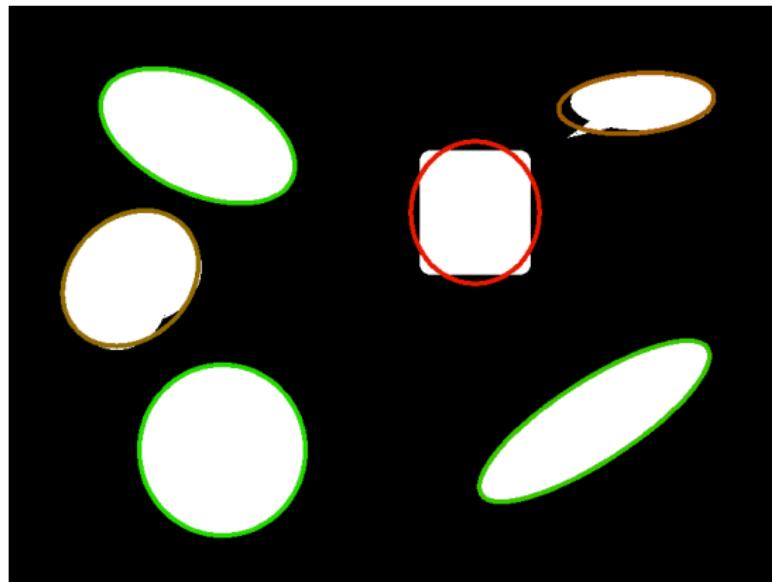
- Мяч яркий и преимущественно белый
- Он выглядит как белое пятно на более темном фоне
- Мы примерно знаем размер этого пятна
- Ищем все области (заданного размера) на изображении которые заметно светлее своего окружения
- Аппроксимируем контур найденной области эллипсом через МНК
- Если эллипс близок к кругу и нужного размера - мяч найден

AdaptiveThreshold для выделение объекта целиком

```
cv2.adaptiveThreshold(img, 128, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,  
cv2.THRESH_BINARY_INV, 101, -70)
```



FitEllipse



<https://stackoverflow.com/questions/35121045/find-cost-of-ellipse-in-opencv>

Фильтр зеленого + Карта расстояний + Связные компоненты - идея метода

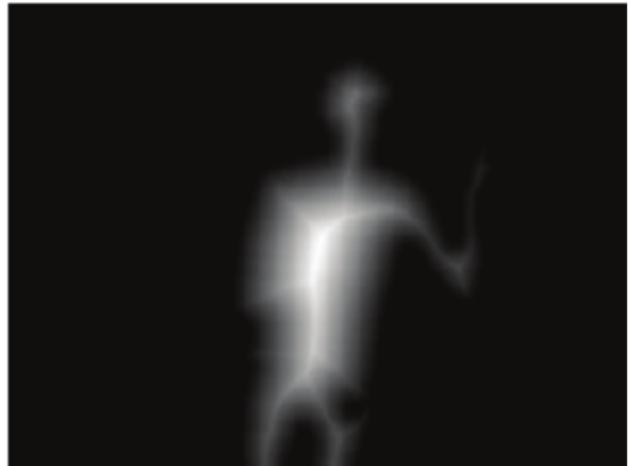


- Мяч круглый
- Мяч закрывает собой зеленое поле
- Значит мяч это круглое черное пятно на зеленом фоне
- В центре этого пятна есть центральная точка, равноудаленная от всех ближайших зеленых пикселей
- Найдем все точки удаленные от зеленых пикселей на расстояние не менее радиуса мяча
- Объединим все такие смежные точки в кластеры
- Если площадь кластера мала (он похож на точку) - это центр искомого мяча

GreenFilter (фильтр зеленого)

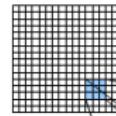
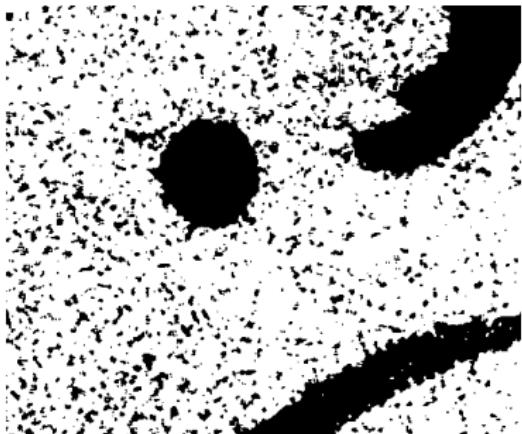


