**АЛГОРИТМ РАСПОЗНАВАНИЯ ЦВЕТА**

Щепеткин С.А.

Екатеринбург

2018

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc514775608)

[ПОСТАНОВКА ЗАДАЧИ 4](#_Toc514775609)

[Цель работы 4](#_Toc514775610)

[Задачи работы 4](#_Toc514775611)

[Информационные потоки алгоритма 4](#_Toc514775612)

[ОПИСАНИЕ РАЗРАБОТАННОГО АЛГОРИТМА 5](#_Toc514775613)

[Преобразование кадра в формат HSV 5](#_Toc514775614)

[Фильтрация в заданном диапазоне HSV 10](#_Toc514775615)

[Морфологическое преобразование 10](#_Toc514775616)

[Размывание 12](#_Toc514775617)

[Детектирование 12](#_Toc514775618)

[АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 13](#_Toc514775619)

[ЗАКЛЮЧЕНИЕ 16](#_Toc514775620)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc514775621)

[ПРИЛОЖЕНИЕ А – ЛИСТИНГ ПРОГРАММЫ 18](#_Toc514775622)

# ВВЕДЕНИЕ

В настоящее время робототехника и искусственный интеллект развиваются быстрыми темпами. Уже сейчас нейронные сети и адаптивные алгоритмы используются во многих областях науки и жизни. В робототехнике одной из важнейших задач, является задача реализации машинного зрения - научного направления в области искусственного интеллекта, в частности робототехники, и связанные с ним технологии получения изображений объектов реального мира, их обработки и использования полученных данных для решения разного рода прикладных задач без участия (полного или частичного) человека[1].

Машинное зрение – довольно обширное направление, которое в свою очередь делится на более конкретные задачи, такие как:

1. Распознавание
2. Идентификация
3. Обнаружение
4. Распознавание текста
5. Восстановление 3D формы по 2D изображениям
6. Оценка движения
7. Восстановление сцены
8. Восстановление изображений
9. Выделение на изображениях структур определенного вида, сегментация изображений
10. Анализ оптического потока

# ПОСТАНОВКА ЗАДАЧИ

## Цель работы

Главная цель работы состоит в том, чтобы изучить и реализовать алгоритм распознавания определенного цвета на изображениях и видео.

## Задачи работы

1) Написание алгоритма распознавания цвета на языке Python

2) Апробирование алгоритма в условиях лабораторного эксперимента, с полной подготовкой и проведением эксперимента

3) Исследование возможности применения алгоритма в смежных областях деятельности

## Информационные потоки алгоритма

**Система получает на вход**

* Исходное изображение;
* Диапазон распознаваемого цвета в формате HSV;

**Выходные данные системы**

* Результирующее изображение;

# ОПИСАНИЕ РАЗРАБОТАННОГО АЛГОРИТМА

Алгоритм работы python-скрипта для распознавания состоит из пяти основных шагов:

1. Преобразование кадра в формат HSV
2. Фильтрация в заданном диапазоне HSV
3. Морфологическое преобразование
4. Размывание
5. Детектирование

Рассмотрим поподробнее каждый шаг.

## Преобразование кадра в формат HSV

В модели RGB (от англ. red – красный, green – зелёный, blue – голубой) все цвета получаются путём смешения трёх базовых (красного, зелёного и синего) цветов в различных пропорциях. Доля каждого базового цвета в итоговом может восприниматься, как координата в соответствующем трёхмерном пространстве, поэтому данную модель часто называют цветовым кубом. На Рис. 1 представлена модель цветового куба.

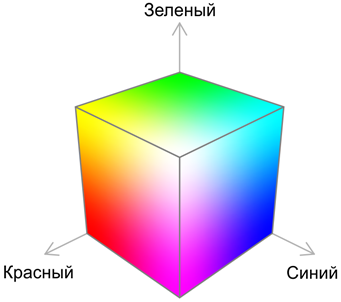


Рис. 1. Модель цветового куба

Чаще всего модель строится так, чтобы куб был единичным. Точки, соответствующие базовым цветам, расположены в вершинах куба, лежащих на осях: красный – (1;0;0), зелёный – (0;1;0), синий – (0;0;1). При этом вторичные цвета (полученные смешением двух базовых) расположены в других вершинах куба: голубой — (0;1;1), пурпурный — (1;0;1) и жёлтый – (1;1;0). Чёрный и белые цвета расположены в начале координат (0;0;0) и наиболее удалённой от начала координат точке (1;1;1). Рис. 2 показывает только вершины куба.

