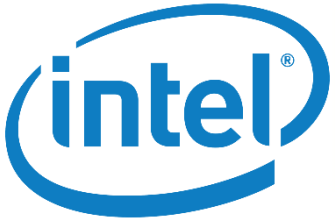
Северный (Арктический) федеральный университет имени М.В. Ломоносова
Высшая школа информационных технологий и авторизированных систем
Центр инновационного обучения

Intel Delta Course



Введение в разработку мультимедийных приложений с использованием библиотек OpenCV и OpenCV DNN

Докладчик: Ян Пустынный,
инженер отдела картографии и геоинформационных систем
ГАУ АО «Управление ИКТ АО», г. Архангельск

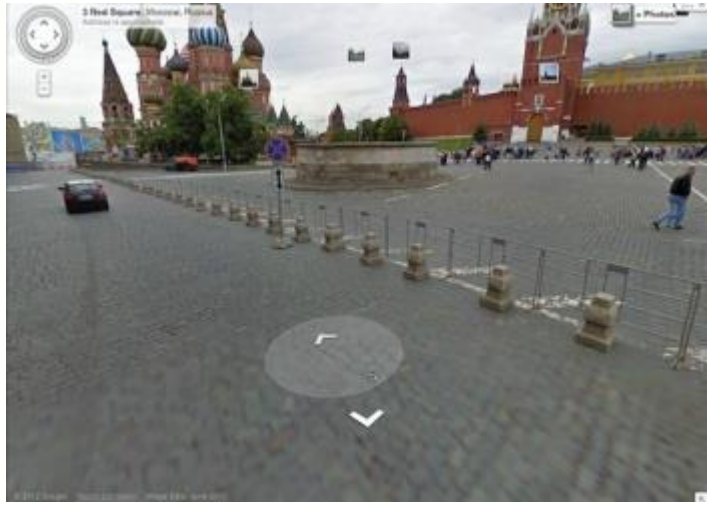
16 апреля 2019

Краткая справка

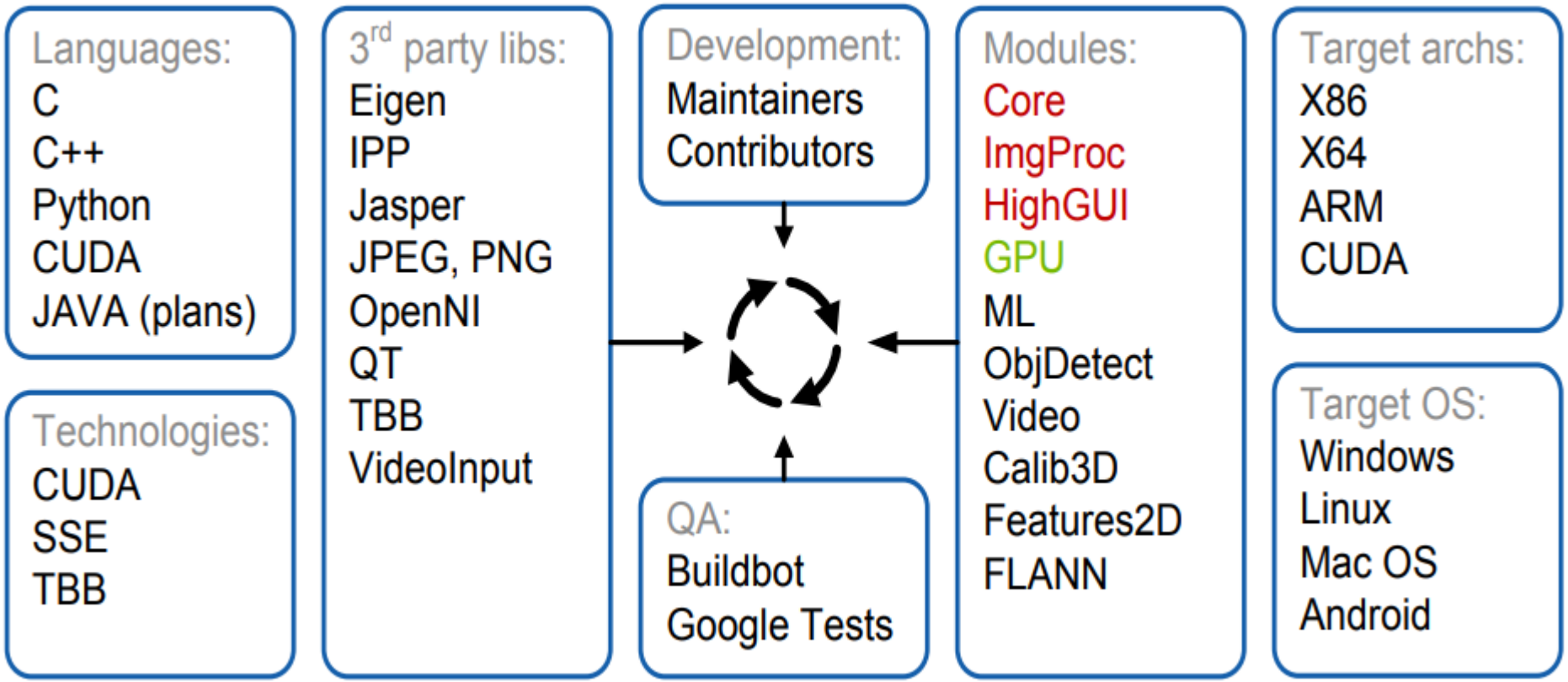
- Страничка: http://opencv.org
- Фактически, самая популярная библиотека компьютерного зрения.
- Написана на C/C++, исходный код открыт, включает более 1000 функций и алгоритмов.
- Лицензия BSD (разрешается бесплатное использование дома, для учебы, на работе)
- Разрабатывается с 1998г, сначала в Intel, теперь в компании Itseez при активном участии сообщества.
- Используется многими компаниями, организациями, ВУЗами, например в NVidia, Willow Garage, Intel, Google, Stanford …

