ИНФОРМАТИКА

УДК 004.93:004.77 DOI: https://doi.org/10.54341/20778708_2025_1_62_108
EDN: IRYMTE

РЕАЛИЗАЦИЯ СИСТЕМЫ КОМПЬЮТЕРНОГО ЗРЕНИЯ ДЛЯ ДЕТЕКТОРОВ УГЛОВ

Н.А. Аксенова

Гомельский государственный университет имени Франциска Скорины

IMPLEMENTATION OF A COMPUTER VISION SYSTEM FOR ANGLE DETECTORS

N.A. Aksionova

Francisk Skorina Gomel State University

Аннотация. Описывается разработка системы машинного зрения, позволяющей определять особые точки углов на статических изображениях архитектурных планов. Основной задачей данной разработки является создание библиотеки на языке программирования Python, содержащей детекторы углов Харриса, Ши – Томази и FAST. Разработанные программные модули являются дополнением для Blender и позволяют ускорить процесс трехмерного моделирования на основе распознанных угловых точек.

Ключевые слова: *детекторы углов*, *особые точки*, *компьютерное зрение.*

Для цитирования: *Аксенова*, *Н.А*. Реализация системы компьютерного зрения для детекторов углов / Н.А. Аксенова // Проблемы физики, математики и техники. – 2025. – № 1 (62). – С. 108–112. – DOI: https://doi.org/10.54341/20778708_2025_1_62_108. – EDN: IRYMTE

Abstract. The development of a machine vision system that allows you to determine the special points of angles on static images of architectural plans are described. The main objective of this development is to create a library in the Python programming language containing the Harris, Shi – Tomasi and FAST angle detectors. The developed software modules are an addition to Blender and allow you to speed up the process of three-dimensional modeling based on recognized corner points.

Keywords: *angle detectors*, *singular points*, *computer vision.*

For citation: *Aksionova*, *N.A*. Implementation of a computer vision system for angle detectors / N.A. Aksionova // Problems of Physics, Mathematics and Technics. – 2025. – № 1 (62). – P. 108–112. – DOI: https://doi.org/10.54341/20778708_2025_1_62_108 (in Russian). – EDN: IRYMTE

Введение

В современном мире программные системы машинного зрения широко применяются в различных областях, включая компьютерное зрение, робототехнику, беспилотные автомобили и многое другое. Эти системы способны анализировать и интерпретировать изображения и видео, что позволяет компьютерам обрабатывать информацию, подобно тому, как это делает человеческий глаз. Важным аспектом программных систем машинного зрения является способность обнаруживать и извлекать особенности изображений, такие как грани, углы и особые точки интереса [1].

В данной работе описывается разработка библиотеки детекторов углов для программной системы машинного зрения. Углы являются важными структурными элементами визуальных данных и нахождение их положения и характеристик имеет большое значение для анализа изображений и определения геометрических свойств объектов [2]. Детектирование углов в программных системах машинного зрения позволяет обнаруживать пересечения линий, углы поворота объектов и другие ключевые особенности, что может быть важным в таких задачах как навигация, распознавание объектов, отслеживание движущихся объектов и многое другое [3].

Целью является разработка эффективной библиотеки детекторов углов, которая позволит исследователям и разработчикам использовать их в своих приложениях для программных систем машинного зрения. Библиотека должна предоставлять набор алгоритмов и методов для обнаружения и извлечения углов на изображениях с высокой точностью и скоростью. При этом важно учитывать различные условия освещения, шумы, изменения масштаба и повороты объектов.

1 Структура проекта

Библиотека представляет собой три функции, каждая из которых реализует один из методов по нахождению ключевых точек: Harris [4], Shi – Tomasi и FAST. Функции принимают на входе путь к изображению и параметры, регулируемые пользователем. Из таких параметров

можно выделить количество точек для поиска, числовое значение порога для определения пикселя ключевой точкой, количество пикселей, среди которых находится локальная ключевая точка.

Функционал:

– поддержка всех современных форматов графических файлов, такие как: PNG, JPEG, BMP, TIFF, GIF и SVG;

– подсчёт количества распознанных углов для каждого метода;

– оценка скорости распознавания углов.

