УДК 621.865,8:004.4

Использование компьютерного зрения при программировании промышленного робота

А. А. Меновщикова, К. В. Аверков, Д. С. Макашин
Омский государственный университет путей сообщения
Омск, Россия

Аннотация.

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

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

При создании проекта основной составляющей стала библиотека компьютерного зрения OpenCV. OpenCV имеет открытый исходный код, распространяется в условиях лицензии BSD.

*Ключевые слова***:** *Компьютерное зрение, робот, OpenCV, Pyton, координаты, видеозахват.*

Одной из наиболее важных и сложных задач при программировании промышленного робота является планирование движения и определение координат объекта манипулирования. Чаще всего для определения координат во время настройки приходится подводить схват робота к объекту манипулирования и таким образом определять его координаты. Далее необходимо переместить объект в требуемую позицию и снова зафиксировать координаты конечной точки. Затем необходимо указать эти координаты в программе.

Такой подход имеет недостатки:

1. высокая трудоемкость (если траектория движения робота сложная, то приходится определять координаты множества точек);
2. накопленная погрешность (погрешности перемещения робота с каждой новой точкой будут увеличиваться);

3. сложность переналадки (если возникает необходимость изменения траектории, то работу нужно делать заново).

Устранить эти недостатки можно за счет использования технического зрения.

Для работы с библиотекой OpenCV, необходимо импортировать ее в проект и запустить видеозахват. Для корректной работы библиотеки требуется указать цели камеры и размеры получаемого изображения с видеозахвата.

Данную концепцию кода легко модернизировать, меняя лишь показатели обнаружения, сопоставления и действия. Основной алгоритм просчета универсален. Например, данный скрипт возможно применить в производственных условиях в машиностроении для контроля качества изделий на конвейерной ленте сборки после незначительной модернизации. Камеры передают данные на сервер, где с помощью специального ПО происходит распознавание. Системы состоят из фото- или видеокамеры и специализированного программного обеспечения, которое идентифицирует и классифицирует объекты. Чтобы научить компьютер «видеть», используются технологии машинного обучения. Собирается множество данных, которые позволяют выделить признаки и комбинации признаков для дальнейшей идентификации похожих объектов.

Ниже представлен пример программы с использованием технического зрения. Нами была использована библиотека OpenCV и язык программирования Pyton, Данная программа позволяет определять координаты объектов, находящихся в поле зрения камеры и передавать их роботу. Определить, непосредственно координаты не сложно, но можно столкнуться с определенными проблемами:

1) Камера выдает координаты в пикселях
2) Начало системы координат не совпадают

Поэтому необходимо внести следующие корректировки программы

Рис. 1– Вывод объекта на экран и его координат.

На рисунке 2 показан объект манипулирования, рабочая поверхность, робот и камера отслеживающая положение объекта.

Рассчитаем масштаб изображения:

$$\mathrm{M} = \frac{L}{\sqrt{area}}$$

$$\mathrm{M} = \frac{25}{14{,}61} = 6{,}79$$

Где:

L– длина стороны объекта;

Area– площадь куба.

