МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ГАЗА

ИМЕНИ И. М. ГУБКИНА

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА ИНФОРМАТИКИ

Расчетно-графическая работа по дисциплине

“Основы алгоритмизации и программирования”

на тему: “Распознавание показаний аналоговых приборов

методами компьютерного зрения”

Выполнил:

Ефанов Иван Николаевич

Студент группы АИ-19-02

Проверил:

# Оглавление

Оглавление

[Оглавление 2](#_Toc38630724)

[Введение 3](#_Toc38630725)

[Основная часть 3](#_Toc38630726)

[Постановка задачи 3](#_Toc38630727)

[Технические особенности работы 4](#_Toc38630728)

[Методы решения, Описание алгоритмов 4](#_Toc38630729)

[Метод поиска по цвету 4](#_Toc38630730)

[Метод моментов 5](#_Toc38630731)

[Результаты 5](#_Toc38630732)

[Заключение 5](#_Toc38630733)

[Список использованных источников 5](#_Toc38630734)

[Приложение 1 6](#_Toc38630735)

[Приложение 2 10](#_Toc38630736)

[Метод поиска по цвету 10](#_Toc38630737)

[Метод моментов 12](#_Toc38630738)

[Прочие функции 17](#_Toc38630739)

[GUI 17](#_Toc38630740)

# Введение

Распознавание показаний приборов методами компьютерного зрения – актуальная проблема нефтегазовой отрасли, поскольку позволяет снимать показания с приборов в труднодоступных районах, например, фотографируя их с дронов. Мной было разработано несколько алгоритмов, решающих эту задачу, в качестве материала для тестирования использовались фотографии из сети интернет.

# Основная часть

## Постановка задачи

На данный момент, в открытых источниках решения этой проблемы не существует, однако, есть алгоритмы, позволяющие распознавать фрагменты изображений по некоторым признакам. Чтобы распознать показания прибора, необходимо найти угол наклона его стрелочки, затем, в зависимости от особенностей модели прибора можно создать формулу, описывающую зависимость его показаний от угла наклона стрелки. Таким образом, проблема сводится к поиску этого угла. Я не занимался созданием формул, поскольку, они полностью зависят от модели прибора. Мной было разработано несколько алгоритмов решения задачи, к их описанию приступим в следующих частях. 

## Технические особенности работы

Мной был использован язык программирования Python версии 3.6, виртуальное окружение pip и Pycharm IDE. Использовались библиотеки питон opencv2, numpy, math, os, tkinter последих версий.

## Методы решения, Описание алгоритмов

Мной было разработано 2 метода решения задачи, один показал крайне низкую точность срабатывания, тем не менее, его я тоже приведу в работе.

### Метод поиска по цвету

Данный метод заключается в поиске на изображении фигуры (стрелочки) определенного цвета, отличного от цвета корпуса. Т.к на большинстве приборов они имеют красный цвет, я использовал его. Алгоритм:

1. Импорт изображения и преобразование его в цветовую модель HSV (opencv cvtColor)
2. Применение цветового фильтра по нужному цвету (opencv threshold)
3. Поиск контуров на полученном изображении (opencv findContours)
4. Поиск прямоугольных контуров (opencv minArearect)
5. Очистка помех, стрелочка – самый большой
6. Поиск угла наклона большей стороны прямоугольника
7. Если прямоугольник расположен выше центра фотографии, то угол (0, 180), если ниже, (180, 360). Т.к Принадлежность к верхней или нижней полуплоскости определяется дополнительно, если нижняя, то к углу прибавляется 180 гр.
8. Вывод угла

Минусом метода является необходимость наличия стрелочки цвета отличающегося от цвета корпуса.

В работе представлен функцией sq\_method

### Метод моментов

Этот метод имеет очень низкую точность и в реальном проекте использоваться сможет только после доработки. Я обзорно опишу его алгоритм.

