Государственное образовательное учреждение высшего образования



*«Московский государственный технический университет*

*им. Н.Э. Баумана (национальный исследовательский университет)»*

*(МГТУ им. Н.Э. Баумана)*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*ФАКУЛЬТЕТ «СПЕЦИАЛЬНОЕ МАШИНОСТРОЕНИЕ»*

*КАФЕДРА «ПОДВОДНЫЕ АППАРАТЫ И РОБОТЫ»*

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА СТУДЕНТА НА ТЕМУ

«СРАВНЕНИЕ РАЗЛИЧНЫХ ТИПОВ ОПОРНЫХ МАРКЕРОВ»

|  |  |  |
| --- | --- | --- |
| Руководитель НИРС | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (Макашов А.А.) |
|  | (подпись, дата) |  |
| Исполнитель НИРС,  студент группы СМ11-11М |  | (Андреев Е.В.) |

Москва, 2019 г.

РЕФЕРАТ

Отчёт на \_\_ стр., \_ ч., 20 рис., 11 источников, 3 таблицы.

СРАВНЕНИЕ РАЗЛИЧНЫХ ТИПОВ ОПОРНЫХ МАРКЕРОВ.

Перечень ключевых слов:

Целью данной работы является

Результаты работы:

В данной работе были определены

Оглавление

[Введение 4](#_Toc26523688)

[1. Маркеры ARuCO 5](#_Toc26523689)

[1.1 Определение зависимости размера маркера от расстояния и разрешения камеры 6](#_Toc26523690)

[2. Светодиод как опорный маркер 10](#_Toc26523691)

[2.1 Алгоритм детектирования светодиодов на основе ярокости 10](#_Toc26523692)

[2.2 Алгоритм детектирования светодиодов на основе преобразования Хаффа 11](#_Toc26523693)

[3. Выбор камеры 11](#_Toc26523694)

[Заключение 12](#_Toc26523695)

[Список использованных источников 13](#_Toc26523696)

[Приложение 1. Исходный код примера алгоритма детектирования светодиодов 14](#_Toc26523697)

# Введение

В данной работе объектом исследования являются опорные маркеры, использующиеся в робототехнике для навигации в пространстве. АНПА необходимо произвести стыковку с донной зарядной станцией. Считаем, что аппарат оснащён всеми необходимыми датчиками, вопросы маневрирования в данной работе не рассматриваются. Подход к станции на расстояние 10 м осуществляется по гидроакустическому маячку.

Предлагается осуществлять наведение и позиционирование аппарата вблизи с помощью видеокамер (ВК). установленных на АНПА и опорных маркеров определённого типа.

Цель работы – изучение различных типов опорных маркеров и определение их применимости в подводной робототехнике.

Задачи:

- исследование имеющихся готовых программных решений для обнаружения и определения маркеров;

- определение зависимости габаритов маркера от дальности и разрешения ВК;

- определение влияния шумов и эффекта размытия на точность идентификации маркеров;

- разработка алгоритма распознавания светодиодов в качестве реперных объектов;

- компоновка конструкции зарядной станции и выбранных средств наведения.

# 1. Маркеры ARuCO

Опорным маркером может являться любая фигура. Однако на практике разработчики программного обеспечения мобильных роботов ограничины такими факторами, как разрешение видеокамеры, особенностями цветопередачи и освещённости конкретной среды эксплуатации, вычислительной мощностью оборудования. Поэтому выбирается обычно черно-белый маркер простой формы. Как правило это прямоугольник или квадрат со вписанным во внутрь идентификатором-образом.  
В статье [1] описаны основные типы маркеров (рис. Рисунок 1).

