

КОМПЬЮТЕРНОЕ ЗРЕНИЕ, ОПРЕДЕЛЕНИЕ ИЗМЕНЕНИЙ ПОСРЕДСТВОМ КОМПЬЮТЕРНОГО ЗРЕНИЯ (НА ПРИМЕРЕ СОЗДАНИЯ ЛАЗЕРНОГО ТИРА)

Кофанов П.И., Тупикин Д.А., Звягина Е.А.

*Ливенский филиал Орловского государственного университета
имени И.С. Тургенева, г. Ливны*

Ключевые слова: компьютерное зрение, система контроля и измерения, управляющая система.

Аннотация. В данной статье рассматривается скрипт по созданию лазерного тира, который имеет универсальный алгоритм и может использоваться в качестве мониторинга объектов.

XXI век – это век создания нейронных сетей, искусственного интеллекта, различных алгоритмов вычисления в зависимости от ситуации. Все отрасли производства, внедряют в свои производственные мощности автоматизацию на основе алгоритмов и расчетов. Одна из концепций активно привлекает к себе интерес со стороны программистов, математиков является создание компьютерного зрения для анализа ситуации, сравнении и принятии решения в реальном времени.

Данная концепция лазерного тира использует в себе бесконечный цикл, в котором заключены следующие действия: мониторинг, обнаружение, сопоставление, действие. Бесконечный цикл наблюдает через веб-камеру за изменением окружающей среды в поле действия камеры. В данном коде скрипта используется обнаружение объекта точки необходимого радиуса пределах видимости камеры. Обнаруженные изменения сопоставляются с исходными данными, в данном случае - с мишенью. На основе сопоставления выполняется расчет: попала ли обнаруженная точка в границы мишени, указанных в исходных данных скрипта. Если да, то запускается действие «звук выстрела».

Для создания данного проекта, использовался язык программирования Python. Python – это высокоуровневый язык программирования общего назначения. Данный язык активно развивается и поддерживается разработчиками. Данный язык имеет множество разнообразных модулей и библиотек, которые использовались при создании данного проекта.[2]

При создании проекта «лазерного тира» основной составляющей стала библиотека компьютерного зрения OpenCV. OpenCV имеет открытый исходный код, распространяется в условиях лицензии BSD.[1] Для начала работы с данной библиотекой, необходимо импортировать ее в проект и запустить видеозахват. Для корректной работы библиотеки требуется указать цели камеры и размеры получаемого изображения с видеозахвата. Код на языке Python представлен на рисунке 1.

```
#!/usr/bin/python

import sys # для работы системы
import cv2 # для компьютерного зрения
import math # для мат расчетов
import subprocess

if __name__ == '__main__':

    #цели камеры
    CenterX = 420.5
    CenterY = 190.5
    Radius = 100.0

    width = 480
    height = 640
    capture = cv2.VideoCapture(1) #Выполняем видеозахват. Где 1 - внешняя камера.
    capture.set(3, width);
    capture.set(4, height);
```

Рис. 1. Импорт библиотеки CV и начало работы с ней

Для работы компьютерного зрения, требуется конвертировать изображения в черно-белый формат высокой контрастности. Это позволяет на основе контуров распознавать требуемые объекты. Затем необходимо убрать контуры, которые не доведены до конца (в которых существуют промежутки), чтобы скрипт мог понимать реальное местоположение требуемого объекта. Для удаления промежутков между белыми пикселями изображения применяем операцию «закрытия». Работа всех описанных операций с комментариями приведена на рисунке 2.

```
if cv2.waitKey(1) >= 0:
    break
ret, frame = capture.read()
grey_image = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # делаем изображение серый цвет

ret, grey_image = cv2.threshold(grey_image, 245, 255, cv2.THRESH_BINARY)

grey_image = cv2.erode(grey_image, None, iterations = 1)
grey_image = cv2.dilate(grey_image, None, iterations = 1)
#Находим контуры
(_, contour, _) = cv2.findContours(grey_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

Рис. 2. Работа с изображением

После того, как требуемая цель найдена в нашем изображении, воспроизводится действие «звук выстрела», который берется из папки с запускаемым скриптом. В тот же момент выводиться на экран картинка (в которую требуется попасть), которая сопоставляется с окружностью изображения и производятся основные расчеты при выстреле. В папку со скриптом добавляются данные о результатах вычислений. Цикл повторяется автоматически, пока не завершить работу скрипта. По окончании работы скрипта, убираются все окна. Фрагмент кода работы вычислений и действия при обнаружении показан на рисунке 3.

```

if contour:
    subprocess.Popen(r'C:\Program Files (x86)\Windows Media Player\wmplayer.exe Shot.wav', shell = True) #воиспроизводим звук выстрела
    cnt = sorted(contour, key = cv2.contourArea, reverse = True)[0]
    (x,y), radius = cv2.minEnclosingCircle(cnt) #находим окружность объекта
    center = (x, y) #вычисляем центр
    shot_x = (float(x) - CenterX)/Radius
    shot_y = (float(y) - CenterY)/Radius
    dist = math.sqrt(shot_x*shot_x+shot_y*shot_y)
    shot_x = target_x + shot_x*target_Radius
    shot_y = target_y + shot_y*target_Radius
    Shot = (int(shot_x), int(shot_y)) #конечный расчет выстрела
    print('Coords: {}'.format(Shot)) #показываем координаты выстрела
    cv2.circle(target, Shot, 5, (60,60,255),10)
    cv2.circle(target, Shot, 10, (120,120,120),1)
    cv2.imshow("Result", target) #показываем результат на картинке
    #calibrate
    print (center, dist)
    print ("Shots", ShotCount+1)
    if dist < 1.0:
        Scoore += 1 - dist
        ShotCount += 1
    if ShotCount > 0:
        ShotCount = 0;
        Scoore = Scoore/7.0*100.0
        print("You Scoore: ", Scoore)
        Scoore = 0
        target = image.copy()
        cv2.waitKey(300)
        subprocess.Popen('vlc 924.wav', shell = True)
        cv2.waitKey(1000)
        cv2.waitKey(50)

cv2.destroyAllWindows() #убираем все окна при выходе

```

Рис. 3. Работа с стрельбой и выводом очков

Данную концепцию кода легко модернизировать, меняя лишь показатели обнаружения, сопоставления и действия. Основной алгоритм просчета универсален. На пример, данный скрипт возможно применить в производственных условиях в машиностроении для контроля качества изделий на конвейерной ленте сборки после незначительной модернизации.

Список литературы / References

1. OpenCV // URL: <https://ru.wikipedia.org/wiki/OpenCV>
2. Python // URL: <https://ru.wikipedia.org/wiki/Python>

Сведения об авторах:

Кофанов Павел Игоревич – преподаватель, Ливенский филиал ОГУ им. И.С.Тургенева, г. Ливны;

Тутикин Дмитрий Александрович – к.т.н., доцент Ливенский филиал ОГУ им. И.С. Тургенева, г. Ливны;

Звягина Елена Александровна – к.т.н., доцент Ливенский филиал ОГУ им. И.С.Тургенева, г. Ливны.

COMPUTER VISION, CHANGE DETECTION VIA COMPUTER VISION (FOR EXAMPLE, CREATE A LASER SHOOTING GALLERY)

Kofanov P.I., Tupikin D.A., Zvyagina E.A.

Keywords: computer vision, control and measurement system, control system.

Abstract. This article discusses the script to create a laser shooting gallery, which has a universal algorithm and can be used as a monitoring of objects.