1. Импорт изображения, его преобразование в черно-белое, размытие методом Гаусса и удаление помех. Сложно подобрать нужные коэффициенты, такие, чтобы на изображении остался виден корпус и стрелочка прибора и были удалены все лишние детали. (opencv GaussianBlur, threshold) в работе – функции get\_ready и get\_ready2
2. Изображение обрезается, так, чтобы остался только центр прибора с стрелкой, а корпус был бы не виден.
3. Вычисляются цветовые моменты (opencv moments)
4. Из них можно найти тангенс угла наклона стрелочки и, соответственно, сам угол.

На практике, метод нахождения угла из цветовых моментов имеет низкую точность, недостаточную для работы с приборами, и коэффициенты гаусса подбирать – задача сложная, приходится выполнять её вручную, сто нивелирует смысл программы.

## Результаты

Метод поиска по цвету уверенно находит результат, картинки с примерами работы см. в Приложении. Код в Приложении 2

# Заключение

В рамках РГР было проведено исследование методов компьютерного зрения. И изучил библиотеку Opencv, создание GUI c Tkinter. Разработан метод, спосоный решать поставленные задачи, хотя и с некоторыми ограничениями.

# Список использованных источников

Oфициальный сайт Opencv

<https://opencv.org/>

Пример похожей проблемы

<https://tproger.ru/translations/finding-books-python-opencv/>

<https://ru.stackoverflow.com/questions/922401/Проблема-установки-opencv-python-на-3-питон>

Поиск фигур на изобржении

<https://линуксблог.рф/opencv-mashinnoe-zrenie-na-python-pryamougolnye-obekty-chast-3/>

<https://robotclass.ru/tutorials/opencv-python-find-contours/>

Tkinter

<https://habr.com/ru/post/133337/>

<https://habr.com/ru/company/edison/blog/480884/>

Статьи про распознавание изображений

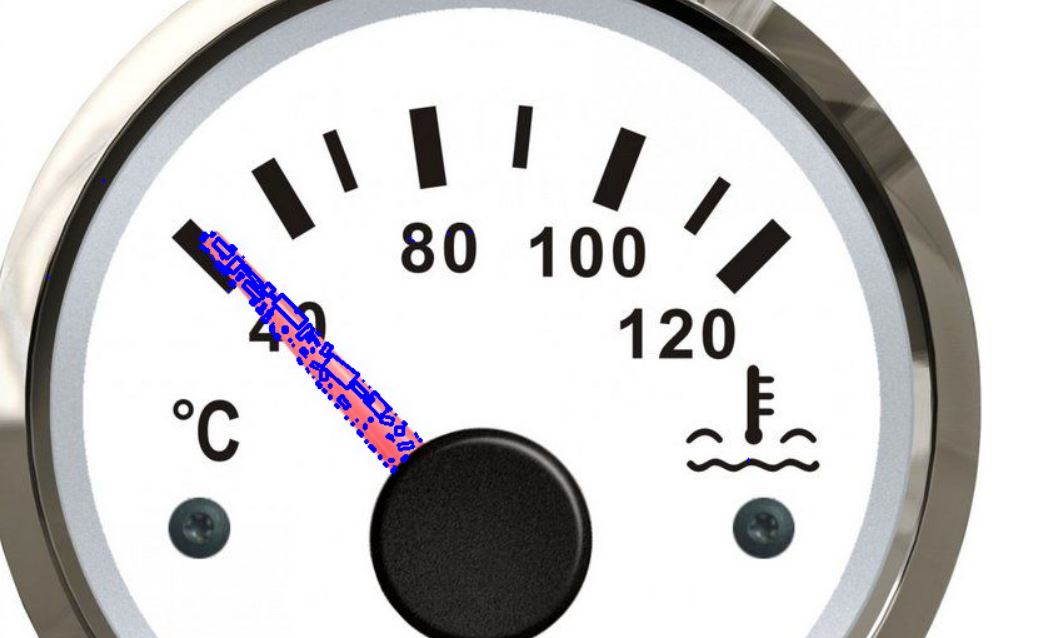
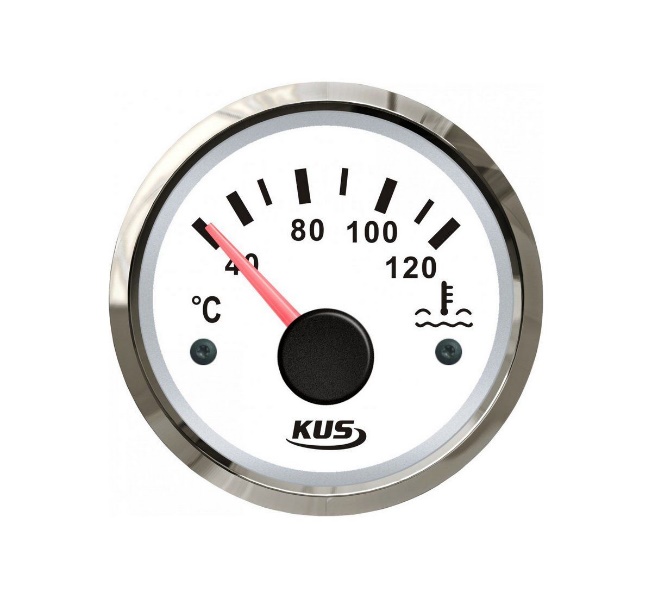
<https://habr.com/ru/post/249661/>