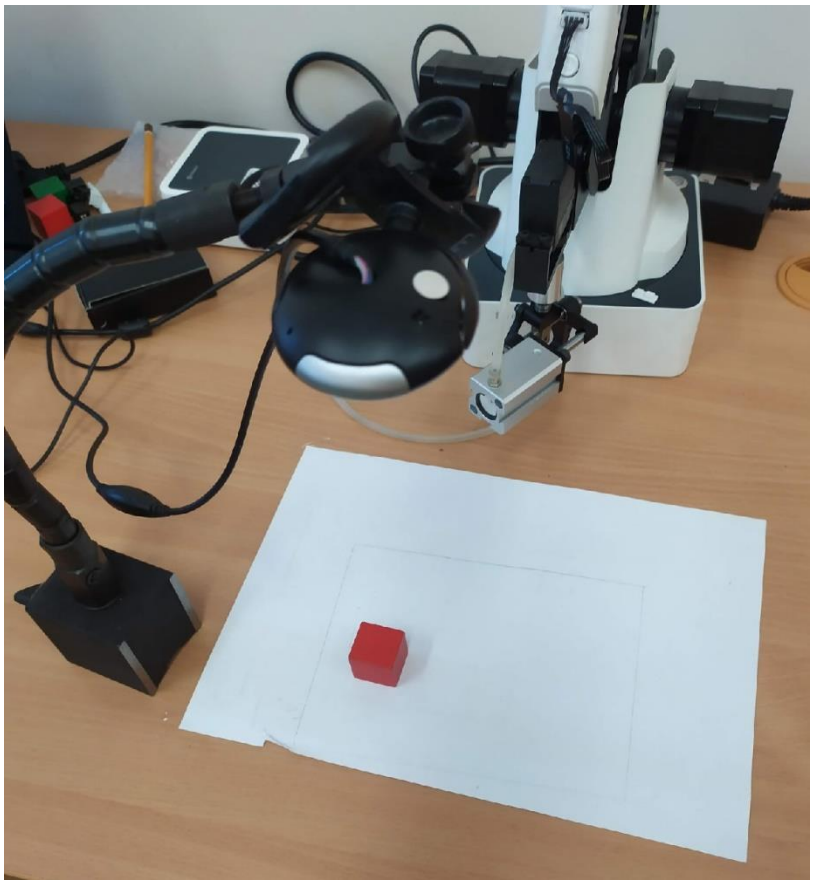

Рис. 2 – Команда настройки наведения на объект

Программа (рис.3) предназначена для определения координат объекта манипулирования.

```
import numpy as np
import cv2

capImg = cv2.VideoCapture(1)

ft = "Test_5.blockly"

while True:
    #Поиск объекта
    ret, frame = capImg.read()

    frame_hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    low_Green = np.array([0, 0, 0], dtype = "uint8")
    high_Green = np.array([60, 255, 255], dtype = "uint8")

    green_mask = cv2.inRange(frame, low_Green, high_Green)



    # Определение координат
    moments = cv2.moments(green_mask, 1)
    x_moment = moments["m01"]
    y_moment = moments["m10"]
    area = moments["m00"]
    x = int(x_moment / area)
    y = int(y_moment / area)
    x_r = (x/3.52)+232.1
    y_r = (y/3.52)-59.4

    with open(ft, "r") as f:
        old_data = f.read()
    new_data_1 = old_data.replace("111", str(x_r))
    with open(ft, "w") as f:
        f.write(new_data_1)

    with open(ft, "r") as f:
        old_data = f.read()
    new_data_2 = old_data.replace("222", str(y_r))
    with open(ft, "w") as f:
        f.write(new_data_2)

    cv2.putText(green_mask, "Object", (x,y),cv2.FONT_HERSHEY_SIMPLEX, 1, (255,0,0), 2)
    cv2.putText(green_mask, "%d, %d" % (x,y), (x,y+30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,0,0),

    cv2.imshow("Video_mask", green_mask)

    key_press = cv2.waitKey(30)
    if key_press == ord("q"):
        break

print(area)


capImg.release()
cv2.destroyAllWindows()
```

Рис. 3 – Программа по наладке робота

Система координат камеры (Хк;Yк) и система координат робота (Xp;Yp) данные системы отличаются тем, что система координат робота выдает значения в миллиметрах, а система координат камеры в пикселях, поэтому переведем пиксели в миллиметры и прибавим Δx, Δy соответсвенно.Так как направление осей совпадает, а начало нет, поэтому нужно вносить корректировки в программу(рис.3).