Некоторые примеры использования

- Система зрения робота PR2, сделанного WillowGarage
- Аудио-визуальная инсталляция в Музее Современного Искусства (Сан-Франциско)
- Контроль качества монет, изготавливаемых Центробанком Китая
- Курсы компьютерного зрения в Стэнфорде
- Панорамы улиц в картах Google



Архитектура и разработка OpenCV



Использование

Базовая функциональность

A + B
Ax = B
FFT(A)
<?xml>...

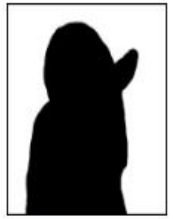
Обработка изображений


Фильтрация

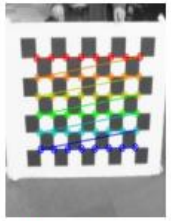

Трансформации


Ребра, контурный анализ


Особые точки


Сегментация

Видео, Стерео, 3D


Калибрация камер

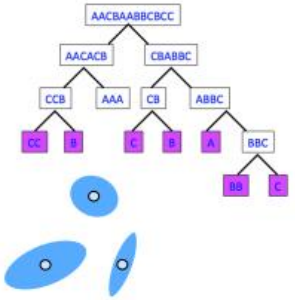

Вычисление положения в пространстве


Оптический поток

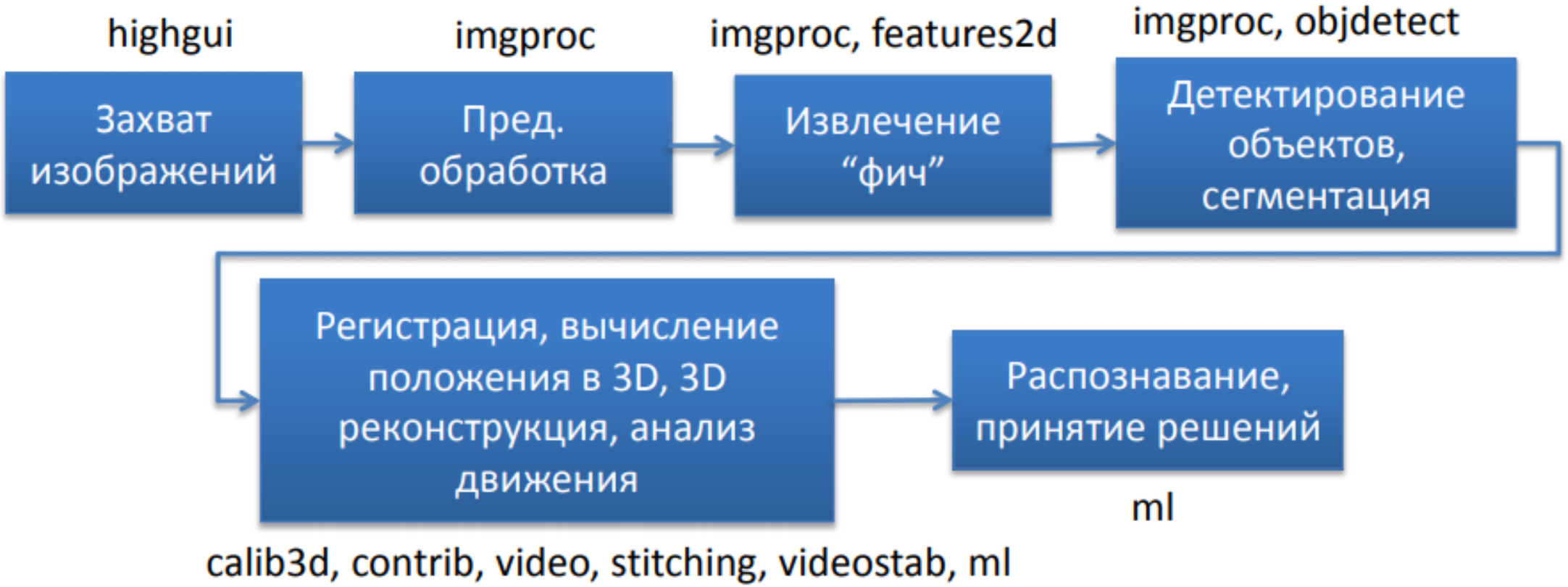

Построение карты глубины


Нахождение объектов

Машинное обучение



OpenCV в задачах компьютерного зрения



Как начать?