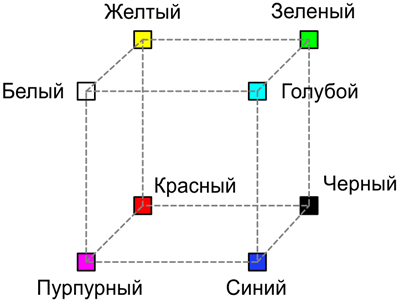


Рис. 2. Вершины модели цветового куба

В модели RGB для каждой составляющей цвета отводится определённое количество бит, например, если для кодирования каждой составляющей отводить 1 байт, то с помощью этой модели можно закодировать 2^(3\*8)≈16 млн. цветов. На практике такое кодирование избыточно, т.к. большинство людей не способно различить такое количество цветов. Часто ограничиваются т.н. режимом «High Color» в котором на кодирование каждой компоненты отводится 5 бит. В некоторых приложениях используют 16-битный режим, в котором на кодирование R и B составляющих отводится по 5 бит, а на кодирование G составляющей 6 бит. Этот режим, во-первых, учитывает более высокую чувствительность человека к зелёному цвету, а во-вторых, позволяет более эффективно использовать особенности архитектуры ЭВМ. Количество бит, отводимых на кодирование одного пиксела, называется глубиной цвета.

Цветовая модель RGB весьма проста в плане аппаратной реализации, но у нее есть один существенный недостаток. Человеку очень тяжело оперировать цветами, заданными в этой модели, т.к. человек, описывая цвета, пользуется не содержанием в описываемом цвете базовых составляющих, а несколько иными категориями.

Чаще всего люди оперируют следующими понятиями: цветовой тон, насыщенность и светлота. При этом, говоря о цветовом тоне, обычно имеют в виду именно цвет. Насыщенность показывает, насколько описываемый цвет разбавлен белым (розовый, например, это смесь красного и белого). Понятие светлоты наиболее сложно для описания, и с некоторыми допущениями под светлотой можно понимать интенсивность света.

Если рассмотреть проекцию RGB-куба в направлении диагонали белый-чёрный, то получится шестиугольник:

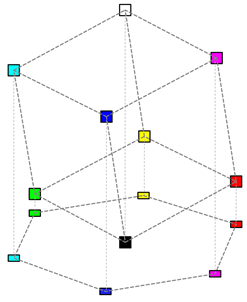


Рис. 3. Проекция модели цветового куба

Все серые цвета (лежащие на диагонали куба) при этом проецируются в центральную точку. Чтобы с помощью этой модели можно было закодировать все цвета, доступные в RGB-модели, необходимо добавить вертикальную ось светлоты (или интенсивности) (I). В итоге получается шестигранный конус:

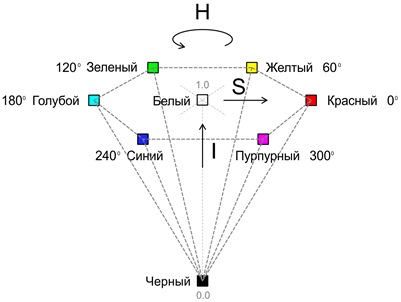


Рис. 4. Вершины конуса HSV(HSI)

При этом тон (H) задаётся углом относительно оси красного цвета, насыщенность (S) характеризует чистоту цвета (1 означает совершенно чистый цвет, а 0 соответствует оттенку серого). Важно понимать, что тон и насыщенность не определены при нулевой интенсивности.

