МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания к выполнения практических работ по дисциплине «Анализ и обработка больших данных»

Направление подготовки	**09.04.02 Информационные системы и технологии**
Направленность (профиль) подготовки	**«Технологии работы с данными и знаниями, анализ информации»**

Пятигорск 2025

Содержание

ВВЕДЕНИЕ

OpenCV (Open Source Computer Vision Library) - это библиотека компьютерного зрения с открытым исходным кодом, которая содержит более 2500 алгоритмов для обработки изображений и видео, включая распознавание объектов и лиц, сегментацию изображений, обработку видео и создание трехмерных моделей. Она поддерживает различные языки программирования, включая C++, Python и Java, и может работать на разных операционных системах, таких как Windows, Linux и MacOS. OpenCV используется в различных областях, таких как медицина, автомобильная промышленность, робототехника, безопасность и многих других. Она имеет открытый исходный код и поддерживает различные языки программирования, что делает ее доступной для разработчиков со всего мира. OpenCV используется в различных областях, где требуется анализ изображений, таких как медицина, автомобильная промышленность, робототехника и безопасность. Благодаря своим многочисленным алгоритмам и функциям, OpenCV становится все более популярным инструментом для разработки приложений, связанных с компьютерным зрением.

ПРАКТИЧЕСКАЯ РАБОТА № 1 НАЧАЛО РАБОТЫ С БИБЛИОТЕКОЙ OPEN CV

Задача: научиться пользоваться соответствующими функциями открытой библиотеки OpenCV

a. Открытие изображения

Теперь, когда OpenCV установлен, первое, что нужно сделать, это импортировать модуль и открыть изображение, что довольно просто.

```
1  import cv2.cv as cv
2
3  image=cv.LoadImage('img/image.png', cv.CV_LOAD_IMAGE_COLOR)#Load the image
4  #Or just: image=cv.LoadImage('img/image.png')
5
6  cv.NamedWindow('a_window', cv.CV_WINDOW_AUTOSIZE) #Facultative
7  cv.ShowImage('a_window', image) #Show the image
8
9  cv.WaitKey(0) #wait for user input and quit
```

Дополнительная информация:

- image будет содержать объект iplimage, который чаще всего используется для обработки изображений. Порядок наследования - CvArr, который в основном представляет собой просто

массив, а затем CvMat, который может быть многомерным массивом с некоторой дополнительной информацией, такой как столбцы чисел или строк. И затем идет IplImage, который в основном содержит CvMat количество каналов (объяснено далее в учебнике) кодировка, используемая для изображения..

- аргумент *cv.CV_LOAD_IMAGE_COLOR* указывает тип изображения, которое мы хотим загрузить, но это необязательно и выполняется динамически. Существует также *CV_LOAD_IMAGE_GRAYSCALE* , если мы хотим загрузить цветное изображение как изображение в градациях серого, или *CV_LOAD_IMAGE_UNCHANGED* , которые ничего не меняют в изображении при его открытии.
- все окна, созданные и отображаемые на экране с помощью OpenCV, индексируются по их имени, поэтому NamedWindow создаст окно с заданным именем, а ShowImage поместит в окно данное изображение. Если окно не было объявлено ShowImage, создайте его таким образом, что NameWindow является факультативным.
- WaitKey дождитесь ввода пользователя, если вы забудете его поставить, программа завершится, и вы никогда не увидите изображение. 0 — это тайм-аут, а 0 означает бесконечное ожидание.

Примечание. Способ импорта cv может различаться в зависимости от установки и операционной системы. Например, это может быть просто импорт cv2 как cv. Только будьте осторожны, чтобы не импортировать старый синтаксис, унаследованный от swig.

Примечание. Все изображения, используемые для этого руководства, доступны здесь и были взяты из различных источников, таких как образцы OpenCV или те, которые приведены в Поваренной книге по программированию OpenCV 2.

б. HELLO WORLD !

Как и в каждом новом языке, первое, что мы хотим сделать, это классический Hello World! Код ниже, как вы можете видеть, почти такой же, как и выше.

```
import cv2.cv as cv

image=cv.LoadImage('img/lena.jpg', cv.CV_LOAD_IMAGE_COLOR) #Load the image

font = cv.InitFont(cv.CV_FONT_HERSHEY_SIMPLEX, 1, 1, 0, 3, 8) #Creates a font

y = image.height / 2 # y position of the text
x = image.width / 4 # x position of the text

cv.PutText(image,"Hello World !", (x,y),font, cv.RGB(255, 255, 255)) #Draw the t

cv.ShowImage('Hello World', image) #Show the image

cv.WaitKey(0)
```

Дополнительная информация:

- Положение оси Y устанавливается равным image.height/2, чтобы поместить его в центр изображения.
- Положение по оси X установлено равным image.width/4, чтобы текст начинался с первой четверти изображения.
- PutText просто помещает данную строку в заданную позицию, используя указанный шрифт и цвет. Обратите внимание, что RGB, установленный на 255, дает белый цвет.

в. Операции сохранения и изменения размера

В приведенном ниже примере показаны очень простые операции, такие как изменение размера и сохранение изображения. Действительно, все изменения, примененные к изображениям, не влияют на файл в самой файловой системе, а только в памяти. Вот почему мы можем захотеть сохранить изменения.

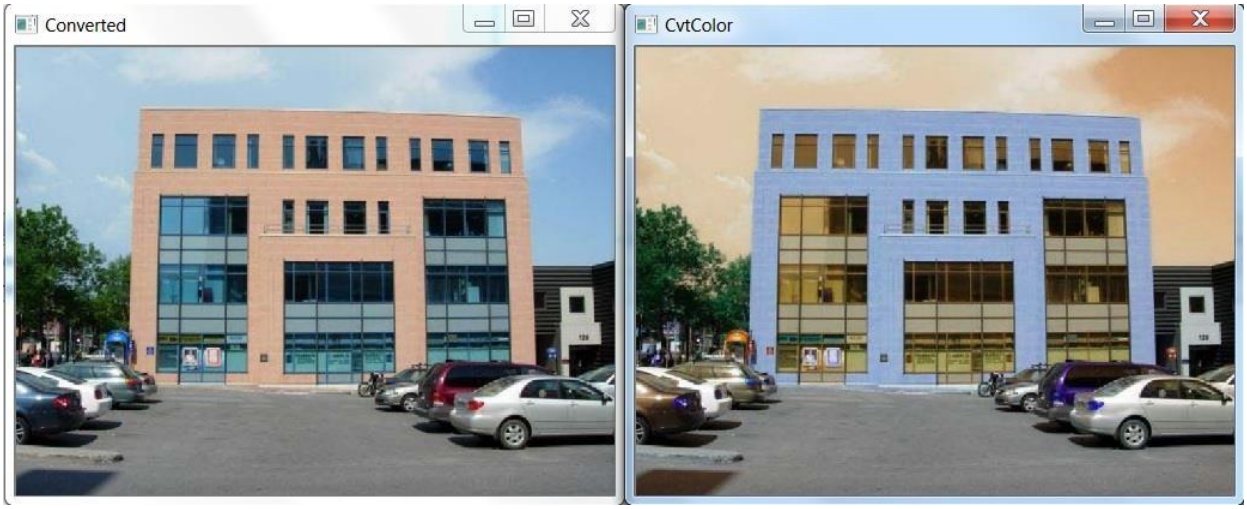
```
import cv2.cv as cv

im=cv.LoadImage('img/fruits.jpg',cv.CV_LOAD_IMAGE_COLOR)

res = cv.CreateImage(cv.GetSize(im), cv.CV_8UC2, 3) #cv.CV_32F, cv.IPL_DEPTH_16

cv.Convert(im, res) cv.ShowImage("Converted",res)
res2 = cv.CreateImage(cv.GetSize(im), cv.CV_8UC2, 3)
cv.CvtColor(im, res2, cv.CV_RGB2BGR) # HLS, HSV, YCrCb, ....

cv.ShowImage("CvtColor", res2)
cv.WaitKey(0)
```

Дополнительная информация: вместо cv.Resize вы также можете использовать *cv.PyrDown(im, thumb)* , которые изменяют размер, но применяют фильтр. Фильтр по умолчанию — GAUSSIAN 5x5.

д. Преобразование

Этот раздел очень важен, потому что в нем будут использоваться все уроки и примеры. Важно понять концепцию канала. Канал — это подизображение. Например, цветное изображение имеет 3 канала: красный, зеленый, синий. Таким образом, мы можем работать только с красным, работая на красном канале. Изображение в градациях серого имеет только один канал, который представляет для каждого пикселя величину яркости от 0 до 255. Другая концепция, которую необходимо понять, — это концепция используемой основы. В основном для пикселя значение для каждого цвета (RGB) включается от 0 до 255 и кодируется 8 битами без знака, но это не всегда так, и мы можем захотеть использовать другую основу. Фрагмент кода ниже показывает, как это сделать.

```
import cv2.cv as cv

im=cv.LoadImage('img/fruits.jpg',cv.CV_LOAD_IMAGE_COLOR)

res = cv.CreateImage(cv.GetSize(im), cv.CV_8UC2, 3) #cv.CV_32F, cv.IPL_DEPTH_16

cv.Convert(im, res) cv.ShowImage("Converted",res)
res2 = cv.CreateImage(cv.GetSize(im), cv.CV_8UC2, 3)
cv.CvtColor(im, res2, cv.CV_RGB2BGR) # HLS, HSV, YCrCb, ....

cv.ShowImage("CvtColor", res2)
cv.WaitKey(0)
```



Дополнительная информация:

- Окно преобразования (слева) показывает преобразованное изображение, но используется та же основа, поэтому мы не видим никакой разницы.
- Окно CvtColor (справа) показывает изображение, преобразованное из BGR в RGB, поэтому легко заметить, что на самом деле красный и синий каналы были переключены. Существует много различий между цветовым кодированием, оттенками серого, HSV, HLS... как мы увидим далее в руководстве.

Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?
3. Какие методы используются для обработки изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 2 ФИЛЬТРЫ И АРИФМЕТИЧЕСКИЕ ДЕЙСТВИЯ С ИЗОБРАЖЕНИЯМИ

Задача: научиться пользоваться соответствующими функциями открытой библиотеки OpenCV

a.Фильтры

Обработка изображений в основном связана с фильтрами, которые мы можем применять к изображениям. Эта часть быстро познакомит вас с более интересными фильтрами и с тем, как их использовать.

В приведенном ниже примере показано использование различных фильтров, и результат показал:

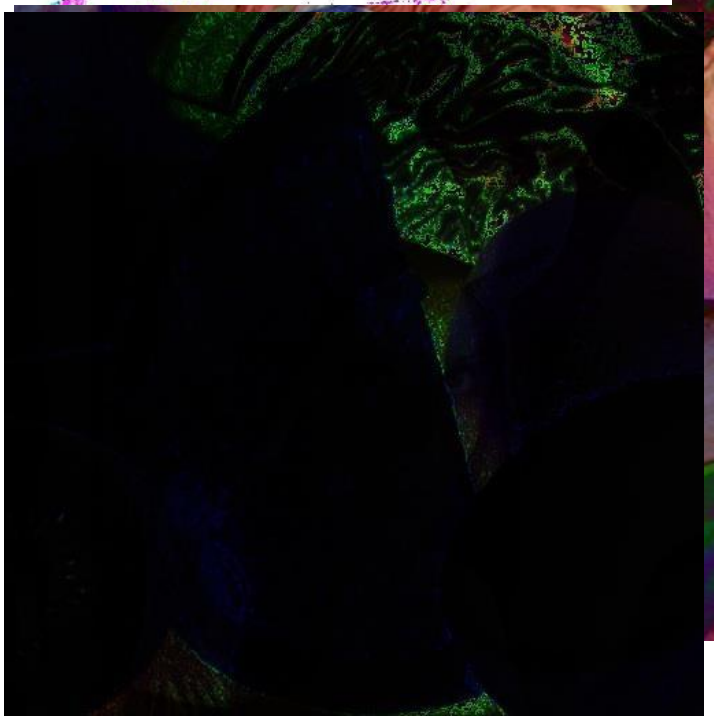
```
import cv2.cv as cv#or simply import cv

im = cv.LoadImage("img/lena.jpg")
im2 = cv.LoadImage("img/fruits-larger.jpg")
cv.ShowImage("Image1", im)
cv.ShowImage("Image2", im2)

res = cv.CreateImage(cv.GetSize(im2), 8, 3)

cv.Add(im, im2, res) #Add every pixels together (black is 0 so low change and whi
cv.ShowImage("Add", res)

cv.AbsDiff(im, im2, res) # Like minus for each pixel im(i) - im2(i)
cv.ShowImage("AbsDiff", res)

cv.Mul(im, im2, res) #Multiplie each pixels (almost white)
cv.ShowImage("Mult", res)

cv.Div(im, im2, res) #Values will be low so the image will likely to be almost bl
cv.ShowImage("Div", res)

cv.And(im, im2, res) #Bit and for every pixels
cv.ShowImage("And", res)

cv.Or(im, im2, res) # Bit or for every pixels
cv.ShowImage("Or", res)

cv.Not(im, res) # Bit not of an image
cv.ShowImage("Not", res)

cv.Xor(im, im2, res) #Bit Xor
cv.ShowImage("Xor", res)
```

Оригинальные изображения:

Добавлять:

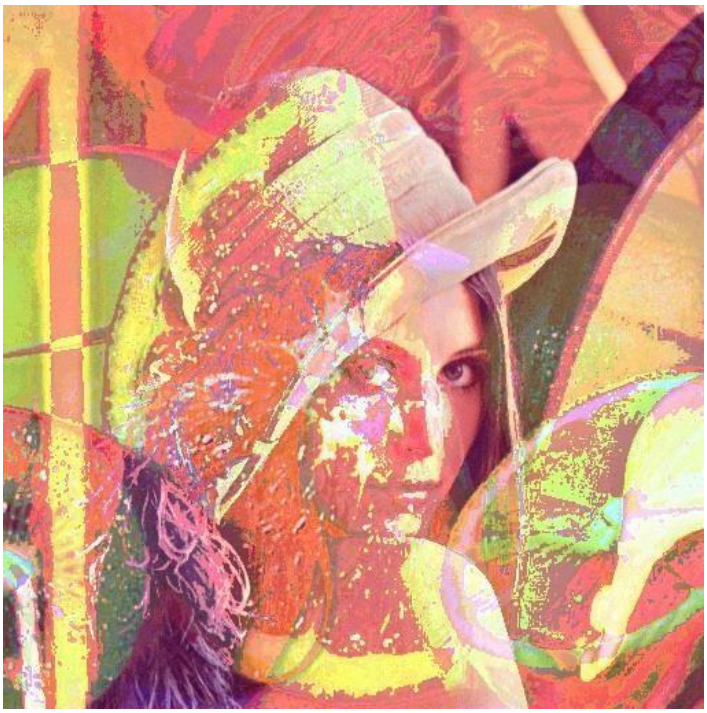
AbsDiff:

Мул:



Раздел:

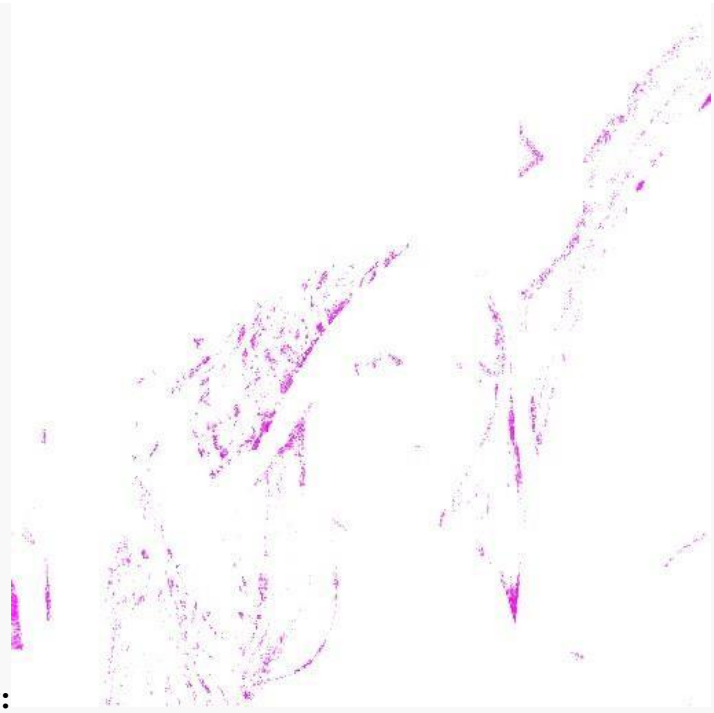
And:

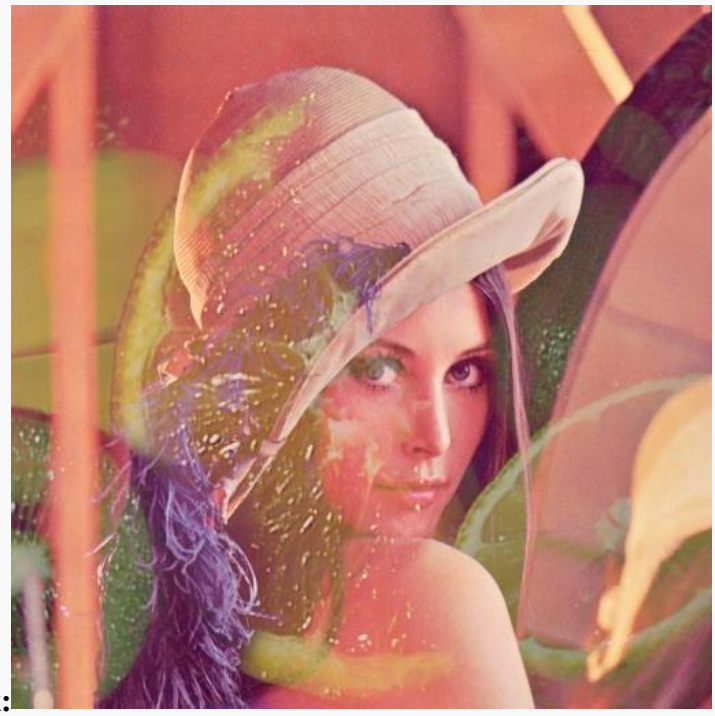


Or:

Not: 

Xor: 

Pow: 

Max: 

Вопросы для самоконтроля:

1. Какие методы используются для сегментации изображений в OpenCV и как они работают? 2. Какие методы используются для обнаружения границ на изображении в OpenCV и как они работают?
3. Какие методы используются для улучшения качества изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 3 ДОСТУП К ПИКСЕЛЯМ И МАТРИЧНЫЕ ИТЕРАЦИИ

Задача: научиться пользоваться соответствующими функциями открытой библиотеки OpenCV

а. Пиксельный доступ

Существует несколько способов доступа к пикселю. Они есть:

- Лучший способ - сделать как для многомерного списка. Для изображения «im», если вы хотите получить доступ к пикселю с координатой (3,3), просто выполните: im[3,3]. Это в основном возвращает кортеж значений. Кортеж из 3 значений от 0 до 255 для цветных изображений.
- Или вы можете использовать функции OpenCV: Get1D для одномерного изображения, Get2D для двухмерного массива и т. д.