Для реализации системы компьютерного зрения для детекторов углов было разработано приложение на языке программирование Python с применением импортированных библиотек OpenCV, Numpy, PIL, а также встроенных библиотек Math и Time. Данное приложение позволяет найти ключевые точки на выбранном пользователем изображении и отрегулировать некоторые параметры при их поиске. Приложение представляет собой функцию по вызову выбранного метода на выбранное изображение, а также отчет для выбранного изображения в виде таблицы. В таблице отображаются несколько показателей, характеризующих работу алгоритмов по нахождению особых точек для каждого метода.

Приложение состоит из двух исходных файлов: первый – сама библиотека, второй – интерфейс приложения. Также, имеется изображение, которое может применяться для тестирования программы.

2 Описание основных модулей

Метод Харриса. Функция по нахождению ключевых точек методом Харриса принимает несколько параметров: путь до изображения, размер блока окрестности, а также числовое значение порога для определения пикселя ключевой точкой. Порог принимает значения от нуля включительно до единицы не включительно. Чем меньше значение порога, тем больше углов будет найдено, при вводе нуля это значение будет максимальным.

Функция начинается с чтения изображения и преобразования его в черно-белую шкалу с помощью *cv2*; после этого оно преобразуется в изображение типа *float32*:

```
def harris_corner(image, threshold,
neighborhood):
 start_time=time.time()
 img=cv2.cvtColor(img,
     cv2.COLOR_BGR2GRAY)
 gray=np.float(gray)
```

Также в самом начале вызывается функция *time.time*() и записывается ее текущее значение, чтобы получить время выполнения кода данной функции. Эта строка будет во всех последующих функциях.

Далее вызывается функция *cv2.cornerHarris*. Новое индивидуальное значение порога равно *threshold·dst.max*(), согласно нему находятся координаты углов и их количество:

```
dst=cv2.cornerHarris(gray.Neighborh
ood,3,0.04)
corner_map=dst[dst> thresh-
old*dst.max()]
number_of_points=
 np.count_nonzero(corner_map)
```

Функция *cv2.cornerHarris* принимает несколько параметров:

img – входное черно-белое изображение типа float32;

blockSize – размер окрестности, учитываемой при обнаружении углов;

ksize – используемый параметр апертуры производной Собеля;

k – свободный параметр детектора Харриса.

```
img[dst> threshold*dst.max()]=
 [255,0,0]
runtime=time.time()-start_time
return img, number_of_points,
 str(neighborhood), threshold,
 round(runtime,2)
```

Пиксели, определяемые как углы по алгоритму, помечаются красным цветом.

Далее записывается время работы программы, которое равно разности текущего значения функции *time.time*() и ранее записанного ее значения.

Функция возвращает:

– изображение с обозначенными углами;

– размер блока окрестности пикселя, в котором производится сравнение по контрастности;

– порог для определения пикселя как ключевой точки;

– округленное до двух знаков после запятой время работы программы в секундах.

Детектор Харриса инвариантен к поворотам, частично инвариантен к аффинным изменениям интенсивности. К недостаткам можно отнести чувствительность к шуму и зависимость детектора от масштаба изображения. Для устранения этого недостатка используют многомасштабный детектор Харриса: multi-scale Harris detector.

Метод Ши – Томази. Функция по нахождению ключевых точек методом Shi – Tomasi принимает несколько параметров: путь до изображения, желаемое количество точек для поиска, а также числовое значение порога для определения пикселя ключевой точкой.

Порог принимает значения от нуля до единицы не включительно. Чем меньше значение порога, тем больше углов будет найдено.

Функция начинается с чтения изображения и преобразование его в черно-белую шкалу с помощью *cv2*:

```
def shi_tomasi_corner(image,
 threshold, number_of_points):
 start_time=time.time()
 img=cv2.imread(image)
 gray= cv2.cvtColor(img,
       cv2.COLOR_BGR2GRAY)
```

После этого получается массив найденных углов также с помощью функции *cv2.goodFeaturesToTrack*. Она принимает на вход несколько параметров:

image – данное изображение в черно-белом формате;

corners – максимальное количество углов для возврата. Если углов больше, чем найдено, возвращается самые контрастные из них;

qualityLevel – параметр, характеризующий минимально допустимое качество углов изображения;

minDistance – минимально возможное расстояние между возвращенными углами в пикселях;

mask – необязательный параметр для заполнения. Если изображение не пустое, он указывает область, в которой обнаруживаются углы;

blockSize – размер блока для вычисления производной ковариационной матрицы по окружению каждого пикселя;

HarrisDetector – устанавливает использование метода Ши – Томази;

k – свободный параметр детектора Harris.