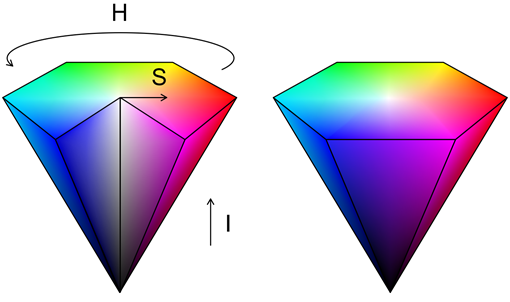
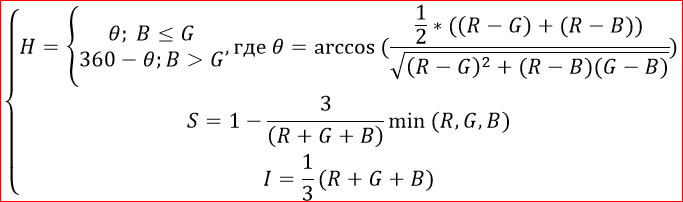


Рис. 5. Модель цветового пространства HSV(HSI)

Алгоритм перевода из RGB в HSI можно выполнить, воспользовавшись следующими формулами:

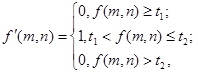


Цветовая модель HSI очень популярна среди дизайнеров и художников, т.к. в этой системе обеспечивается непосредственный контроль тона, насыщенности и яркости. Эти же свойства делают эту модель очень популярной в системах машинного зрения [2].

## Фильтрация в заданном диапазоне HSV

Основой этого шага является бинаризация изображения. Операция порогового разделения, которая в результате дает бинарное изображение, называется бинаризацией. Целью операции бинаризации является радикальное уменьшение количества информации, содержащейся на изображении. В процессе бинаризации исходное полутоновое изображение, имеющее некое количество уровней яркости, преобразуется в черно-белое изображение, пиксели которого имеют только два значения – 0 и 1.

Для выделения областей, в которых значения яркости пикселей может меняться в известном диапазоне, вводится бинаризация с двойным ограничением (t1<t2):



Так же возможны другие вариации с порогами, где пропускается только часть данных (средне полосовой фильтр) [3].

## Морфологическое преобразование

Термин морфология относится к описанию свойств формы и структуры каких-либо объектов. В контексте машинного зрения этот термин относится к описанию свойств формы областей на изображении. Операции математической морфологии изначально были определены как операции над множествами, но скоро выяснилось, что они также полезны в задачах обработки множества точек в двумерном пространстве. Множествами в математической морфологии представляются объекты на изображении. Легко заметить, то множество всех фоновых пикселов бинарного изображения является одним из вариантов его полного описания.  
В первую очередь математическая морфология используется для извлечения некоторых свойств изображения, полезных для его представления и описания. Например, контуров, остовов, выпуклых оболочек. Также интерес представляют морфологические методы, применяемые на этапах предварительной и итоговой обработки изображений. Например, морфологическая фильтрация, утолщение или утоньшение.  
Входными данными для аппарата математической морфологии являются два изображения: обрабатываемое и специальное, зависящее от вида операции и решаемой задачи. Такое специальное изображения принято называть примитивом или структурным элементом. Как правило, структурный элемент много меньше обрабатываемого изображения. Структурный элемент можно считать описание области с некоторой формой. Понятно, что форма может быть любой, главное, чтобы её можно было представить в виде бинарного изображения заданного размера. Во многих пакетах обработки изображений наиболее распространенные структурные элементы имеют специальные названия: BOX[H,W] –прямоугольник заданного размера, DISK[R] — диск заданного размера, RING[R] – кольцо заданного размера.