б. Доступ к строкам и столбцам

Вы также можете получить доступ к строке или столбцу, даже к набору строк или набору столбцов. Чтобы получить его, вы должны использовать функции, предоставляемые opencv, а именно:

- cv.GetCol(im, 0): возвращает первый столбец в виде одномерного массива.
- cv.GetCols(im, 0, 10): возвращает 10 первых столбцов в виде матрицы
- cv.GetRow(im, 0): вернуть первую строку cv.GetRows(im, 0, 10): вернуть 10 первых строк

в. Повторить матрицу

Самый простой способ перебрать матрицу — зациклить ее следующим образом:

```
import cv2.cv as cv

im = cv.LoadImage("img/lena.jpg")

for i in range(im.height):
    for j in range(im.width):
        im[i,j] #Do whatever you want with your pixel
```

Другой способ перебора матрицы — использовать LineIterator, который позволяет перебирать матрицу. Вы должны указать координату начальной точки и координату конечной точки. Таким образом, он будет перебирать каждый пиксель между двумя точками.

в. Поиграем с пикселями

Этот бесполезный пример показывает пример программы, которая заменяет случайные пиксели на изображении случайным цветом.

```
1
2
3
4import cv2.cv as cv
5
6import random
7
8im = cv.LoadImage("img/lena.jpg") #or LoadImage and access pixel with Get2D/Set2D 9
10for k in range(5000): #Create 5000 noisy pixels
11    i = random.randint(0,im.height-1)
12    j = random.randint(0,im.width-1)
13    color = (random.randrange(256),random.randrange(256),random.randrange(256)) 14    im[i,j] = color
    cv.ShowImage("Noize", im) cv.WaitKey(0)
```

д. Расстояние

Программа ниже показывает дистанционную технику в действии. Этот метод вычисляет расстояние данного пикселя от заданного ссылочного цвета и позволяет изолировать его, если это так. В этом примере эталонный цвет будет черным (0,0,0), а минимальное расстояние будет равно 100. Таким образом, все пиксели, расстояние которых меньше 100, будут заменены другим цветом. В нашем случае белый. Этот метод позволяет изолировать диапазон цветов и применить операцию к совпадающим пикселям.

```
1import cv2.cv as cv
2
3im = cv.LoadImageM("img/fruits.jpg",cv.CV_32F)
4
```

```
5 def getDistance(pixel,refcolor):
6     return abs( (pixel[0]-refcolor[0]) + (pixel[1]-refcolor[1]) + (pixel[2]-refcolor[2]) )
7
8
9 refcolor = (0,0,0)
10 minDist = 100
11
12 for row in range(im.rows):
13         for col in range(im.cols):
14         if getDistance(im[row,col], refcolor)&lt;minDist:
15         im[row,col] = (255,255,255)
16
17 cv.ShowImage("Distance", im)
18
19 cv.WaitKey(0)
```

Вопросы для самоконтроля

1.Какие методы используются для обработки видео в OpenCV и как они работают?

2.Какие методы используются для распознавания лиц на изображениях в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 4 ГИСТОГРАММА И ОБРАТНАЯ ПРОЕКЦИЯ

Задача: научиться пользоваться соответствующими функциями открытой библиотеки OpenCV

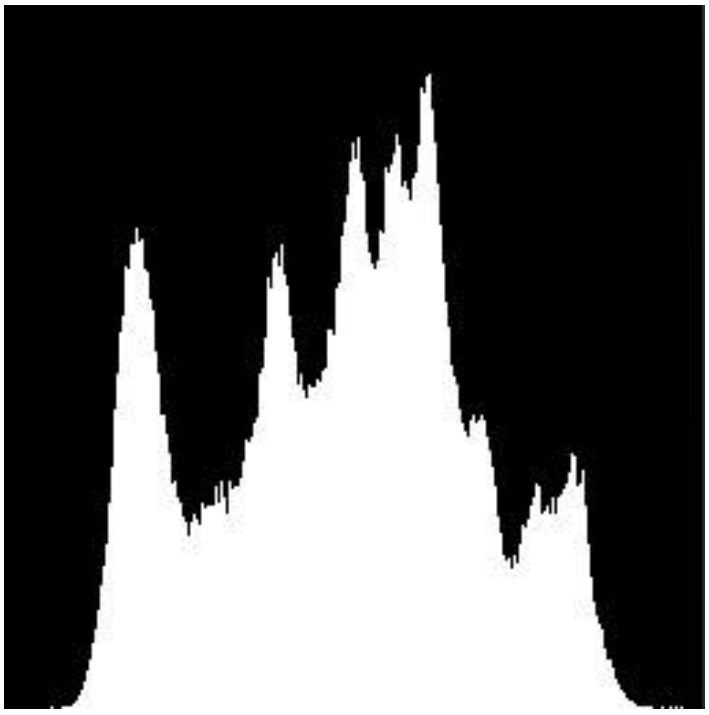
а. Гистограмма

Гистограмма спектр распределения интенсивности. Конкретно это список, содержащий для каждого возможного значения пикселя количество пикселей, имеющих это значение. Вы можете сказать хорошо, но в конце концов, как это может быть действительно полезно? для каких целей? Например, вычислите гистограмму области на фотографии, тогда та же область на другой фотографии с другой точки зрения, например, с большей вероятностью будет иметь ту же гистограмму. Так что может быть действительно очень полезно сделать распознавание образов «с нуля». Действительно, OpenCV предоставляет все необходимые функции для этого благодаря CalcHist и CompareHist, которые возвращают поплавок расстояния (точность).

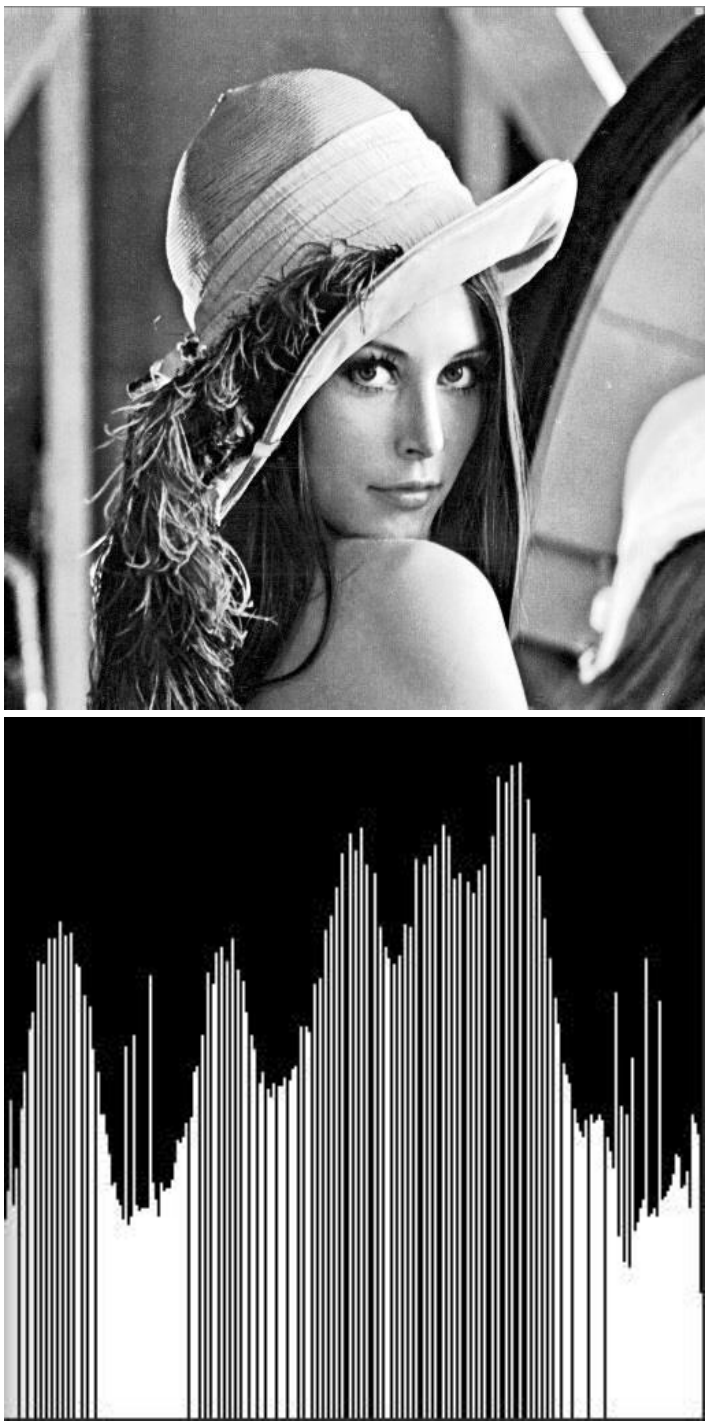
Как мы видели в главе о фильтрах, функция equalizehist изменяет изображение, чтобы сгладить гистограмму изображения, чтобы баланс между черным и белым был почти одинаковым. В приведенном ниже примере вычисляется гистограмма серого изображения и отображается график связанной гистограммы (на другом изображении), затем выполняется выравнивание изображения и пересчет гистограммы, чтобы показать разницу.

```
import cv2.cv as cv

def drawGraph(ar,im, size): #Draw the histogram on the image
    minV, maxV, minloc, maxloc = cv.MinMaxLoc(ar) #Get the min and max value
    hpt = 0.9 * histsize
    for i in range(size):
        intensity = ar[i] * hpt / maxV #Calculate the intensity to make enter in
        cv.Line(im, (i,size), (i,int(size-intensity)),cv.Scalar(255,255,255)) #D
        i += 1

#---- Gray image
orig = cv.LoadImage("img/lena.jpg", cv.CV_8U)

histsize = 256 #Because we are working on grayscale pictures which values within

hist = cv.CreateHist([histsize], cv.CV_HIST_ARRAY, [[0,histsize]], 1)

cv.CalcHist([orig], hist) #Calculate histogram for the given grayscale picture

histImg = cv.CreateMat(histsize, histsize, cv.CV_8U) #Image that will contain th
drawGraph(hist.bins, histImg, histsize)

cv.ShowImage("Original Image", orig)
cv.ShowImage("Original Histogram", histImg)
#--------------------

#---- Equalized image
imEq = cv.CloneImage(orig)
cv.EqualizeHist(imEq, imEq) #Eqlize the original image

histEq = cv.CreateHist([histsize], cv.CV_HIST_ARRAY, [[0,histsize]], 1)
cv.CalcHist([imEq], histEq) #Calculate histogram for the given grayscale picture
eqImg = cv.CreateMat(histsize, histsize, cv.CV_8U) #Image that will contain the
drawGraph(histEq.bins, eqImg, histsize)

cv.ShowImage("Image Equalized", imEq)
cv.ShowImage("Equalized HIstogram", eqImg)
#--------------------------------
```