Далее с помощью цикла *for* на изображении рисуются точки в соответствии с полученными координатами ключевых точек.

```
number_of_points=len(corners)
runtime= time.time()-start_time
return img, number_of_points,
 str(neighborhood), threshold,
 round(runtime,2)
```

Функция возвращает изображение с обозначенными углами; длину стороны окружения пикселя, в котором производится сравнение по контрастности, которая является константой для этого метода и равна 3; порог, заданный пользователем ранее; округленное до двух знаков после запятой время работы программы в секундах.

Метод FAST. Функция по нахождению ключевых точек методом FAST принимает на вход только путь до изображения.

Функция начинается с чтения изображения с помощью *cv2* и создания локального детектора углов:

```
def FAST_corner(image):
 start_time=time.time()
 img=cv2.imread(imgage)
 fast=
  cv.FastFeatureDetector_create()
```

С помощью созданного детектора находятся ключевые точки. Далее записываются параметры для возврата. Число углов равно длине кортежа ключевых точек, значение порога можно получить функцией *getThreshold*(), а значение окрестности для пикселя – функцией *getType*():

```
Key_point=fast.detect(img, None)
Threshold=fastThreshold()
Neighborhood=fast.getType()
Number_of_points=len(key_points)
```

После этого на изображении помечаются найденные ключевые точки с помощью функции *cv2.drawKeyPoints*, которая принимает входное изображение, кортеж ключевых точек, выходное изображение и цвет для ключевых точек в формате RGB:

```
img=cv2.drawKeypoints(img,
    key_points,None,(250,0,0))
runtime= time.time()-start_time
return img, number_of_points,
 str(neighborhood),
 threshold, round(runtime,2)
```

Функция возвращает:

– изображение с обозначенными углами;

– значение окрестности пикселя, в котором производится сравнение по контрастности;

– порог для определения пикселя как ключевой точки;

– округленное до двух знаков после запятой время работы программы в секундах.

3 Тестирование алгоритмов

Для тестирования алгоритмов было выбрано два графических изображения: первое изображение с шумами и невысоким контрастом (рисунок 3.1, *а*) и второе изображение чёткое с высоким контрастом (рисунок 3.1, *б*).

Результаты поиска углов методом Ши – Томази представлены на рисунке 3.2.

Выходные данные по методу Ши – Томази для первого изображения:

– время выполнения – 0,01 сек;

– найденные ключевые точки – 204;

– предел – 0,01;

– окружение – 3×3 пикселей.

Выходные данные по методу Ши – Томази для второго изображения:

– время выполнения – 0,01 сек;

– найденные ключевые точки – 311;

– предел – 0,01;

– окружение – 3×3 пикселей.

Результаты поиска углов методом Харриса представлены на рисунке 3.3.

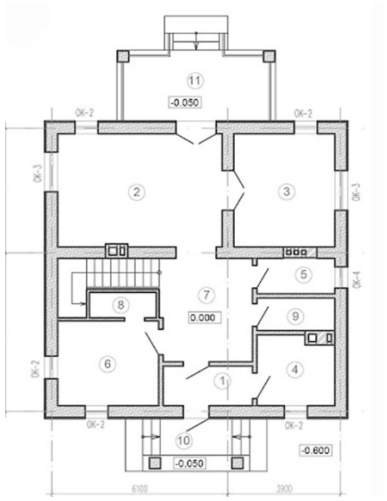
Выходные данные по методу Харриса для первого изображения:

– время выполнения – 0,03 сек;