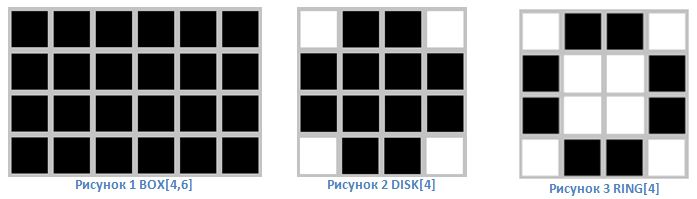


Рис. 6. Виды структурных элементов морфологического преобразования

Результат морфологической обработки зависит как от размера и конфигурации исходного изображения, так и от структурного примитива.  
Размер структурного элемента как правило равен 3\*3, 4\*4 или 5\*5 пикселов. Это обусловлено главной идеей морфологической обработки, в процессе которой отыскиваются характерные детали изображения. Искомая деталь описывается примитивом, и в результате морфологической обработки можно подчеркнуть или удалить такие детали на всём изображении.

Основными операциями математической морфологии являются наращивание, эрозия, замыкание и размыкание. В этих названиях отражена суть операций: наращивание увеличивает область изображения, а эрозия делает её меньше, операция замыкания позволяет замкнуть внутренние отверстия области и устранить заливы вдоль границы области, операция размыкания помогает избавиться от маленьких фрагментов, выступающих наружу области вблизи её границы. Далее будут представлены математические определения морфологических операций [4].

## Размывание

Фильтр размытия по Гауссу достаточно часто применяется сам по себе или как часть других алгоритмов обработки изображений. Он нужен для сглаживания шероховатостей после морфологического преобразования.

## Детектирование

На этом шаге производится нахождение контуров объекта, имеющего искомый цвет. В основе поиска контуров используется алгоритм Сузуки-Касами [5].

# АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

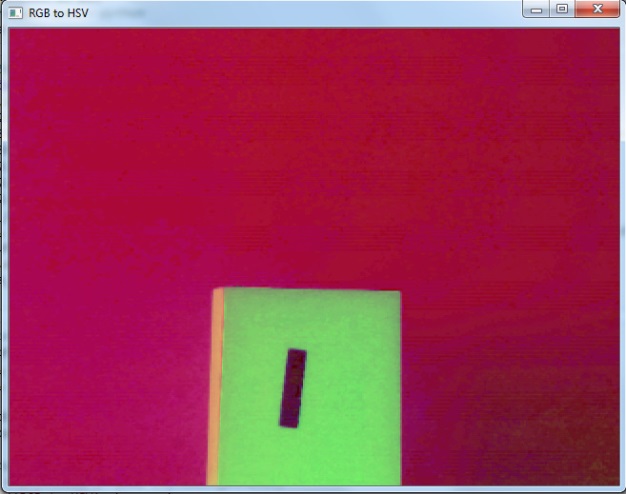
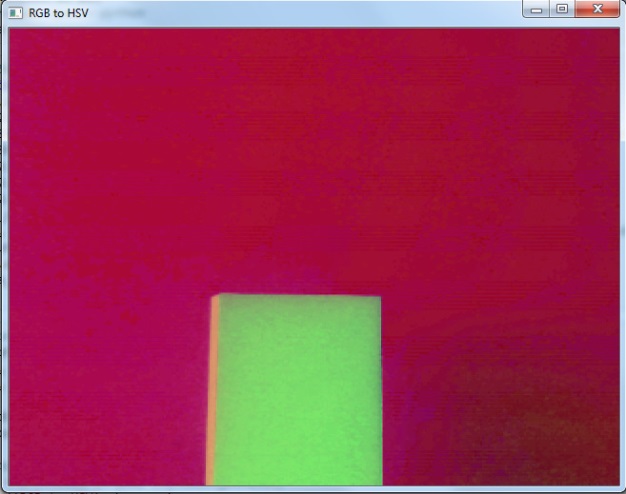