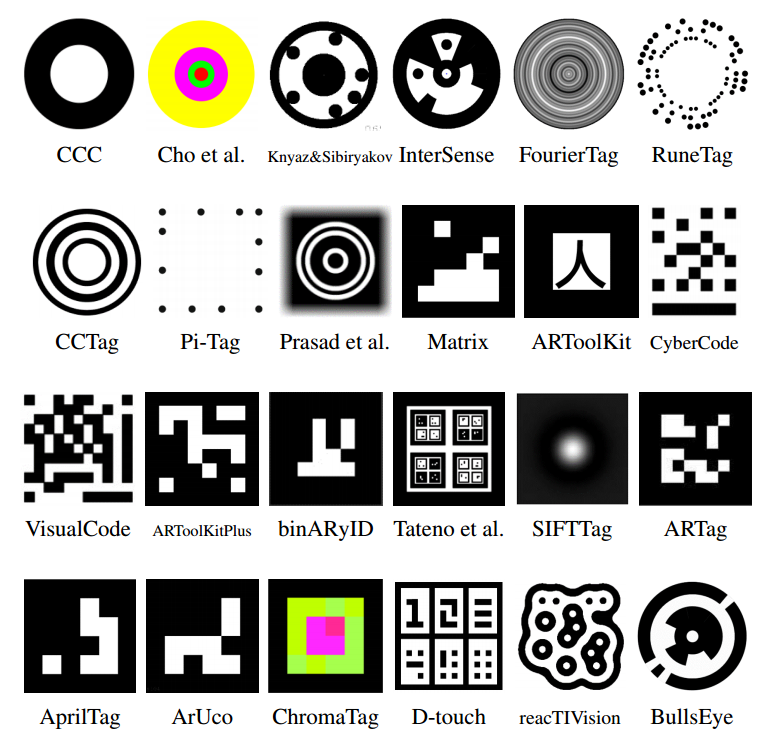


Рисунок – Различные типы опорных маркеров

Одним из наиболее популярных типов является ARuCo маркер, поскольку его реализация включена в стандартный пакет поставки билиотеки компьютерного зрения OpenCV (до версии 4) и вынесена в отдельный модуль пакета OpenCV-contrib начиная с версии 4.

## 1.1 Определение зависимости размера маркера от расстояния и разрешения камеры

Выявим зависимость габаритов маркера от дальности обнаружения и разрешения камеры экспериментальным путём (рис. Рисунок 2).

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок – Проведение эксперимента

Для этого выбранный маркер был распечатан разных размеров, условно обозничаемых далее как «xs», «s», «m» и «l». ВК настроена на максимально возможное разешение – 1080p.

На рисунке Рисунок 3 показана схема эксперимента. Соответствующие расстояния указаны ниже:

*l1* = 515 мм,

*l2* = 352 мм,

*l3* = 187 мм.

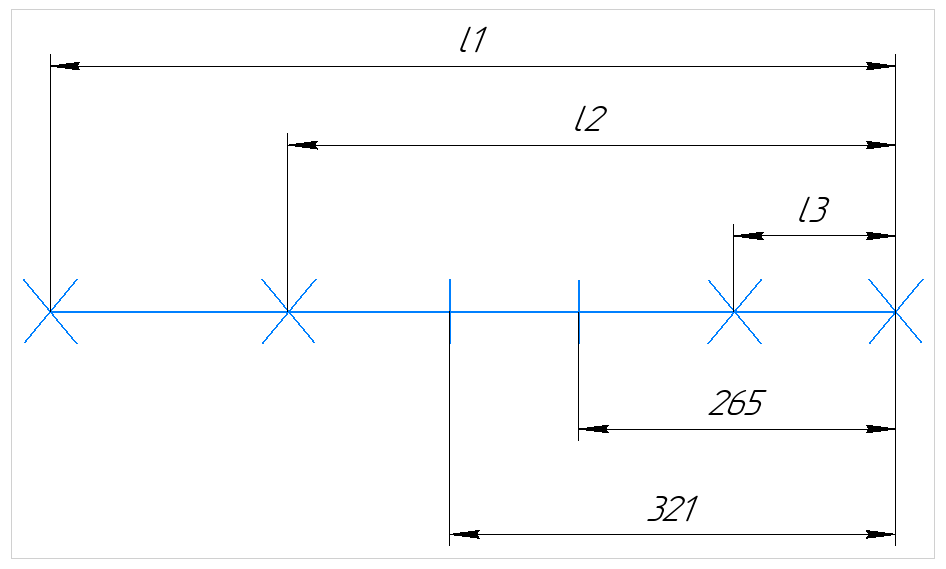


Рисунок – Схема эксперимента

Требовалось определить возможность идентификации маркеров «xs» и «s» на выбранных расстояниях. Результаты эксперимента приведены в таблице Таблица 1.

Таблица .