1. Качаем Python с http://python.org , качаем расширение Питона для вычислений NumPy: http://numpy.scipy.org (выбираем версию,соответствующую вашему Питону). Ставим и то и другое.
2. Устанавливаем OpenCV с помощью pip (нужно также предварительно установить) – pip install opencv-python, либо из самораспаковывающегося архива: https://sourceforge.net/projects/opencvlibrary/
3. Если брали sourceforge, то копируем cv2.pyd из распакованного каталога OpenCV в <каталог Python>\Lib\site-packages.
4. Можно начинать!
5. Опционально устанавливаем редактор с поддержкой Питона, например Sublime Text 3 или PyCharm.

Практическая работа №1

Набираем следующую программу в текстовом редакторе

```
import sys, cv2 as cv
img = cv.imread(sys.argv[1], 1)
cv.imshow("original", img)
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
gray = cv.GaussianBlur(gray, (7, 7), 1.5)
edges = cv.Canny(gray, 0, 50)
cv.imshow("edges", edges)
cv.waitKey()
```

practice1.py image.jpg

Практическая работа №2

```
import sys, cv2 as cv

img = cv.imread(sys.argv[1], 1)
cascade = cv.CascadeClassifier("haarcascade_frontalface_default.xml")
sf = min(640./img.shape[1], 480./img.shape[0])
gray = cv.resize(img, (0,0), None, sf, sf)
rects = cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=4,
minSize=(40, 40), flags=cv.CASCADE_SCALE_IMAGE)
for x, y, w, h in rects:
    cv.rectangle(gray, (x, y), (x+w, y+h), (0,0,255))

cv.imshow("edges+face", gray)
cv.waitKey()
```

Практическая работа №3

```
import cv2

scale_factor = 1.2
min_neighbors = 3
min_size = (50, 50)

def detect(path):

    cascade = cv2.CascadeClassifier(path)
    video_cap = cv2.VideoCapture("video.mp4")
    while True:
        # Capture frame-by-frame
        ret, img = video_cap.read()

        #конвертация в черно-белое для увеличения скорости работы
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        rects = cascade.detectMultiScale(gray, scaleFactor=scale_factor, minNeighbors=min_neighbors,minSize=min_size)
        # если обнаружено хотя бы одно лицо
        if len(rects) >= 0:
            # Рисуем прямоугольник вокруг лица
            for (x, y, w, h) in rects:
                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

            # Отображаем результат
            cv2.imshow('Face Detection on Video', img)
            if cv2.waitKey(1) & 0xFF == ord('c'):
                break
    video_cap.release()

def main():
    cascadeFilePath="haarcascade_frontalface_alt.xml"
    detect(cascadeFilePath)
    cv2.destroyAllWindows()


if __name__ == "__main__":
    main()
```

Практическая работа №4

```
import numpy as np
import argparse
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
	help="path to input image")
ap.add_argument("-p", "--prototxt", required=True,
	help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
	help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
	help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
```

Практическая работа №4

```
# Загружаем модель с диска
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# Загружаем изображение и создаем блоб
# уменьшаем размер до 300x300 пикселей и нормализуем
image = cv2.imread(args["image"])
(h, w) = image.shape[:2]
blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 1.0,
            (300, 300), (104.0, 177.0, 123.0))

# прогоняем блоб через сеть
print("[INFO] computing object detections...")
net.setInput(blob)
detections = net.forward()
```

Практическая работа №4

```
# Итерируем через все обнаруженные лица
for i in range(0, detections.shape[2]):
        # извлекаем число, характеризующее степень принадлежности обнаруженной детали к «лицу»
        confidence = detections[0, 0, i, 2]

        # отфильтровываем детали со слабой «принадлежностью»
        if confidence > args["confidence"]:
                # вычисляем координаты прямоугольника вокруг лица
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")

                # выводим на экран степень уверенности
                text = "{:.2f}%".format(confidence * 100)
                y = startY - 10 if startY - 10 > 10 else startY + 10
                cv2.rectangle(image, (startX, startY), (endX, endY),
                        (0, 0, 255), 2)
                cv2.putText(image, text, (startX, y),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)

cv2.imshow("Output", image)
cv2.waitKey(0)
```

Практическая работа №5

```
from imutils.video import VideoStream
import numpy as np
import argparse
import imutils
import time
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
                help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
                help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
                help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
time.sleep(2.0)
```

Практическая работа №5

```
while True:
        frame = vs.read()
        frame = imutils.resize(frame, width=400)

         (h, w) = frame.shape[:2]
        blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0,
                (300, 300), (104.0, 177.0, 123.0))

        net.setInput(blob)
        detections = net.forward()
```

```
for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]

        if confidence < args["confidence"]:
                continue
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        text = "{:.2f}%".format(confidence * 100)
        y = startY - 10 if startY - 10 > 10 else startY + 10
        cv2.rectangle(frame, (startX, startY), (endX, endY),
                (0, 0, 255), 2)
        cv2.putText(frame, text, (startX, y),
                cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255), 2)
```

Спасибо за внимание!