Рис. 6. Кадр в формате HSV

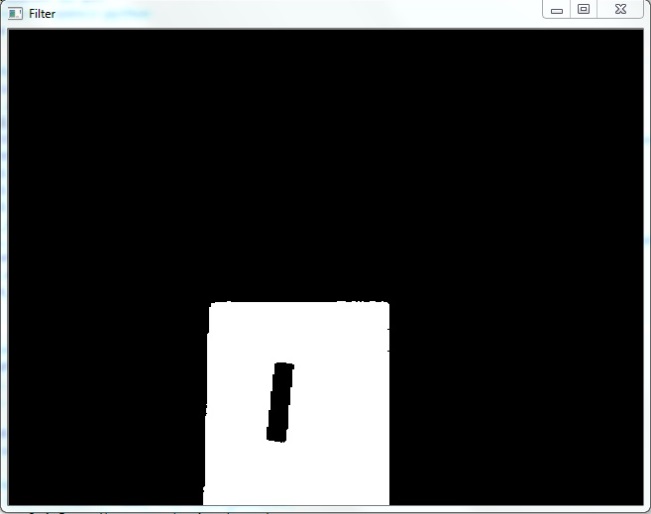


Рис. 7. Применение фильтра (бинаризация)

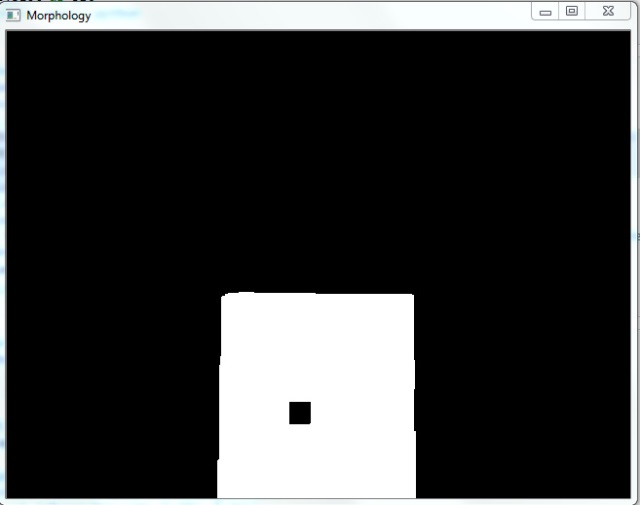
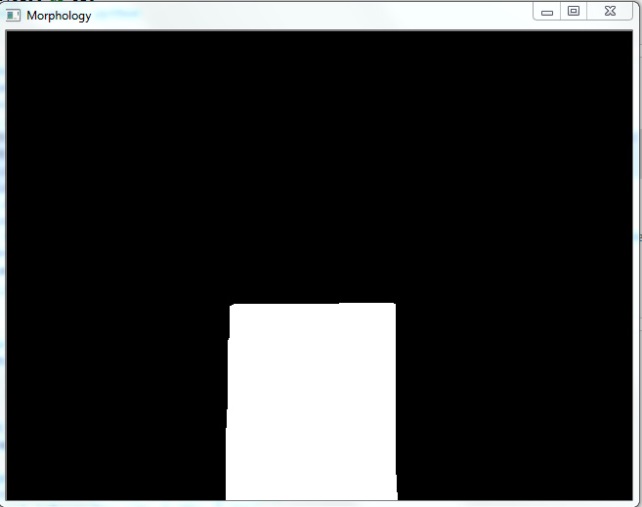