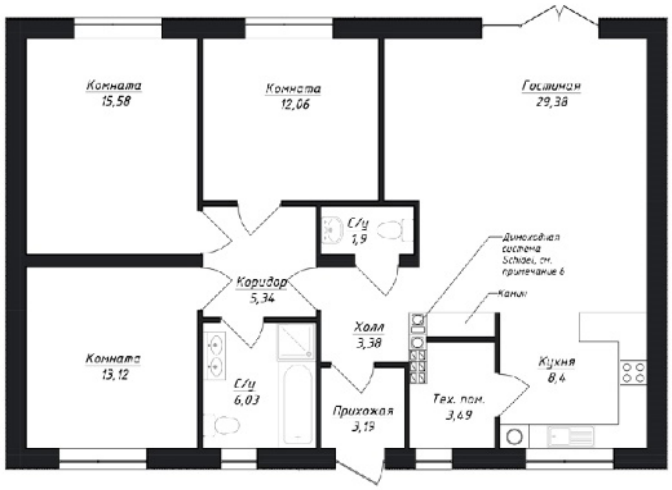
– найденные ключевые точки – 136;

– предел – 0,01;

– окружение – 4×4 пикселей.



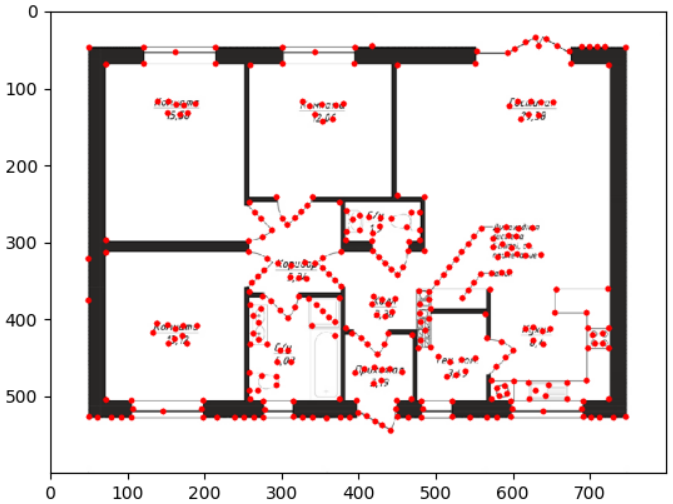
а) графическое изображение невысокого контраста с шумами;



б) графическое изображение высокого контраста без шумов

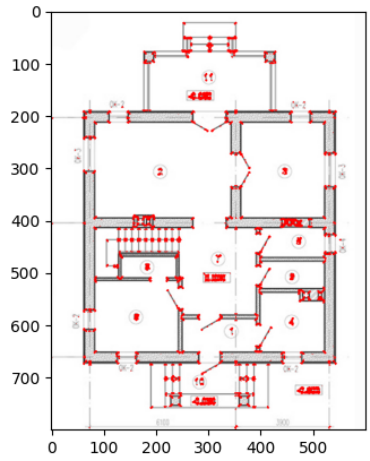
Рисунок 3.1 – Входные изображения для апробации алгоритмов