Исходное изображение:



Выровненное изображение:



б. обратная проекция

Обратная проекция рассчитывается по гистограмме. Он в основном заменяет каждый пиксель вероятностью его появления на изображении. Его не очень полезно использовать вручную, но он используется другим алгоритмом, таким как среднее смещение. В приведенном ниже примере показано обычное использование обратной проекции, связанной с интересующей областью. В этом примере мы выбираем прямоугольник в верхнем левом углу изображения, вычисляем гистограмму, а затем применяем обратную проекцию ко всему изображению, чтобы обнаружить другие части изображения, которые имеют такую же гистограмму.

```
import cv2.cv as cv

im = cv.LoadImage("img/lena.jpg", cv.CV_8U)

cv.SetImageROI(im, (1, 1,30,30))

histsize = 256 #Because we are working on grayscale pictures
hist = cv.CreateHist([histsize], cv.CV_HIST_ARRAY, [[0,histsize]], 1)
cv.CalcHist([im], hist)


cv.NormalizeHist(hist,1) # The factor rescale values by multiplying values by th
_,max_value,_,_ = cv.GetMinMaxHistValue(hist)

if max_value == 0:
    max_value = 1.0
cv.NormalizeHist(hist,256/max_value)

cv.ResetImageROI(im)

res = cv.CreateMat(im.height, im.width, cv.CV_8U)
cv.CalcBackProject([im], res, hist)

cv.Rectangle(im, (1,1), (30,30), (0,0,255), 2, cv.CV_FILLED)
cv.ShowImage("Original Image", im)
cv.ShowImage("BackProjected", res)
cv.WaitKey(0)
```

Примечание: результат более или менее точен. Чтобы получить лучший результат, вы можете сделать это на каждом канале цветного изображения, чтобы получить гораздо более точный результат.

Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?
3. Какие методы используются для обработки изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 5 ОБНАРУЖЕНИЕ ЛИНИЙ, КРАЕВ И КОНТУРОВ

а. Алгоритм Лапласа

Я не буду объяснять, как работает алгоритм Лапласа, но единственное, что вам нужно знать, это то, что этот алгоритм является первым шагом к обнаружению линий и ребер. Это также используется более сложным алгоритмом, включенным в OpenCV.

В следующем примере показано, как применить алгоритм Лапласа как к серому, так и к цветному изображению. Как мы видим, на результирующих изображениях начинают появляться линии и края.

```
import cv2.cv as cv

im=cv.LoadImage('img/building.png', cv.CV_LOAD_IMAGE_COLOR)

# Laplace on a gray scale picture
gray = cv.CreateImage(cv.GetSize(im), 8, 1)
cv.CvtColor(im, gray, cv.CV_BGR2GRAY)

aperture=3

dst = cv.CreateImage(cv.GetSize(gray), cv.IPL_DEPTH_32F, 1)
cv.Laplace(gray, dst,aperture)

cv.Convert(dst,gray)

thresholded = cv.CloneImage(im)
cv.Threshold(im, thresholded, 50, 255, cv.CV_THRESH_BINARY_INV)

cv.ShowImage('Laplaced grayscale',gray)
#-----------------------------------

# Laplace on color
planes = [cv.CreateImage(cv.GetSize(im), 8, 1) for i in range(3)]
laplace = cv.CreateImage(cv.GetSize(im), cv.IPL_DEPTH_16S, 1)
colorlaplace = cv.CreateImage(cv.GetSize(im), 8, 3)

cv.Split(im, planes[0], planes[1], planes[2], None) #Split channels to apply
for plane in planes:
    cv.Laplace(plane, laplace, 3)
    cv.ConvertScaleAbs(laplace, plane, 1, 0)

cv.Merge(planes[0], planes[1], planes[2], None, colorlaplace)

cv.ShowImage('Laplace Color', colorlaplace)
#-----------------------------------

cv.WaitKey(0)
```

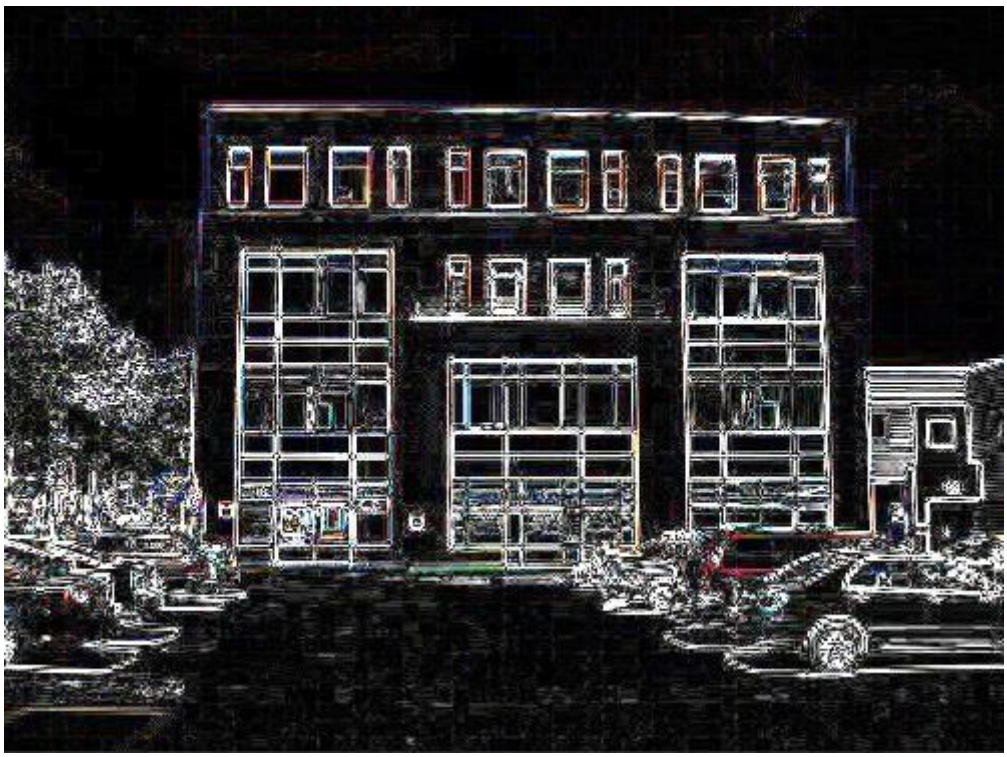
Дополнительная информация:

- Используемый размер апертуры является значением по умолчанию. Это размер ядра, лучше оставить это значение, если вы не знаете, что делаете.
- Чтобы применить Лаплас к цветному изображению, мы должны разделить все каналы, а затем объединить их обратно.
- Как видно из результатов, разница между серым изображением и цветным изображением несущественна. Так что работать с изображением в градациях серого интереснее, чем с цветными.

Исходное изображение:

Лапласов серый образ:

Цветное изображение Лапласа:



б. Sobel

Sobel — это хорошо известный алгоритм, используемый для обнаружения контуров. Для получения дополнительной информации об этом алгоритме посетите страницу Википедии . Небольшой пример ниже использует алгоритм Собеля, чтобы показать контуры сборки, которая работает достаточно хорошо.

```
1import cv2.cv as cv
2
3im=cv.LoadImage('img/building.png', cv.CV_LOAD_IMAGE_GRAYSCALE)
4
5sobx = cv.CreateImage(cv.GetSize(im), cv.IPL_DEPTH_16S, 1)
6cv.Sobel(im, sobx, 1, 0, 3) #Sobel with x-order=1
7
8soby = cv.CreateImage(cv.GetSize(im), cv.IPL_DEPTH_16S, 1)
9cv.Sobel(im, soby, 0, 1, 3) #Sobel withy-oder=1
10
11cv.Abs(sobx, sobx)
12cv.Abs(soby, soby)
13
14result = cv.CloneImage(im)
15cv.Add(sobx, soby, result) #Add the two results together.
16
17cv.Threshold(result, result, 100, 255, cv.CV_THRESH_BINARY_INV) 18
19cv.ShowImage('Image', im)
20cv.ShowImage('Result', result)
```

```
21
22cv.WaitKey(0)
```