Рис. 8. Морфологическое преобразование

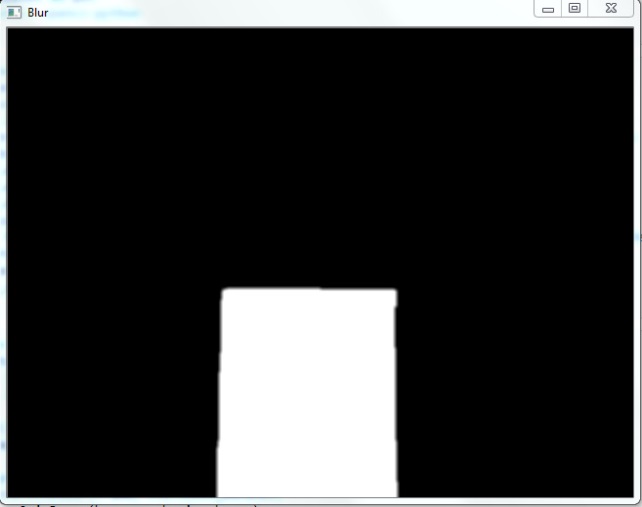
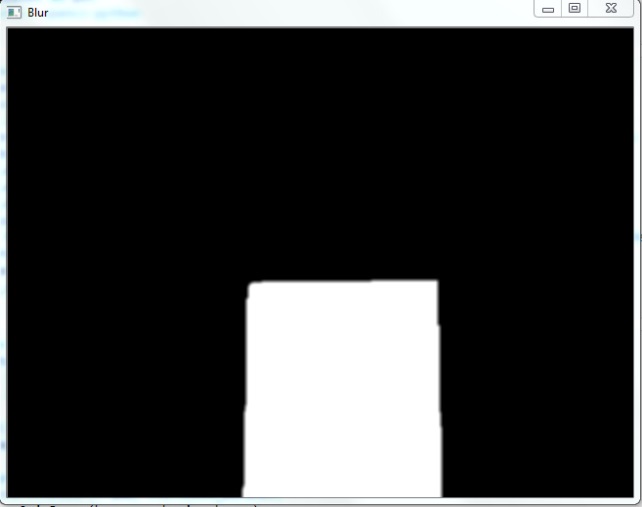


Рис. 9. Гауссовское размытие

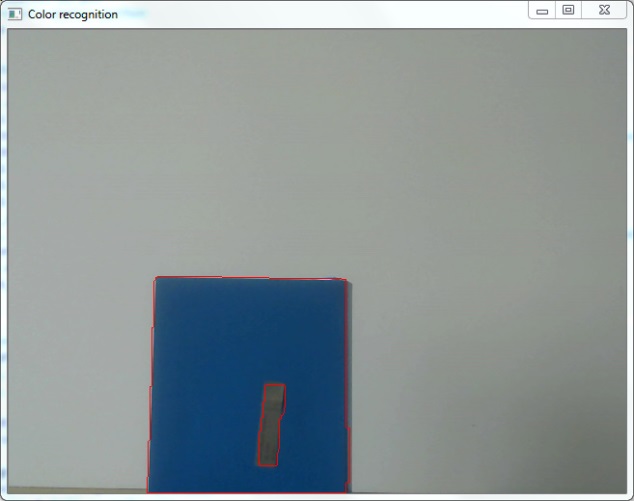
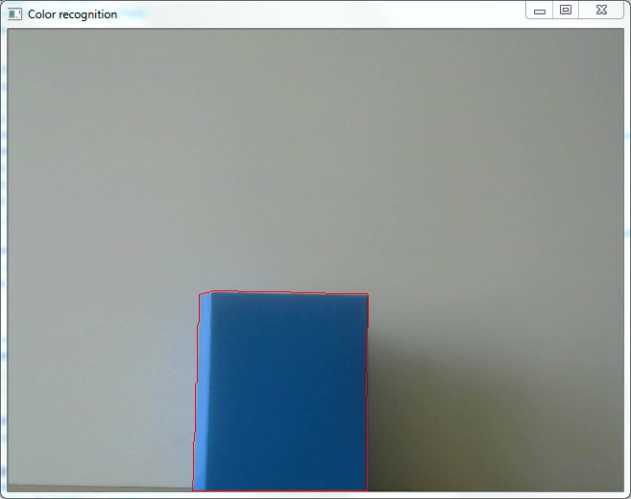


Рис. 10. Детектирование и рисование контуров объекта

Мне долго не удавалось настроить маску для выделения нужного цвета. Лишь спустя множество попыток, я пришел к выводу что камера видит этот мир совсем не так как я. Смотря на синий объект, я вижу синий объект. Когда на этот же объект смотрит камера - она видит очень яркий синий объект. Такой яркий, что он уже не синий. Дело в том, что матрица камеры очень хорошо чувствует ближнее инфракрасное излучение (Near-infrared). Эти ИК лучи в больших количествах излучаются галогеновыми лампами и солнцем.