Экспериментальные данные обнаружения

|  |  |  |
| --- | --- | --- |
| Размер | Расстояние | Определяется? |
| xs (60 x 60 мм) | *l1* | Нет |
| *l2* | Нет |
| *l3* | Да |
| s (90 x 90 мм) | *l1* | Нет |
| *l2* | Нет |
| *l3* | Да |
| m (130 x 130 мм) | *l1* | Да |
| *l2* | Да |
| *l3* | Да |

Пороговое расстояние (см. рис. Рисунок 3Рисунок ) для маркера xs равно 265 мм, при этом сам маркер имеет размеры 33 х 33 пикселя (рис. Рисунок 4).

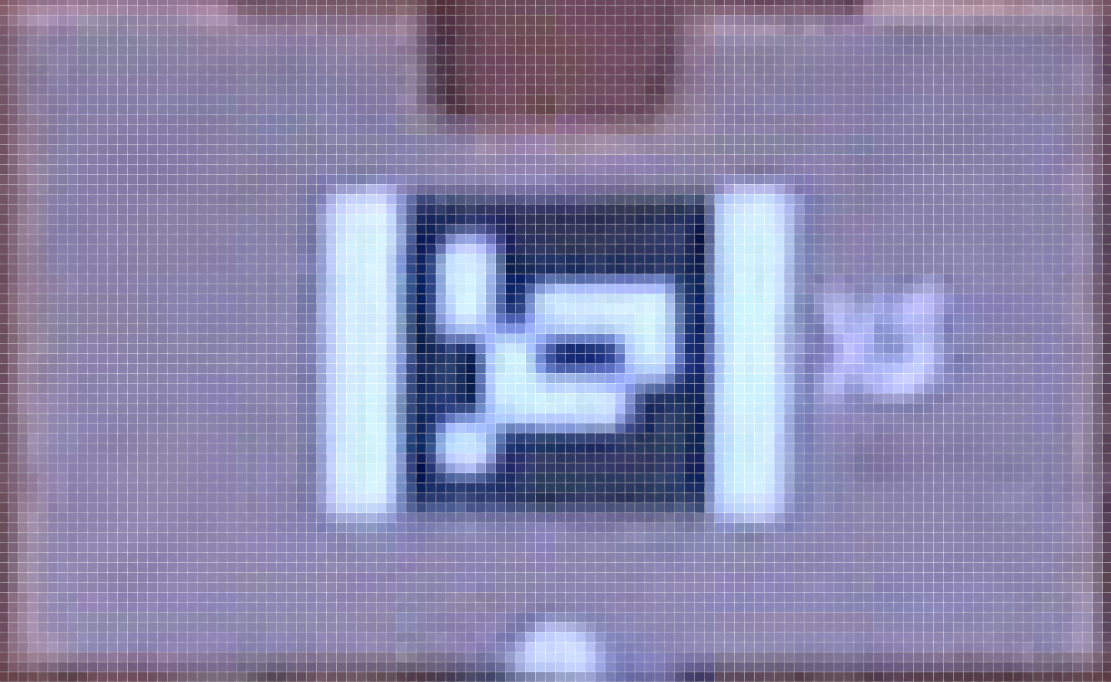


Рисунок – Пикселизация маркера «xs»

Пороговое расстояние для маркера s равно 321 мм, при этом сам маркер имеет размеры 40 х 40 пикселя (рис. Рисунок 5).