в. Еще один алгоритм

Последняя базовая функция, которая будет представлена для базового обнаружения краев/контуров, — это *cv.MorphologyEx* . Документация, доступная здесь, объясняет, как работают все аргументы, но нас здесь интересует *CV_MOP_GRADIENT* , который расширяет и вычитает результат для размытия. Применение этих фильтров в указанном порядке должно привести к освобождению всех контуров и краев изображения. Это основано на том, что сравнение изображения и эрозии будет в основном отличаться по расположению краев (где интенсивность соседей различается больше). В качестве аргумента мы также можем предоставить элемент структурирования, такой как крестик и ромб, к которому применяются фильтры, а также количество итераций. Дополнительные итерации могут дать более точные контуры, но также могут стереть некоторые из них.

```
 1import cv2.cv as cv
 2
 3image=cv.LoadImage('img/build.png', cv.CV_LOAD_IMAGE_GRAYSCALE)
 4
 5#Get edges
 6morphed = cv.CloneImage(image)
 7cv.MorphologyEx(image, morphed, None, None, cv.CV_MOP_GRADIENT) # Apply a dilate - Erode 8
 9cv.Threshold(morphed, morphed, 30, 255, cv.CV_THRESH_BINARY_INV)
10
11cv.ShowImage("Image", image)
12cv.ShowImage("Morphed", morphed)
```

13

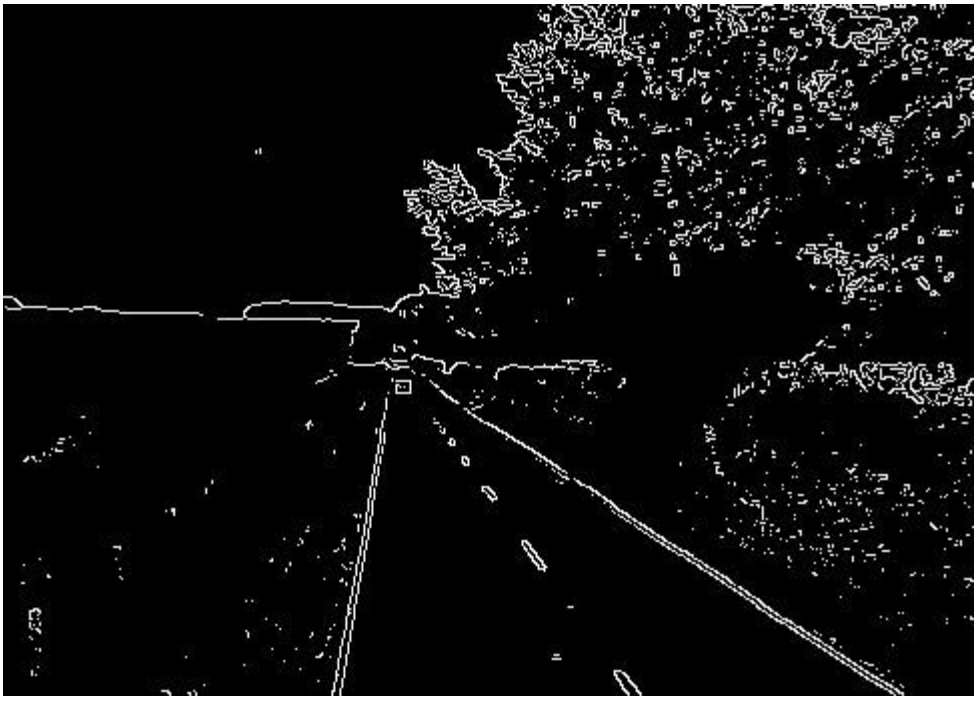
14cv.WaitKey(0)

д. Обнаружение линий с помощью Canny

Canny — это алгоритм, созданный для обнаружения границ. Это базовый алгоритм для определения края любой линии или контура из-за его точности и простоты использования. Пример, представленный ниже, покажет, как обнаружить линии на изображении с помощью алгоритма хитрости. Обратите внимание, что алгоритм canny использует алгоритм Sobel в фоновом режиме. Чтобы обнаружить линии на изображении, мы будем использовать cv.HoughLines2, который выполняет работу по поиску линий из «консервированного» изображения. В примере показан результат с использованием стандартных HoughLines и вероятностного способа.

Исходное изображение:

Осторожный образ:



Стандартные HoughLines:

Вероятностные HoughLines:

e. Обнаружение контуров

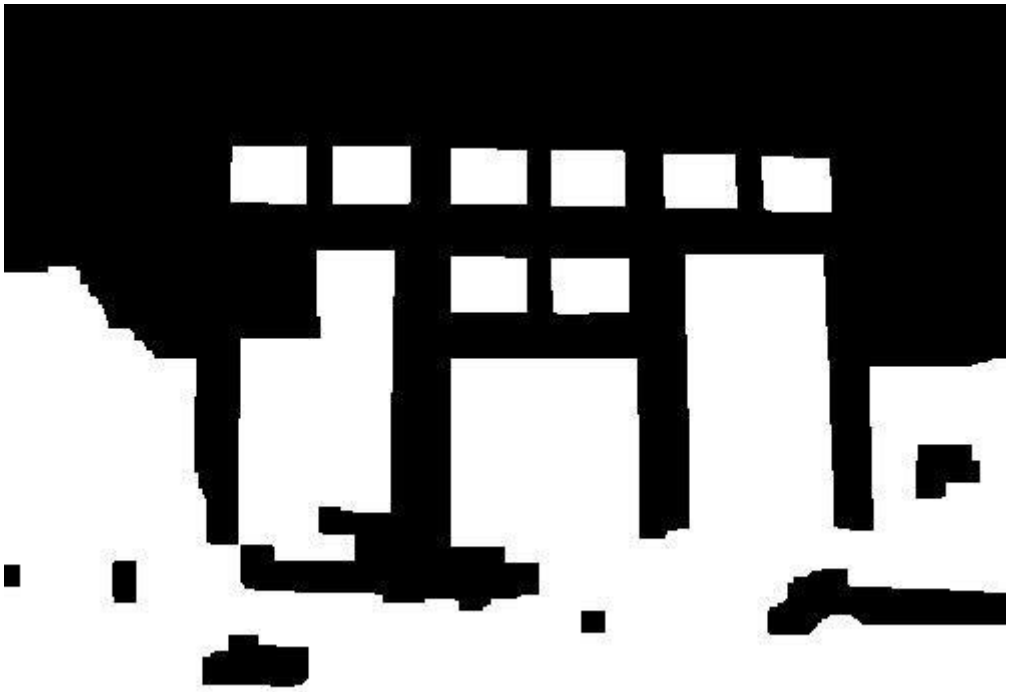
Для обнаружения контуров OpenCV предоставляет функцию *FindContours* , которая предназначена для поиска контуров на изображении. Конечно, к изображению должна быть применена определенная обработка, чтобы получить хорошее определение контуров. В приведенном ниже примере мы сначала используем функцию *MorphologyEx* с методом *CV_MOP_OPEN* и *CV_MOP_CLOSE* для освобождения контуров. Затем мы применяем функцию FindContours, чтобы найти контуры и распечатать их на цветном изображении, даже если мы работаем с версией изображения в градациях серого.

Дополнительная информация:

- Изображение сначала преобразуется в серый цвет, потому что мы привыкли работать с изображением в градациях серого.
- Для грубого получения контуров применяется простой порог.
- Трюк *OPEN* и *CLOSE* применяется для получения более гладких контуров.
- Вызывается функция FindContours и извлекается список контуров.
- Контуры окончательно нарисованы цветом, как если бы мы работали над ним. **Исходное изображение:**

После первого порога:

После фильтров MorphologyEx и второго порога:



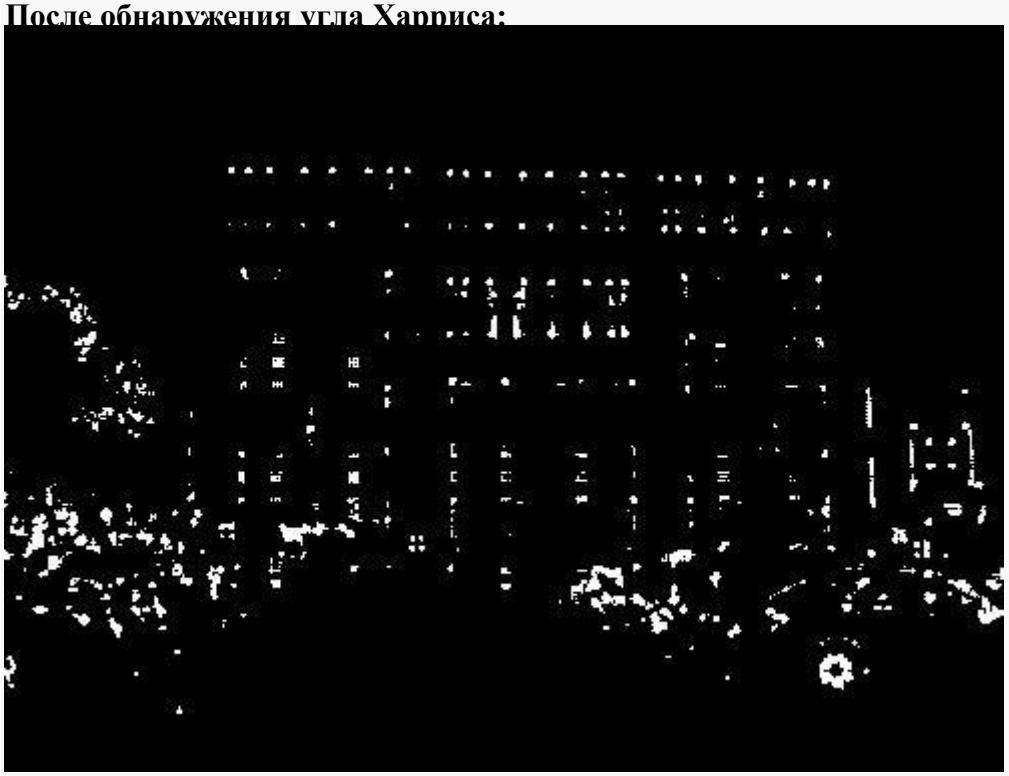
Нарисованы контуры:

ф. Обнаружение краев

Последняя функция, которая нас интересует, — это обнаружение границ, которое будет реализовано с помощью алгоритма Харриса. Для получения дополнительной технической информации об этом алгоритме посетите страницу википедии по этому адресу . В этом примере мы применим cv.CornerHarris к изображению, которое примерно возвращает углы. Затем применяются различные фильтры для

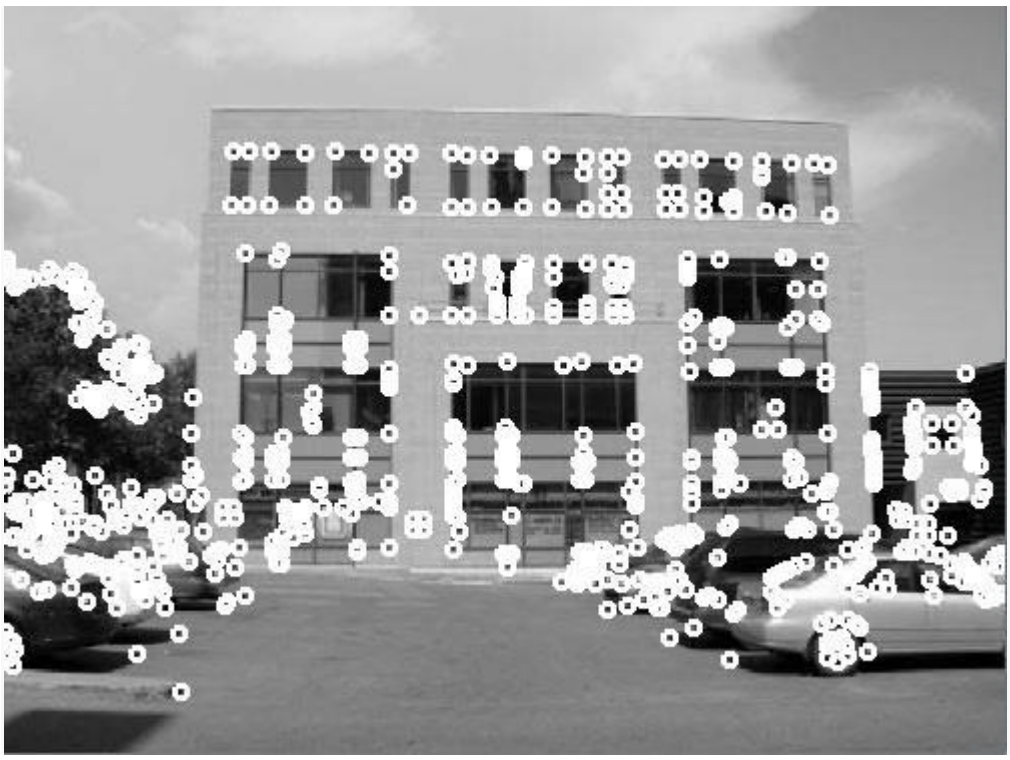
уменьшения необработанных углов, чтобы оставить только пиксель для каждого угла, из которого мы извлекаем координаты, а затем рисуем их на исходном изображении.

После обнаружения угла Харриса:



Отдельные точки Углы:

Нарисованы углы:



Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?
3. Какие методы используются для обработки изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 6 ОБНАРУЖЕНИЕ ОБЪЕКТОВ

a. Хорошая функция SURF

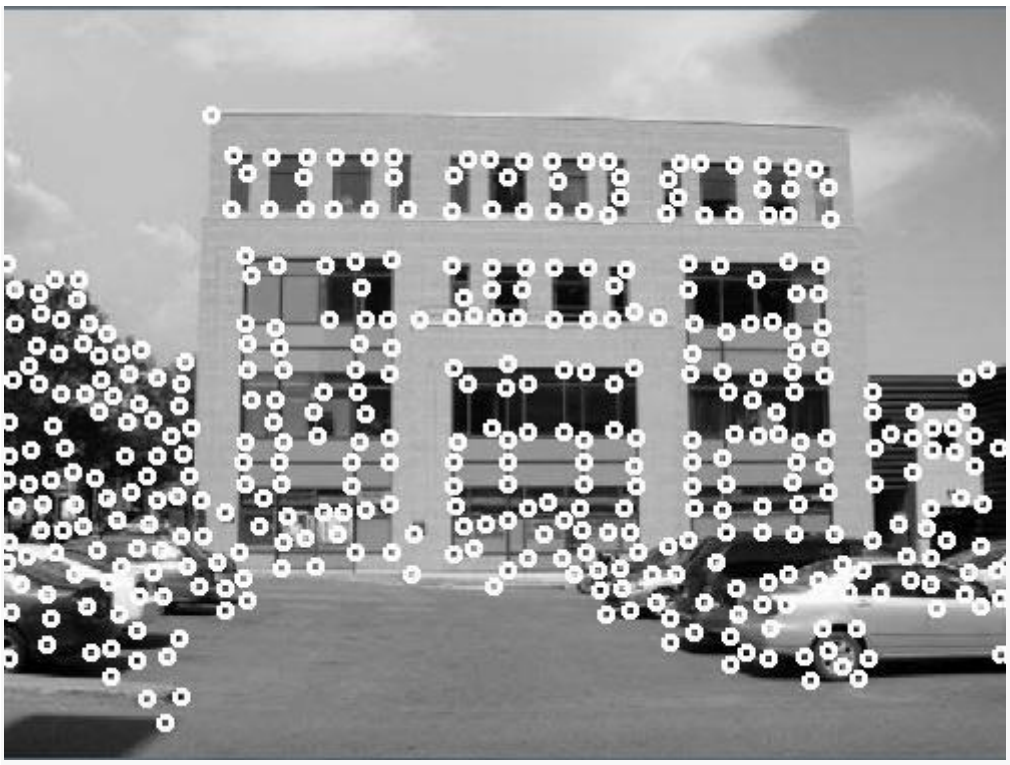
В этом разделе нас интересуют две функции: *GoodFeaturesToTrack* и *ExtractSURF* . Обе они являются функциями высокого уровня, которые позволяют обнаруживать особенности и особенно углы. Ниже приведен пример того, как их использовать самым простым способом, но перед некоторыми пояснениями об этих функциях.

- GoodFeaturesToTrack : эта функция специально разработана для обнаружения углов. По умолчанию он использует алгоритм CornerMinEigenVal, но вы можете изменить его на использование CornerHarris (как показано выше), изменив значение параметра. Основное отличие алгоритма Харриса заключается в том, что вы должны указать минимальное расстояние между каждой точкой, уровень качества и количество углов для обнаружения. Так что в какой-то момент он автоматически делает то, что было сделано в предыдущей главе с примером Харриса.

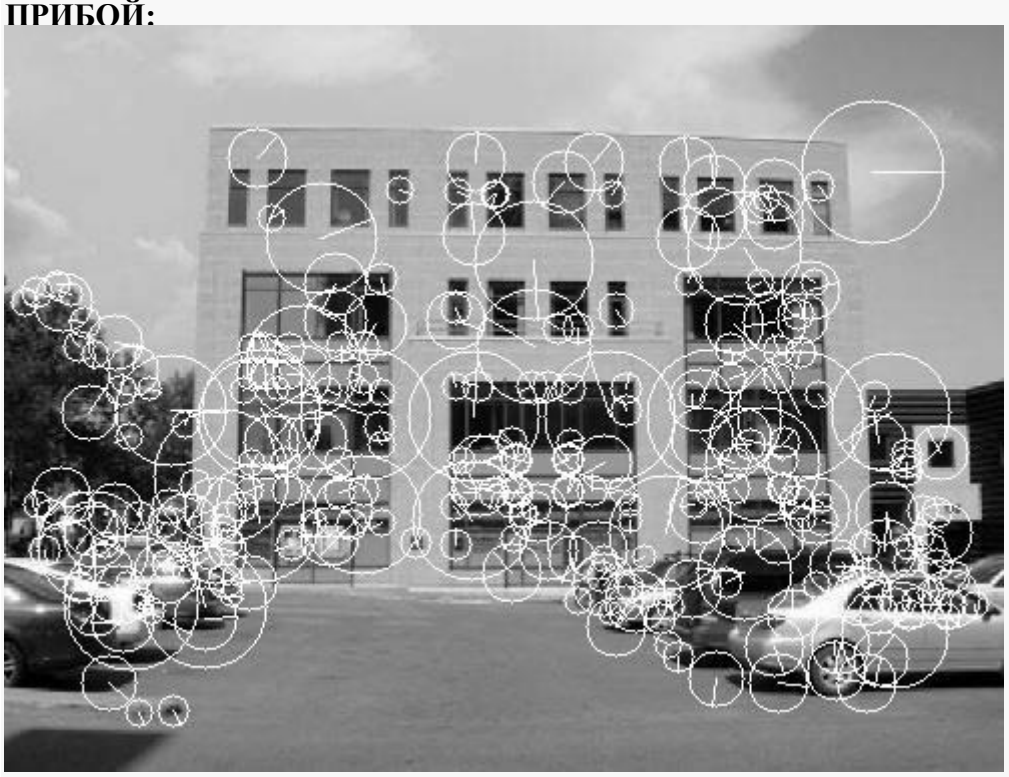
- SURF : Этот алгоритм также специализируется на обнаружении элементов, но предоставляет больше информации, например, ориентацию углов.

```
import cv2.cv as cv
import math

im = cv.LoadImage("img/church.png", cv.CV_LOAD_IMAGE_GRAYSCALE)
im2 = cv.CloneImage(im)

# Goodfeatureto track algorithm
eigImage = cv.CreateMat(im.height, im.width, cv.IPL_DEPTH_32F)
tempImage = cv.CloneMat(eigImage)
cornerCount = 500
quality = 0.01
minDistance = 10

corners = cv.GoodFeaturesToTrack(im, eigImage, tempImage, cornerCount, quality,

radius = 3
thickness = 2

for (x,y) in corners:
    cv.Circle(im, (int(x),int(y)), radius, (255,255,255), thickness)

cv.ShowImage("GoodfeaturesToTrack", im)

#SURF algorithm
hessthresh = 1500 # 400 500
dsize = 0 # 1
layers = 1 # 3 10

keypoints, descriptors = cv.ExtractSURF(im2, None, cv.CreateMemStorage(), (dsize
for ((x, y), laplacian, size, dir, hessian) in keypoints:
    cv.Circle(im2, (int(x),int(y)), cv.Round(size/2), (255,255,255), 1)
    x2 = x+((size/2)*math.cos(dir))
    y2 = y+((size/2)*math.sin(dir))
    cv.Line(im2, (int(x),int(y)), (int(x2),int(y2)), (255,255,255), 1)

cv.ShowImage("SURF ", im2)

cv.WaitKey(0)
```