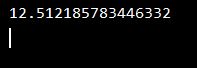
<https://habr.com/ru/post/208090/>

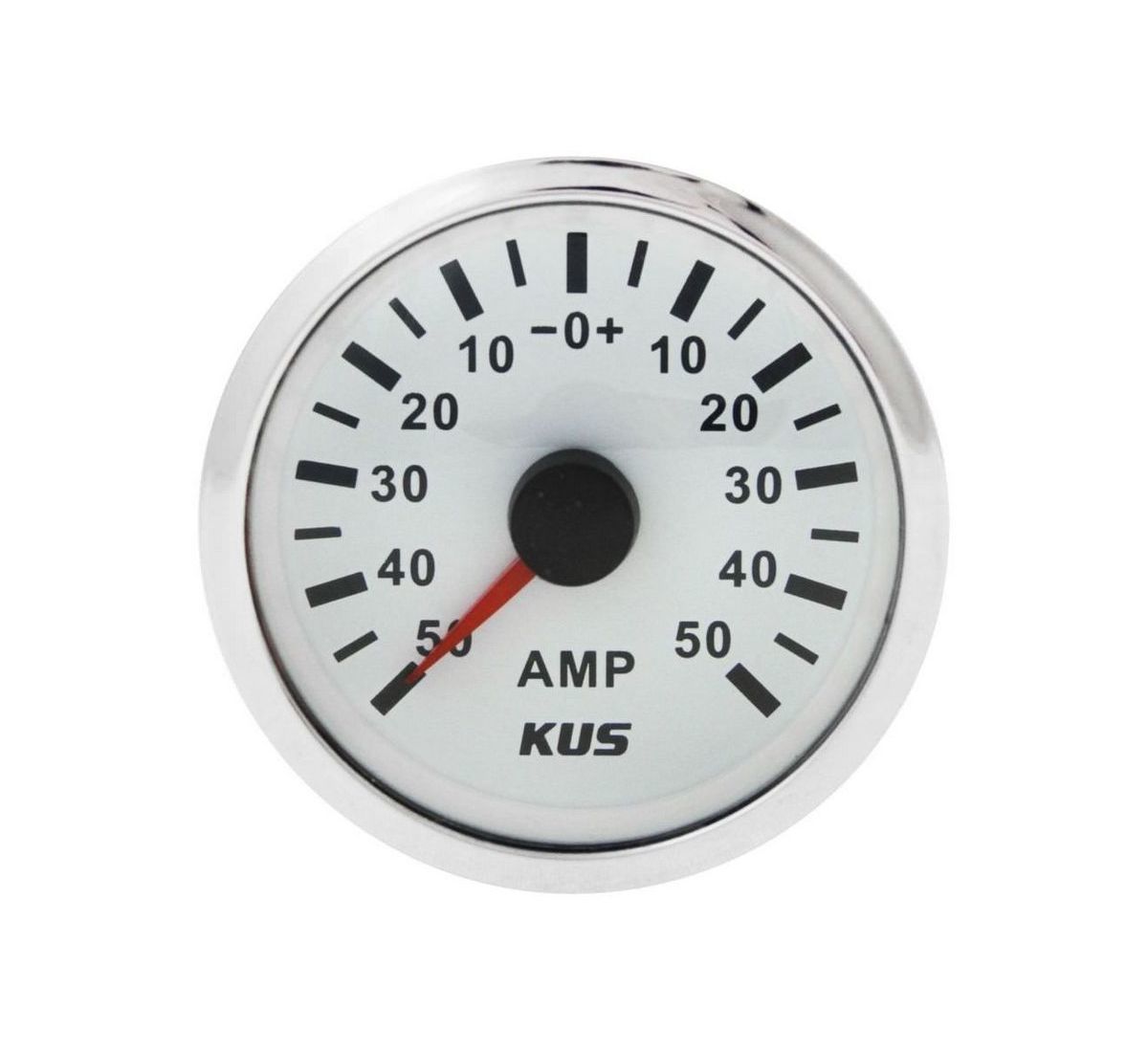
<https://medium.com/nuances-of-programming/распознавание-лиц-с-помощью-opencv-2e01375eb2a0>