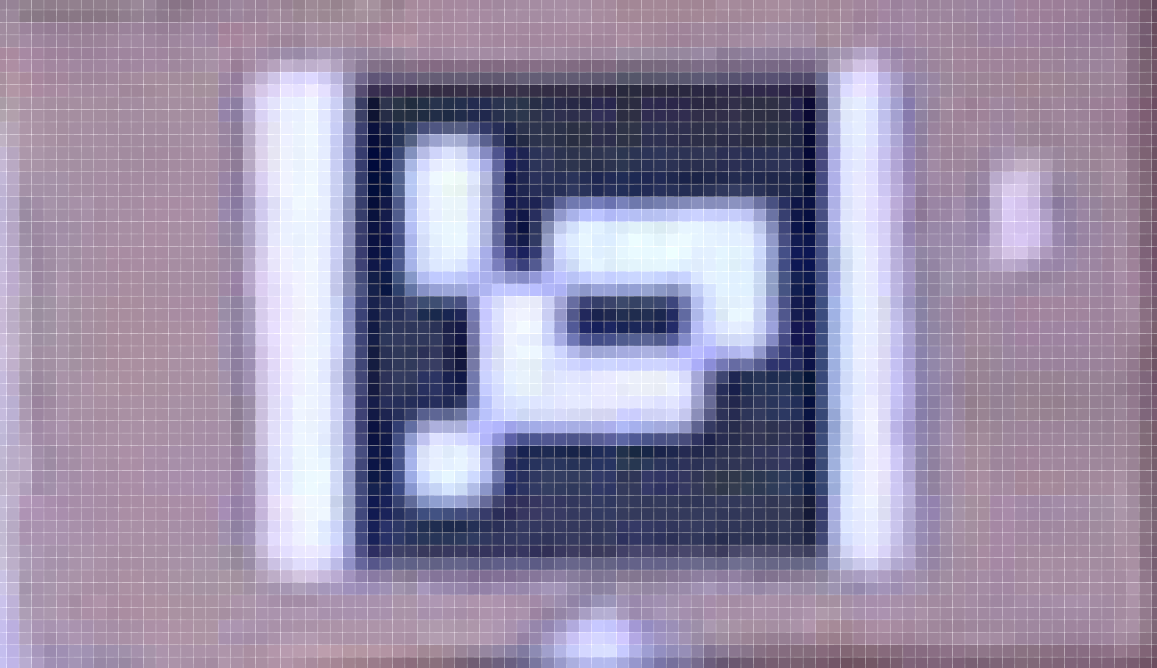


Рисунок – Пикселизация маркера «s»

Выведем формулу, связывающую расстояние от камеры до объекта распознавания с габаритами последнего при заданном разрешении ВК. Очевидно, что размер макрека зависит прямо пропорционально от дальности и обратно пропорционально от разрешения. Влияние угла обзора камеры и его уменьшение при вереходе лучей из воздушной среды в водную, искажения, вызванные движением воды, а также возможность обнаружения маркера под углом на данном этапе не учитываем.

,

M – размер маркера, мм

k – масштабный коэффициент, пиксели

l - расстояние до маркера

r – разрешение камеры по ширине кадра

, отсюда .

Примем.

Таким образом, в идеальных условиях габариты опорного маркера, необходимые для его обнаружения с 10 метров:

,

что является неудовлетворительным результатом.

# 2. Светодиод как опорный маркер

Хорошим опорным объектом может являться светоизлучающий диод (СИД), поскольку он имеем малые размеры и будет в гораздо меньшей степени подвержен влиянию различных донных течений. К тому же, СИД очень близок к точечному источнику света, что облегчает детектирование.

## 2.1 Алгоритм детектирования светодиодов на основе ярокости

Предлагается использовать следующую последовательность действий для обнаружения и определения положения СИД:

1. Получение изображения в оттенках серого.
2. Извлечение порогового изображения.
3. Фильтрация и сглаживание порогового изображения.
4. Нахождение контуров объектов с помощью детектора Канни [].
5. Определене описывающего прямоугольника для каждого контура.
6. Нахождение эллипса для каждого контура.
7. Вычисление центра как полусуммы координат фокусов эллипса.

|  |  |  |
| --- | --- | --- |
| C:\Users\Eugene\AppData\Local\Microsoft\Windows\INetCache\Content.Word\s1.png | C:\Users\Eugene\AppData\Local\Microsoft\Windows\INetCache\Content.Word\s2.png | C:\Users\Eugene\AppData\Local\Microsoft\Windows\INetCache\Content.Word\s3.png |
| C:\Users\Eugene\AppData\Local\Microsoft\Windows\INetCache\Content.Word\s4.png | C:\Users\Eugene\AppData\Local\Microsoft\Windows\INetCache\Content.Word\s5.png | C:\Users\Eugene\AppData\Local\Microsoft\Windows\INetCache\Content.Word\s6.png |

Рисунок – Последовательность преобразования изображения для обнаружения положений светодиодов

Пороговое изображение определяется по уровню яркости, который задан в виде постоянного значения. В качестве сглаживающей операции используется морфологическое преобразование «opening» [], которое является комбинацией преобразований «erode» [] и «dilite» []. Матрица преобразования прямоугольная размера 5х5.

Детектор Канни призван обнаруживать границы объектов на изображении. Для его работы необходимо задать наибольшее и наименьшее ожидаемые значения яркости пикселей на границах обкъектов (в нашем случае контуров).

Исходный код примера исползования алгоритма приведён в приложении 1.