Как показала практика, настройка цветовой маски в условиях ИК засветки является весьма непростой задачей. Решением здесь может стать ИК фильтр, который устанавливается в хороших фотоаппаратах для обеспечения адекватной цветопередачи. Но проблему можно попытаться решить и немного иначе, хоть и не так качественно как с фильтром. Для выделения засвеченного объекта можно применить сразу два фильтра. Один - для засвеченного участка, другой - для затененного.

Для каждого объекта требуется специально настраивать цветовую маску. Чтобы выделить нужный цвет необходимо подобрать границы компонента H (Hue). Параметр S (Saturation) отвечает за насыщенность цвета. Например, кожа человека имеет оранжево-желтый цвет (H), но слабую насыщенность. Наконец V (Value) определяет яркость цвета. Затененный объект будет иметь низкое значение V.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта был изучен и реализован алгоритм распознавания цвета в реальном времени. Были проведены тестовые испытания с целью выявления принципов работы данного алгоритма и областей его применения.

Результаты показали, что подобный алгоритм может применяться для решения широкого спектра задач из области машинного зрения. Алгоритм обладает своими плюсами и минусами, основным минусом является ИК засветка. Основной плюс это простота алгоритма.

Рассмотренный алгоритм является лишь малой частью существующих на данный момент алгоритмов подобного плана. Разнообразие этих алгоритмов позволяет решать огромное число задач по детектированию, распознаванию, выделению объектов на изображении и улучшению качества изображения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Машинное зрение. Что это и как им пользоваться? Обработка изображений оптического источника. Метод доступа: <https://habrahabr.ru/post/350918/>
2. О цветовых пространствах. Метод доступа: <https://habr.com/post/181580/>
3. Сегментация изображения. Метод доступа: <https://habr.com/post/128768/>
4. Математическая морфология. Метод доступа: <https://habr.com/post/113626/>
5. Suzuki–Kasami algorithm. Метод доступа: <https://en.wikipedia.org/wiki/Suzuki%E2%80%93Kasami_algorithm>
6. OpenCV-Python Tutorials. Метод доступа: <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html>

# ПРИЛОЖЕНИЕ А – ЛИСТИНГ ПРОГРАММЫ

#Загрузка библиотек

from skimage.io import imread, imsave, imshow

from skimage import img\_as\_ubyte

import numpy as np

import matplotlib.pyplot as plt

import cv2 #pip install opencv-python

%matplotlib inline

#Функция распознавания цвета

def ColorRecognition(frame, h\_min, h\_max):

hsv\_cam = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV) #Преобразование кадра в формат HSV

filt\_cam = cv2.inRange(hsv\_cam, h\_min, h\_max) #Фильтрация в заданном диапазоне HSV

st1 = cv2.getStructuringElement(cv2.MORPH\_RECT, (21, 21), (10, 10)) #Морфологическое преобразование

st2 = cv2.getStructuringElement(cv2.MORPH\_RECT, (11, 11), (5, 5)) #Морфологическое преобразование

filt\_cam = cv2.morphologyEx(filt\_cam, cv2.MORPH\_CLOSE, st1) #Морфологическое преобразование

filt\_cam = cv2.morphologyEx(filt\_cam, cv2.MORPH\_OPEN, st2) #Морфологическое преобразование

filt\_cam = cv2.GaussianBlur(filt\_cam, (5, 5), 2) #Размывание

im2, contours, hierarchy = cv2.findContours(filt\_cam, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE) #Нахождение контуров объектов

cv2.drawContours(frame, contours, -1, (0, 0, 255), -1) #Отрисовка контуров

frame = frame.astype('uint8')

frame = img\_as\_ubyte(frame)

return frame

#Захват изображения с камеры и применение алгоритма распознавания цвета в реальном времени

cap = cv2.VideoCapture(0)

h\_min = (100, 100, 100) #Минимальное значение цвета в пространстве HSV

h\_max = (110, 255, 255) #Максимальное значение цвета в пространстве HSV

while(cap.isOpened()):

ret, frame = cap.read()

if ret:

ColorRecognition(frame, h\_min, h\_max)

cv2.imshow('Color recognition', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

cap.release()

cv2.destroyAllWindows()