Пересчитаем координаты тела из пикселей в мм:

225 px= 59,4 мм;

329 px=86,86 мм;

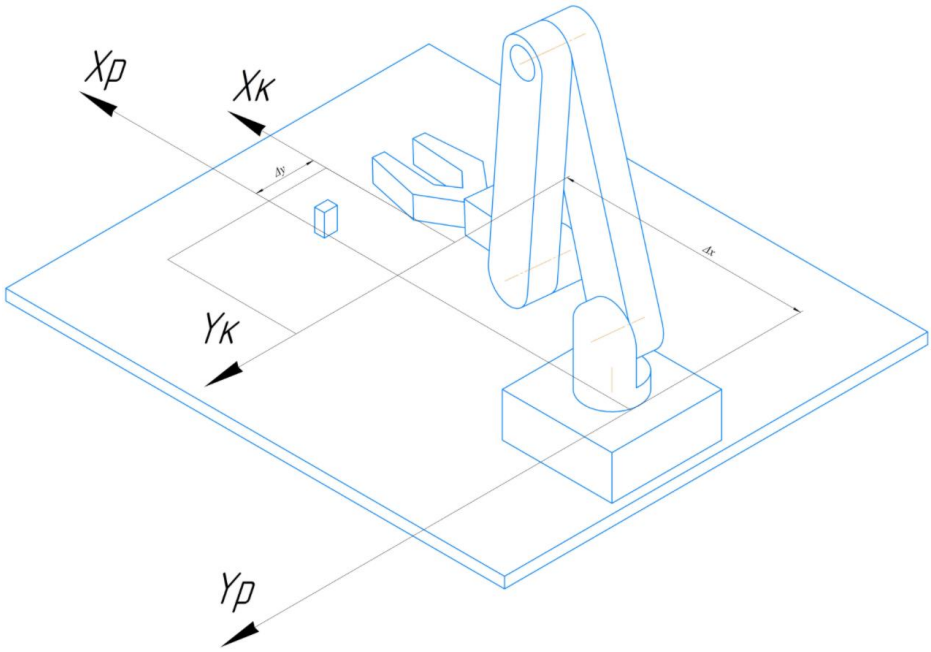

Рис. 4 – Чертеж расположения робота и объекта

Данная программа позволяет обеспечивать высокую производительность и точность работы. Многое зависит от качества камеры и плотности цвета объекта, все это влияет на восприятие робота и на точность его действий, следовательно, улучшая качество техники и объектов мы увеличиваем точность действий робота.

Библиографический список

1. Емельянов, А. А. Применение методов компьютерного зрения для управления 3D- принтерами / А. А. Емельянов, О. И. Завадская // Экосистема цифровой экономики : сборник статей / МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ; ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ». – Санкт-Петербург : Санкт-Петербургский государственный экономический университет, 2021. – С. 47-51. – EDN KSOVCK.
2. Кофанов, П. И. Компьютерное зрение, определение изменений посредством компьютерного зрения (на примере создания лазерного тира) / П. И. Кофанов, Д. А. Тупикин, Е. А. Звягина // Мехатроника, автоматика и робототехника. – 2019. – № 3. – С. 158-160. – DOI 10.26160/2541-8637-2019-3-158-160. – EDN YZFITB.

Для цитирования: *Меновщикова А.А., Аверков К.В., Макашин Д.С. Использование компьютерного зрения при программировании промышленного робота // Омский государственный университет путей сообщения Омск, Россия: матер. одиннадцатой междунар. науч.-техн. конф. (26–28 апреля 2022 года, Омск, Россия). Омск: ОмГТУ, 2021. С. 00–00.*

Сведения об авторах

Аверков Константин Васильевич

Научные интересы:
Email: averok@yandex.ru

Spin-код: 8407-0488
ORCID: https://orcid.org/0000-0002-3642-5292
Publons

Меновщикова Ангелина Александровна
Научные интересы:
Email: angelina.menovschicova@gmail.com
Spin-код:
ORCID:

Publons

Макашин Дмитрий Сергеевич
Научные интересы:

Email: dima.makashin@gmail.com
Spin-код 1763-1883
ORCID: https://orcid.org/0000-0002-8297-5551
Publons