Distance Transform



https://www.researchgate.net/figure/Image-supports-a-silhouette-b-euclidean-distance-transform_fig2_26851982

Median Filter



101	69	0
56	255	87
123	96	157

0	56	69	87	96	101	123	157	255
---	----	----	----	----	-----	-----	-----	-----

N.B. Each template takes the values it sorts from the original image

Connected components



https://scipy-lectures.org/advanced/image_processing/auto_examples/plot_synthetic_data.html

Фильтр зеленого + SimpleBlobDetector - идея метода



- Мяч круглый
- Мяч закрывает собой зеленое поле
- Значит мяч это круглое черное пятно на зеленом фоне
- В OpenCV уже есть функция нахождения "пятен"!
- Она даже умеет проверять насколько эти пятна похожи на круг!!
- Используем эту функцию!!!

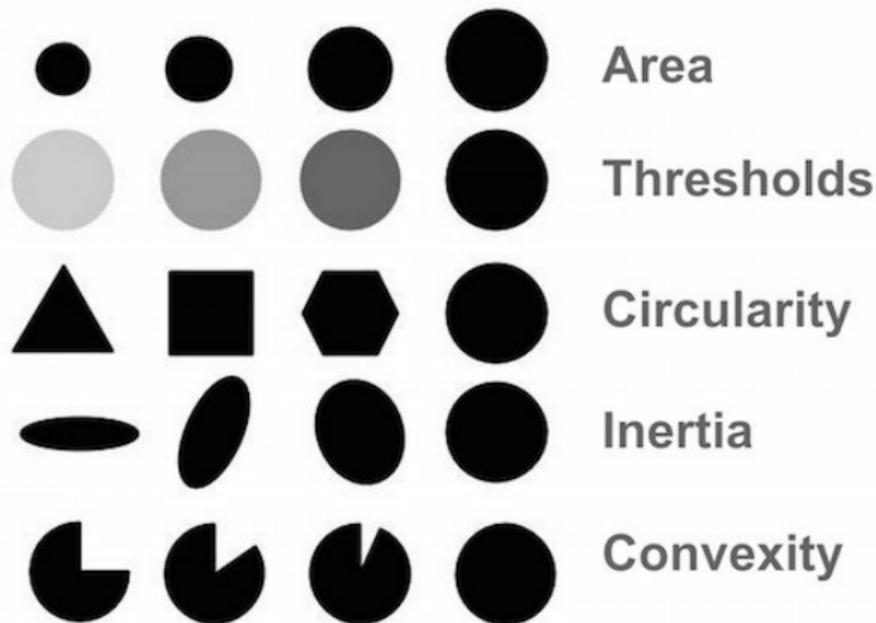
SimpleBlobDetector - введение

Готовый детектор объединяющий часть рассмотренного функционала:

```
detector = cv2.SimpleBlobDetector()  
keypoints = detector.detect(image)
```

- Многоуровневая пороговая обработка : получение черно-белых изображений из исходного путем сравнения с порогом от minThreshold до maxThreshold с шагом thresholdStep
- Группировка : на каждом черно-белом изображении находятся связные области
- Объединение : центры областей на различных черно-белых изображениях объединяются если они ближе чем minDistBetweenBlobs
- Проверка : центры и радиусы найденных областей проверяются на соответствие критериям

SimpleBlobDetector - критерии



<https://www.learnopencv.com/blob-detection-using-opencv-python-c/>

А что если...?



- Мяч не на фоне зеленой части поля, а на фоне белой разметки?
- Мяч не на фоне зеленой части поля, а на фоне белой стойки ворот?
- Мяч с самого края поля, позади него - ноги болельщиков?
- Мяч размыт от движения?
- Мяч подлетел и его размер не совпадает с ожидаемым?
- Мяч частично закрыт противником?

Q & A



Команда:

- starkit.ru
- github.com/StarkitRobots
- vk.com/starkitmipt
- facebook.com/StarkitTeam

Контакты:

- Егор Давыденко
- egordv@gmail.com