## 2.2 Алгоритм детектирования светодиодов на основе преобразования Хаффа

# 3. Выбор камеры

# Заключение

# Список использованных источников

# Приложение 1. Исходный код примера алгоритма детектирования светодиодов

//https://docs.opencv.org/4.1.1/de/d62/tutorial\_bounding\_rotated\_ellipses.html

//https://docs.opencv.org/4.1.1/d3/dbe/tutorial\_opening\_closing\_hats.html

#include <opencv2/opencv.hpp>

#include <opencv2/aruco.hpp>

using namespace cv;

using namespace std;

int threshold\_value = 245;

//int threshold\_type = 3;

int const max\_value = 255;

int const max\_type = 4;

int const max\_binary\_value = 255;

float treshold\_area = 30 \* 20;

string path = "../../sources/";

string dir = "";

string file = "led\_blue.jpg";

string full\_path = path + dir + file;

Mat image, image\_gray, hough, tresh, tresh\_opening, canny\_output, drawing;

RNG rng(12345);

vector<Vec3f> circles;

void morphology\_Operation(int morph\_operator, Mat& src, Mat & dst)

{

int morph\_elem = MORPH\_RECT;

int morph\_size = 5;

Mat element = getStructuringElement(morph\_elem, Size(2 \* morph\_size + 1, 2 \* morph\_size + 1), Point(morph\_size, morph\_size));

morphologyEx(src, dst, morph\_operator, element);

imshow("Morph operation", dst);

}

int main(int argc, char\*\* argv)

{

image = imread(full\_path, IMREAD\_COLOR);

cvtColor(image, image\_gray, COLOR\_BGR2GRAY); // Convert the image to Gray

//GaussianBlur(image, image\_copy, Size(13, 13), 0, 0);

image.copyTo(hough);

HoughCircles(image\_gray, circles, HOUGH\_GRADIENT, 1,

image\_gray.rows / 32, // change this value to detect circles with different distances to each other

100, 30, 20, 50 // change the last two parameters

// (min\_radius & max\_radius) to detect larger circles

);

for (size\_t i = 0; i < circles.size(); i++)

{

Vec3i c = circles[i];

Point center = Point(c[0], c[1]);

// circle center

circle(hough, center, 1, Scalar(0, 100, 100), 3, LINE\_AA);

// circle outline

int radius = c[2];

circle(hough, center, radius, Scalar(255, 0, 255), 3, LINE\_AA);

}

threshold(image\_gray, tresh, threshold\_value, max\_binary\_value, THRESH\_TOZERO);

morphology\_Operation(MORPH\_OPEN, tresh ,tresh\_opening);

//Canny(image\_gray, canny\_output, threshold\_value-30, 255);

//Canny(tresh, canny\_output, 0, 255);

Canny(tresh\_opening, canny\_output, 0, 255);

vector<vector<Point> > contours;

findContours(canny\_output, contours, RETR\_TREE, CHAIN\_APPROX\_SIMPLE, Point(0, 0));

vector<RotatedRect> minRect(contours.size());

vector<RotatedRect> realRect;

vector<RotatedRect> minEllipse;

for (size\_t i = 0; i < contours.size(); i++)

{

minRect[i] = minAreaRect(contours[i]);

if (contours[i].size() > 5 && (float(minRect[i].size.width) \* float(minRect[i].size.height) > treshold\_area)) {

realRect.push\_back(minRect[i]);

minEllipse.push\_back(fitEllipse(contours[i]));

}

}

drawing = Mat::zeros(canny\_output.size(), CV\_8UC3);

for (size\_t i = 0; i < realRect.size(); i++)

{

Scalar color = Scalar(rng.uniform(0, 256), rng.uniform(0, 256), rng.uniform(0, 256));

// contour

//drawContours(drawing, contours, (int)i, color);

// ellipse

ellipse(drawing, minEllipse[i], color, 2);

// rotated rectangle

Point2f rect\_points[4];

realRect[i].points(rect\_points);

for (int j = 0; j < 4; j++)

{

line(drawing, rect\_points[j], rect\_points[(j + 1) % 4], color);

}

}

imshow("Original", image);

imshow("Grayscale", image\_gray);

imshow("Treshold", tresh);

imshow("Hough Detector", hough);

imshow("Canny output", canny\_output);

imshow("Contours", drawing);

waitKey(0);

return 0;

}