# Приложение 1

Картинки сгруппированы по 3: оригинал, контуры, ответ







# Приложение 2

## Метод поиска по цвету

def sq\_method(name\_of\_picture):  
 img = cv.imread(name\_of\_picture)  
  
 hsv\_min = np.array((**0, 50, 50**)**,** np.uint8)  
 hsv\_max = np.array((**10, 255, 255**)**,** np.uint8)  
  
 # hold = height \* width // 5000  
 hold = **0** hsv = cv.cvtColor(img**,** cv.COLOR\_BGR2HSV) # меняем цветовую модель с BGR на HSV  
 thresh = cv.inRange(hsv**,** hsv\_min**,** hsv\_max) # применяем цветовой фильтр  
 contours0**,** hierarchy = cv.findContours(thresh.copy()**,** cv.RETR\_TREE**,** cv.CHAIN\_APPROX\_SIMPLE)  
 angles = []  
 areas = []  
  
 # перебираем все найденные контуры в цикле  
 for cnt in contours0:  
 rect = cv.minAreaRect(cnt) # пытаемся вписать прямоугольник  
 box = cv.boxPoints(rect) # поиск четырех вершин прямоугольника  
 box = np.int0(box) # округление координат  
 sx = sy = **0** for i in range(**4**):  
 sx = sx + box[i][**0**]  
 sy = sx + box[i][**1**]  
  
 center\_of\_box = (sx // **4,** sy // **4**)  
  
 is\_invert = (box[**1**][**1**] > center\_of\_box[**1**])  
  
 area = int(rect[**1**][**0**] \* rect[**1**][**1**])  
  
 if area >= hold:  
 cv.drawContours(img**,** [box]**, 0,** (**255, 0, 0**)**, 2**) # рисуем прямоугольник  
 edge1 = np.int0((box[**1**][**0**] - box[**0**][**0**]**,** box[**1**][**1**] - box[**0**][**1**]))  
 edge2 = np.int0((box[**2**][**0**] - box[**1**][**0**]**,** box[**2**][**1**] - box[**1**][**1**]))  
  
 if cv.norm(edge2) > cv.norm(edge1):  
 usedEdge = edge2  
 else:  
 usedEdge = edge1  
  