GoodFeatureToTrack:



ПРИБОЙ:

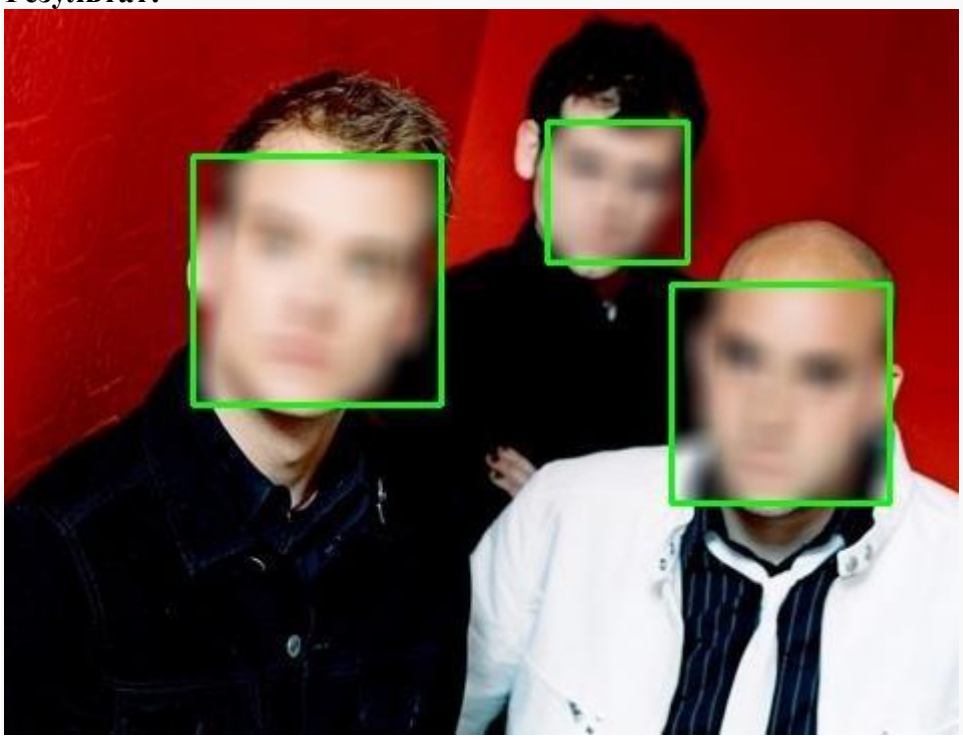


б. Распознавание лиц

Учитывая все, что мы делали раньше, вы можете подумать, что обнаружить лица на картинке довольно сложно, но на самом деле с HaarDetectObjects это довольно *просто* . Эта функция принимает XML -файл в качестве аргумента, описывающего обнаруживаемую функцию. Надеемся, что OpenCV предоставляет несколько XML-файлов для обнаружения таких функций, как лицо, глаза, ушная мышь или нос. Использование этой функции простое, и в приведенном ниже примере показана программа, которая размывает лицо на изображении.

```
import cv2.cv as cv
1
2       image=cv.LoadImage('img/alkaline.jpg', cv.CV_LOAD_IMAGE_COLOR)
3
4
#Load the haar cascade
5       hc = cv.Load("haarcascades/haarcascade_frontalface_alt.xml")
6
7
#Detect face in image
8       faces = cv.HaarDetectObjects(image, hc, cv.CreateMemStorage(), 1.2,2,
9       cv.CV_HAAR_DO_CANNY_PRUNING, (0,0) )
10
11
for ((x,y,w,h),stub) in faces:
12
13
    face = cv.GetSubRect(image, (x,y,w,h)) #Get the coordinate of the face in a rectangle
14
15
    cv.Smooth(face,face,cv.CV_BLUR, 15,15) #Blur the face for instance
16
17
    cv.Rectangle(image,(int(x),int(y)),(int(x)+w,int(y)+h),(0,255,0),2,0) #Draw a rectangle around the face
18
19      cv.ShowImage("Face      detect",
image)
20
cv.WaitKey(0)
```

Результат:



Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?

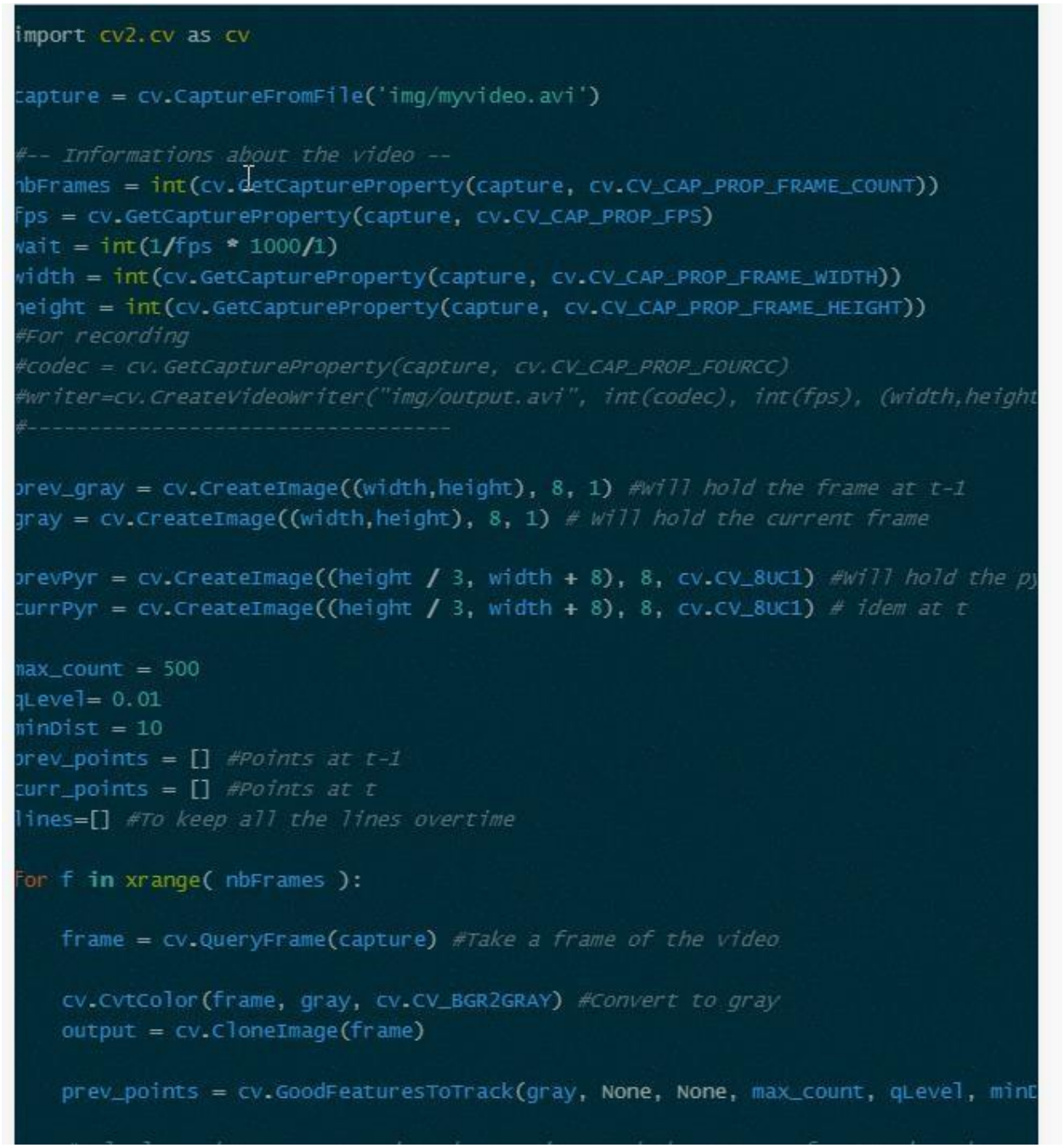
3. Какие методы используются для обработки изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 7 ОТСЛЕЖИВАНИЕ ДВИЖУЩЕГОСЯ ОБЪЕКТА

а. Отслеживание, тривиальный способ

Трек движения в OpenCV основан на функции *CalcOpticalFlowPyrLK* , которая вычисляет поток между изображением и позволяет отслеживать движение объекта. Следующая программа работает, как описано ниже, и я использовал видео, где простой объект пересекает экран слева направо.