а) графическое изображение невысокого контраста с шумами

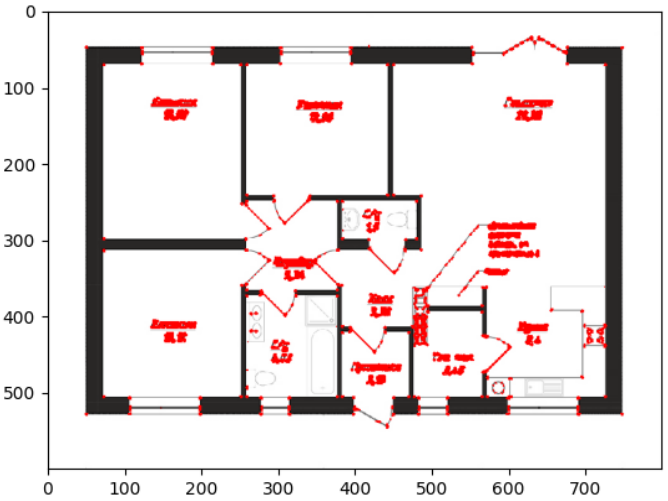


б) графическое изображение высокого контраста без шумов

Рисунок 3.2 – Результат поиска углов методом Ши – Томази



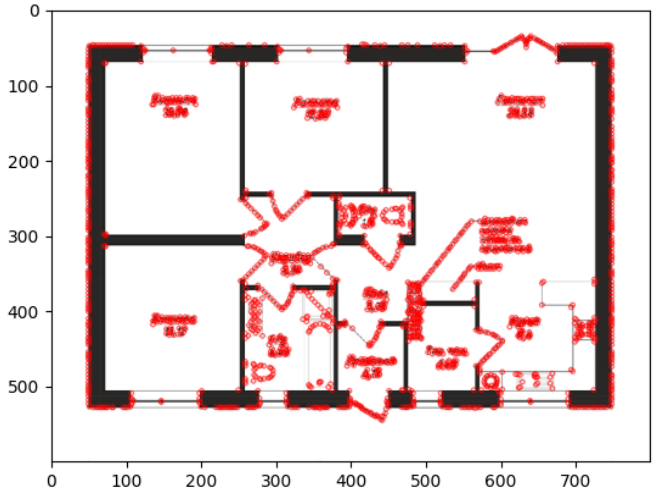
а) графическое изображение невысокого контраста с шумами



б) графическое изображение высокого контраста без шумов

Рисунок 3.3 – Результат поиска углов методом Харриса

а) графическое изображение невысокого контраста с шумами



б) графическое изображение высокого контраста без шумов

Рисунок 3.4 – Результат поиска углов методом FAST

Выходные данные по методу Харриса для второго изображения:

– время выполнения – 0,02 сек;
– найденные ключевые точки – 162;
– предел – 0,01;
– окружение – 4×4 пикселей.

Результаты поиска углов методом FAST представлены на рисунке 3.4.

Выходные данные по методу FAST для первого изображения:

– время выполнения – 0,03 сек;
– найденные ключевые точки – 483;
– предел – 10,0;
– окружение – 2×2 пикселей.

Выходные данные по методу FAST для второго изображения:

– время выполнения – 0,02 сек;
– найденные ключевые точки – 378;
– предел – 10,0;
– окружение – 2×2 пикселей.

Заключение

В результате проведенных исследований разработана система компьютерного зрения для детекторов углов. Система состоит из двух исходных файлов: библиотеки и интерфейса приложения. В процессе разработки библиотеки было уделено особое внимание оптимизации алгоритмов с целью достижения высокой производительности и эффективного использования вычислительных ресурсов. Реализация была проведена с использованием современных программных инструментов и языков программирования, обеспечивая гибкость и легкость внедрения библиотеки в различные программные системы машинного зрения.

Полученная библиотека является ценным инструментом для исследователей и разработчиков в области программных систем машинного зрения. Она позволяет обнаруживать и извлекать углы изображений с высокой точностью и быстродействием, что может быть полезным для широкого спектра задач, включая навигацию, распознавание объектов, трекинг движущихся объектов и многое другое.

ЛИТЕРАТУРА

1. *Aksionova*, *N.A.* Method of construction of three-dimensional structures based on key corner points / N.A. Aksionova, D.S. Sych, A.V. Varuyeu // Proceedings of Francisk Scorina Gomel State University. – 2023. – № 6 (141). – P. 69–75.

2. *Аксёнова*, *Н.А.* Разработка SDK для мобильного приложения с применением технологии дополненной реальности / Н.А. Аксенова, А.И. Кучеров // Известия Гомельского государственного университета имени Ф. Скорины. – 2021. – № 126 (3). – С. 81–84.

3. *Demidenko*, *O.M.* 3D-modeling of Augmented Reality objects using Shi – Tomasi corner detection algorithms. / O.M. Demidenko, N.A. Aksionova, A.V. Varuyeu // J. Phys. CS. 2091. – 2021. – P. 012058. – DOI: https://doi.org/10.1088/1742-6596/2091/1/012058.

4. *Harris*, *C.* A combinet corner and edge detector / C. Harris, M. Stephens // Proceedings of 4th Alvey Vision Conference. – 1988. – P. 147–151.

Поступила в редакцию 27.01.2025.

Информация об авторах

Аксенова Наталья Андреевна – аспирантка