 reference = (**1, 0**)  
 angle = **180.0** / math.pi \* math.acos((reference[**0**] \* usedEdge[**0**] + reference[**1**] \* usedEdge[**1**]) / (  
 cv.norm(reference) \* cv.norm(usedEdge))) # угол с горизонтом  
 if is\_invert:  
 angle = **180** + angle  
 areas.append(area)  
 angles.append(angle)  
  
 cv.imshow('contours'**,** img) # вывод обработанного кадра в окно  
  
 cv.waitKey()  
 cv.destroyAllWindows()  
  
 area\_of\_angle\_of\_biggest\_box = **0** for i in range(len(angles)):  
 if area\_of\_angle\_of\_biggest\_box < areas[i]:  
 area\_of\_angle\_of\_biggest\_box = areas[i]  
 angle\_of\_biggest\_box = angles[i]  
  
 return angle\_of\_biggest\_box

## Метод моментов

def main(name\_of\_picture**,** type\_is**,** zero\_angle**,** one\_angle**,** one\_angle\_cost):  
 P1 = PrMain(name\_of\_picture**,** cv.imread(name\_of\_picture**,** cv.IMREAD\_GRAYSCALE)**,** type\_is**,** zero\_angle**,** one\_angle**,** one\_angle\_cost)  
 img = P1.picture  
 show(img**,** P1.name)  
  
 try:  
 img\_p = get\_ready2(img)  
  
 show(img\_p**,** P1.name)  
 img\_p = resize(img\_p)  
  
 find\_tg(img\_p)  
 except:  
 atn = sq\_method(name\_of\_picture)  
 tan = ""  
  
 result = res\_from\_angle(name\_of\_picture**,** type\_is**,** zero\_angle**,** one\_angle**,** one\_angle\_cost**,** atn)  
 return "Тангенс"**,** tan**,** "Угол:"**,** atn**,** "(град)"**,** "Прибор показывает:"**,** result**,** "+/-"**,** P1.one\_angle\_cost / **2**def show(image**,** name): # показать картинку  
 cv.imshow(name**,** image)  
 cv.waitKey(**0**)  
  
  
def aver(x**,** y):  
 return int((x + y) // **2**) # точно работает корректно))  
  
  
def get\_ready(img**,** gauss\_cell\_size1**,** gauss\_cell\_size2**,** gauss\_mov**,** dual\_treshold): # подготовка картинки через гаусса  
 # и дуализацию, работает корректно, но нужно подбиирать все  
 # параметры вручную  
 img\_g = cv.GaussianBlur(img**,** (gauss\_cell\_size1**,** gauss\_cell\_size2)**,** gauss\_mov)  
 ret**,** image\_dw = cv.threshold(img\_g**,** dual\_treshold**, 255, 60**)  
 return image\_dw  
  
  
def get\_ready2(img):  
 img = cv.GaussianBlur(img**,** (**25, 25**)**, 0**)  
 image\_dw = cv.adaptiveThreshold(img**, 255,** cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C**,** cv.THRESH\_BINARY**, 101, 20**)  
  
 show(img**,** "g")  
  
 return image\_dw  
  
  
def find\_wd\_gate(ar):  
 for i in range(len(ar)):  
 if ar[i] == **0** and ar[i + **1**] == **255**:  
 r = i + **1** return r # возвращает первый переход чб в массиве, работает корректно  
  
  
def get\_wd\_gates(img\_p):  
 h = img\_p.shape[**0**] # находим переходы из черного в белое по 4 сторонам, работает корректно  
 w = img\_p.shape[**1**]  
 x = [**0, 0, 0, 0**]  
 y = [**0, 0, 0, 0**]  
 a = list(img\_p[**1**:h**,** w // **2**]) # вертикаль  
 y[**0**] = find\_wd\_gate(a)  
 a.reverse()  
 y[**2**] = h - find\_wd\_gate(a)  
 a = list(img\_p[h // **2, 1**:w]) # горизонталь  
 x[**1**] = find\_wd\_gate(a)  
 a.reverse()  
 x[**3**] = w - find\_wd\_gate(a)  
 x[**0**] = x[**2**] = w // **2** y[**1**] = y[**3**] = h // **2** return x**,** y  
  