- Запрошен кадр из видео
- *GoodFeaturesToTrack* вызывается для поиска интересных точек на кадре
- CalcOpticalFlowPyrLK вычисляется с использованием предыдущего кадра и вновь найденных точек, что позволяет алгоритму вычислить движение .
- Точки, которые не переместились, удаляются из списка точек
- Затем проводится линия между старым положением точки и новым положением точки.
- Наконец, все линии сохраняются в списке, чтобы их можно было нарисовать на следующих кадрах.



```
import cv2.cv as cv

capture = cv.CaptureFromFile('img/myvideo.avi')

#-- Informations about the video --
nbFrames = int(cv.GetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_COUNT))
fps = cv.GetCaptureProperty(capture, cv.CV_CAP_PROP_FPS)
wait = int(1/fps * 1000/1)
width = int(cv.GetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_WIDTH))
height = int(cv.GetCaptureProperty(capture, cv.CV_CAP_PROP_FRAME_HEIGHT))
#For recording
#codec = cv.GetCaptureProperty(capture, cv.CV_CAP_PROP_FOURCC)
#writer=cv.CreateVideoWriter("img/output.avi", int(codec), int(fps), (width,height
#---------------------------------

prev_gray = cv.CreateImage((width,height), 8, 1) #Will hold the frame at t-1
gray = cv.CreateImage((width,height), 8, 1) # will hold the current frame

prevPyr = cv.CreateImage((height / 3, width + 8), 8, cv.CV_8UC1) #Will hold the py
currPyr = cv.CreateImage((height / 3, width + 8), 8, cv.CV_8UC1) # idem at t

max_count = 500
qLevel= 0.01
minDist = 10
prev_points = [] #Points at t-1
curr_points = [] #Points at t
lines=[] #To keep all the lines overtime

for f in xrange( nbFrames ):

    frame = cv.QueryFrame(capture) #Take a frame of the video

    cv.CvtColor(frame, gray, cv.CV_BGR2GRAY) #Convert to gray
    output = cv.CloneImage(frame)

    prev_points = cv.GoodFeaturesToTrack(gray, None, None, max_count, qLevel, mint
```

Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?
3. Какие методы используются для обработки изображений в OpenCV и как они работают?

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Маккинли, Уэс. Python и анализ данных / Уэс Маккинли ; перевод А. Слинкина. - Python и анализ данных,2024-10-28. - Электрон. дан. (1 файл). - Саратов : Профобразование, 2019. - 482 c. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-54488-0046-7

2. Рик, Гаско. Простой Python просто с нуля / Гаско Рик. - Простой Python просто с нуля,2022-04-10. - Электрон. дан. (1 файл). - Москва : СОЛОН-Пресс, 2019. - 256 c. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-91359-334-4

3. Амоа, К. А. Разработка программных пакетов на языке Python : учебное пособие / К. А. Амоа, Н. А. Рындин, Ю. С. Скворцов. - Разработка программных пакетов на языке Python,2026-05-28. - Электрон. дан. (1 файл). - Воронеж : Воронежский государственный технический университет, ЭБС АСВ, 2020. - 61 c. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-57731-0887-0

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ
Анализ и обработка больших данных

Направление подготовки	**09.04.02 Информационные системы и технологии**
Направленность (профиль)	**«Технологии работы с данными и знаниями, анализ информации»**
Квалификация выпускника	Магистр

Пятигорск, 2025

Введение

Методические указания содержат перечень тем с вопросами для самостоятельной проработки, требования к оформлению работ и пример выполнения задания. Теоретической основой подготовки специалиста являются знания в области информатики, вычислительной систем.

1. Цель, задачи

Целью дисциплины «Анализ и обработка больших данных» является ознакомление с основными понятиями и методами использования теории формальных грамматик, овладение теоретическими знаниями о фазы грамматического разбора при компиляции и интерпретации формальных текстов и практическими навыками по алгоритмам лексического, грамматического и семантического анализа.

Задачи дисциплины: - изучение основных типов графиков и их применение для визуализации различных типов данных; - овладение навыками работы с библиотеками Matplotlib, OPEN CV и другими инструментами для создания качественных и информативных графиков; - понимание принципов распознавания и трекинга, выбора типа инструмента для конкретной задачи, настройки параметров OPEN CV для достижения требуемого эффекта; - развитие навыков анализа и интерпретации данных на основе визуального представления.

В настоящее время искусственные языки, использующие для описания предметной области текстовое представление, широко применяются не только в программировании, но и в других областях. С их помощью описывается структура всевозможных документов, трехмерных виртуальных миров, графических интерфейсов пользователя и многих других объектов, используемых в моделях и в реальном мире. Для того чтобы эти текстовые описания были корректно составлены, а затем правильно распознаны и интерпретированы, применяются специальные методы их анализа и преобразования.

В методических рекомендациях содержатся материалы, необходимые для самостоятельного изучения.

2. Технологическая карта самостоятельной работы обучающегося

Для студентов заочной формы обучения:

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Обьем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
4семестр					
УК-1(ИД1, ИД-2, ИД-3), УК-6(ИД1, ИД-2, ИД-3), ОПК-2 (ИД-1, ИД-2), ОПК-3 (ИД-1, ИД-2)	Подготовка к лекциям	Собеседование	0,36	0,04	0,4

УК-1(ИД1, ИД-2, ИД-3), УК-6(ИД1, ИД-2, ИД-3), ОПК-2 (ИД-1, ИД-2), ОПК-3 (ИД-1, ИД-2)	Самостоятельное изучение литературы	Собеседование	120,96	13,44	134,4
УК-1(ИД1, ИД-2, ИД-3), УК-6(ИД1, ИД-2, ИД-3), ОПК-2 (ИД-1, ИД-2), ОПК-3 (ИД-1, ИД-2)	Подготовка и выполнение практических работ	Отчет письменный	1,08	0,12	1,2
Итого за 4 семестр			**122,4**	**13,6**	**136**

3. Рекомендации для самоподготовки

3.1. Подготовка к лекциям. Самостоятельное изучение литературы

Подготовить доклад и презентацию по вопросам выносимым на самостоятельную работу.

Тема 1. Введение в библиотеки визуализации Matplotlib

Тема 2. Методы анализа в библиотеке OPEN CV

Тема 3. Трекинг объектов в OPEN CV

3.2. Подготовка к практическим работам работам

При подготовке к лабораторным работам необходимо проработать следующие вопросы:

ПРАКТИЧЕСКАЯ РАБОТА № 1 НАЧАЛО РАБОТЫ С БИБЛИОТЕКОЙ OPEN CV

Задача: научиться пользоваться соответствующими функциями открытой библиотеки OpenCV .

Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?

3. Какие методы используются для обработки изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 2 ФИЛЬТРЫ И АРИФМЕТИЧЕСКИЕ ДЕЙСТВИЯ С ИЗОБРАЖЕНИЯМИ

Задача: научиться пользоваться соответствующими функциями открытой библиотеки OpenCV

Вопросы для самоконтроля:

1. Какие методы используются для сегментации изображений в OpenCV и как они работают? 2. Какие методы используются для обнаружения границ на изображении в OpenCV и как они работают?

3. Какие методы используются для улучшения качества изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 3 ДОСТУП К ПИКСЕЛЯМ И МАТРИЧНЫЕ ИТЕРАЦИИ

Задача: научиться пользоваться соответствующими функциями открытой библиотеки OpenCV

Вопросы для самоконтроля

1.Какие методы используются для обработки видео в OpenCV и как они работают?

2.Какие методы используются для распознавания лиц на изображениях в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 4 ГИСТОГРАММА И ОБРАТНАЯ ПРОЕКЦИЯ

Задача: научиться пользоваться соответствующими функциями открытой библиотеки OpenCV

Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?

3. Какие методы используются для обработки изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 5 ОБНАРУЖЕНИЕ ЛИНИЙ, КРАЕВ И КОНТУРОВ

Принципы работы алгоритма Лапласа является первым шагом к обнаружению линий и ребер. Это также используется более сложным алгоритмом, включенным в OpenCV.

Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?

3. Какие методы используются для обработки изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 6 ОБНАРУЖЕНИЕ ОБЪЕКТОВ

Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?

3. Какие методы используются для обработки изображений в OpenCV и как они работают?

ПРАКТИЧЕСКАЯ РАБОТА № 7 ОТСЛЕЖИВАНИЕ ДВИЖУЩЕГОСЯ ОБЪЕКТА

Вопросы для самоконтроля

1. Какие основные функции входят в библиотеку OpenCV и для чего они используются? 2. Какие типы изображений поддерживает OpenCV и какие форматы файлов она может обрабатывать?

3. Какие методы используются для обработки изображений в OpenCV и как они работают?

4.УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Перечень основной и дополнительной литературы, необходимой для освоения дисциплины

Перечень основной литературы:

1. Маккинли, Уэс. Python и анализ данных / Уэс Маккинли ; перевод А. Слинкина. - Python и анализ данных,2024-10-28. - Электрон. дан. (1 файл). - Саратов : Профобразование, 2019. - 482 c. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-54488-0046-7
2. Рик, Гаско. Простой Python просто с нуля / Гаско Рик. - Простой Python просто с нуля,2022-04-10. - Электрон. дан. (1 файл). - Москва : СОЛОН-Пресс, 2019. - 256 c. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-91359-334-4

Перечень дополнительной литературы:

1. Амоа, К. А. Разработка программных пакетов на языке Python : учебное пособие / К. А. Амоа, Н. А. Рындин, Ю. С. Скворцов. - Разработка программных пакетов на языке Python,2026-05-28. - Электрон. дан. (1 файл). - Воронеж : Воронежский государственный технический университет, ЭБС АСВ, 2020. - 61 с. - электронный. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-57731-0887-0

Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины:

1. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий
2. http://www.iqlib.ru - интернет библиотека образовательных изданий, в которой собраны электронные учебники, справочные и учебные пособия;
3. http://www.biblioclub.ru - электронная библиотечная система «Университетская библиотека – online»: специализируется на учебных материалах для ВУЗов по научно-гуманитарной тематике.
4. http://window.edu.ru – образовательные ресурсы ведущих вузов.