  
def xy\_aver(x**,** y):  
 avx = [**0, 0, 0, 0**] # среднее из пар пикселей, работаеет корректно  
 avy = [**0, 0, 0, 0**]  
 for i in range(**4**):  
 avx[i] = aver(x[i - **1**]**,** x[i])  
 avy[i] = aver(y[i - **1**]**,** y[i])  
 x1 = min(avx)  
 x2 = max(avx)  
 y1 = min(avy)  
 y2 = max(avy)  
 return x1**,** y1**,** x2**,** y2  
  
  
def resize(img):  
 x**,** y = get\_wd\_gates(img) # 02 верт 13 гор  
 x1**,** y1**,** x2**,** y2 = xy\_aver(x**,** y)  
  
 img = img[y1:y2**,** x1:x2] # обрезка  
 show(img**,** "0")  
 return img  
  
  
def find\_tg(img): # обрезает картинку по найденным точкам и ищет тангенс стрелочки, возвращает арктангенс  
 # скорее всего содержит  
 # лажу  
 img = resize(img)  
 show(img**,** "")  
  
 M = cv.moments(img)  
 # print(M)  
 cX = int(M["m10"] / M["m00"])  
 cY = int(M["m01"] / M["m00"])  
  
 img1 = img[**0**:height // **2, 0**:width]  
 # M1 = cv.moments(img\_p[y1:aver(y1, y2), x1:x2]) # верхняя полукартинка  
 M1 = cv.moments(img1)  
 # print(M1)  
 cX1 = int(M1["m10"] / M1["m00"])  
 cY1 = int(M1["m01"] / M1["m00"])  
 show(img1**,** "1")  
  
 img2 = img[height // **2**:height**, 0**:width]  
 M2 = cv.moments(img2)  
 # M2 = cv.moments(img\_p[aver(y1, y2):y2, x1:x2]) # нижняя полукартинка  
 # print(M2)  
 cX2 = int(M2["m10"] / M2["m00"])  
 cY2 = int(M2["m01"] / M2["m00"])  
 show(img2**,** "2")  
 print("0p"**,** cX**,** cY**,** "1p"**,** cX1**,** cY1**,** "2p"**,** cX2**,** cY2)  
   
 try:  
 tan = (cY1 - cY) / (cX1 - cX) # проверить  
 atn = math.degrees(math.atan(tan)) # моментов полукартинок и всей картинки), скорее всего содержит лажу !!!  
 except:  
 atn = **90** tan = "n/e"  
 print("ex1")  
 return atn**,** tan

## Прочие функции

def findfiles():  
 res = []  
 subs = [".jfif"**,** ".jpg"**,** ".cam"]  
 for file in os.listdir(os.getcwd()):  
 for sub in subs:  
 if file.endswith(sub):  
 res.append(os.path.join(os.getcwd()**,** file))  
 return res

## GUI

from functions import \*  
from tkinter import \*  
from tkinter import filedialog  
import os  
  
root = Tk()  
root.geometry("550x300+300+150")  
root.resizable(width=True**,** height=True)  
  
  
def openfn():  
 filename = filedialog.askopenfilename(title='open')  
 return filename  
  
  
def open\_img():  
 # x = openfn()  
 # names = findfiles()  
 # img = Image.open(x)  
 name = message1.get()  
 print(name)  
 res = sq\_method(name)  
 print(res)  
  
  
message2 = StringVar()  
message1 = StringVar()  
message\_entry = Entry(textvariable=message1).pack()  
  
label = Label(root**,** textvariable=message2**,** relief=RAISED).pack  
btn = Button(root**,** text='Пуск'**,** command=open\_img).pack()  
  
root.mainloop()