МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования «Казанский национальный исследовательский
технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)

Институт компьютерных технологий и защиты информации
Кафедра автоматизированных систем обработки информации и управления
09.03.02 «Информационные системы и технологии»

КУРСОВАЯ РАБОТА

по дисциплине: «Цифровая обработка изображений»

на тему: «Геометрические характеристики форм»

Исполнитель: 4308 _____ Фейзулов Д.И.
(номер группы) (подпись, дата) (Ф.И.О.)

Руководитель: к. т. н., доцент Шлеймович М.П.

Курсовая работа зачтена с оценкой _____

_____
(подпись, дата)

Казань 2020

The Ministry of Education and Science
Federal State Educational Institution of Higher Education «Kazan National Research Technical University named after A.N. Tupolev»
(KNITU-KAI)
Institute of Computer Technology and Information Protection
Department of automated information processing systems and management

Course work

by discipline «Digital Image Processing»

on the topic: « Geometric characteristics of forms »

Executor: 4308 _____ D.I. Feyzulov
(group number) (signature, date) (full name)

Head: Ph.D., associate professor of ASIPM M.P. Shleimovich

Course work credited with the assessment_____

_____
(signature, date)

Kazan 2020

Аннотация

Распознавание объектов — это технология, способная производить обнаружение, отслеживание и классификацию объектов. Для обнаружения объектов на изображении используются эталонные геометрические характеристики для сравнения с характеристиками объекта на изображении, Распознавание объектов широко применяется в средствах наблюдения, анализе медицинских изображений, а также в других повседневных областях.

В работе рассматривается метод выделения объектов на изображении, определение его характеристик, таких как периметр, площадь, коэффициент округления, инвариантные моменты, а также центр масс и геометрический центр объекта. Этот метод будем реализовать в среде разработки Microsoft Visual Studio 2019, на языке разработки C++ с библиотекой OpenCV 2.4.9.

Annotation

Object recognition is a technology that can detect, track and classify objects. To detect objects in the image, reference geometric characteristics are used to compare with the characteristics of the object in the image. Object recognition is widely used in surveillance tools, analysis of medical images, as well as in other everyday areas.

The paper considers the method of selecting objects in the image, determining its characteristics, such as the perimeter, area, rounding coefficient, invariant moments, as well as the center of mass and the geometric center of the object. We will implement this method in the Microsoft Visual Studio 2019 development environment, in the C ++ development language with the OpenCV 2.4.9 library.

СОДЕРЖАНИЕ

1.Постановка задачи..... 7
2.Описание решения задачи 7
2.1 Общий алгоритм решения задачи..... 7
2.2. Методы цифровой обработки изображений, применяемые для решения поставленной задачи 9
2.2.1. Выделение границ с использованием алгоритма Канни 9
2.2.2 Определение периметра и площади контура 11
2.2.3 Определение дескриптора компактности..... 12
2.2.4 Определение инвариантных моментов, центра масс и геометрического центра 12
2.2.5 Определение схожести объектов со стандартными фигурами……...14
3. Спецификация программы..... 14
3.1. Реализация и алгоритм работы программы 14
3.2. Результат работы программы с разными значениями порогов 20
Заключение..... 21
Conclusion..... 22
Список литературы..... 23
Приложение 1..... 24

CONTENT

1. Statement of the problem 7
2. Description of the solution to problem 7
2.1 General algorithm for solving problem 7
2.2. Digital image processing methods used to solve the problem 9
2.2.1. Extraction of borders using the Canny algorithm 9
2.2.2 Determination of the perimeter and area of the circuit 11
2.2.3 Definition of the compactness descriptor 12
2.2.4 Determination of invariant moments, center of mass and geometric center 12
2.2.5 Determining the Similarity of Objects to Standard Shapes 14
3. Program specification. 14
3.1. Processing methods for solving problem 14
3.2. The result of the program with different thresholds 20
Conclusion. 21
Conclusion. 22
References 23
Appendix 1. 24

1.Постановка задачи

Задача: «Геометрические характеристики форм». С помощью анализа контуров мы должны выявить геометрические характеристики объекта на изображении.

2.Описание решения задачи

2.1. Общий алгоритм решения задачи

Задачей данной работы является выявление геометрических характеристик объектов изображения: периметр, площадь, дескриптор компактности, инвариантные моменты, центр масс, геометрический центр для выявления схожестей объектов с окружностью, квадратом, треугольником.

Объект – сущность в виртуальном пространстве, обладающая определённым состоянием и поведением, имеет заданные значения свойств и операций над ними.

Контур – это граница плоской фигуры.

Периметр – длина замкнутого контура плоской фигуры.

Площадь – величина, измеряющая размер поверхности.

Дескриптор компактности – это коэффициент округлости, который определяется как отношение площади области к площади круга (максимально компактная фигура) с таким же периметром.

Центр масс – геометрическая точка, характеризующая движение тела или системы частиц как целого.

Геометрический центр – это точка, в которой минимизируется сумма расстояний до точек множества.

Инвариантные моменты – это набор моментов, являющийся инвариантным по отношению к параллельному переносу, зеркальному отражению (с точностью до перемены знака), повороту и изменению масштаба.

Для выявления данных характеристик объектов, необходимо сначала выделить, а затем найти их контуры. Наиболее эффективным методом

выделения контуров является детектор Кенни. Алгоритм детектора Кенни основывается на трех критериях: хорошее обнаружение, то есть увеличение отношения сигнал/шум, вторым критерием является высокий уровень локализации, что означает правильное определение положения границы, третьим критерием является единственный отклик на одну границу. Для уменьшения вычислительных затрат изображение преобразовывается в оттенки серого, эти действия выполняются перед применением детектора.

Для нахождения же контуров используется алгоритм [Suzuki85], разработанный Сатоши Сузуки и Кейити Абе в 1983 году. Они разработали два алгоритма для распознавания контуров. Первый определяет отношения окружения между границами двоичного изображения. Поскольку внешние границы и границы отверстий имеют взаимно-однозначное соответствие подключенным компонентам в 1-пиксель и отверстиям соответственно, предлагаемый алгоритм дает представление двоичного изображения, из которого можно извлечь некоторый вид функции без реконструкции изображения. Второй алгоритм, который является модифицированной версией первого, следует только самым внешним границам (то есть внешним границам, которые не окружены отверстиями).

После нахождения необходимых контуров, мы используем алгоритмы для нахождения площади области как число содержащихся в ней пикселей, периметр как длина границы области, дескриптор компактности как отношение площади области к площади круга с таким же периметром, центр масс как отношение моментов первого порядка по x и y к нулевому.

Для определения схожести объекта со стандартными фигурами (окружность, квадрат, треугольник) находится среднеквадратичное отклонение инвариантных моментов от инвариантных моментов стандартных фигур.

2.2. Методы цифровой обработки изображений, применяемые для решения поставленной задачи

2.2.1. Выделение границ с использованием алгоритма Канни

Алгоритм состоит из пяти отдельных шагов: 1. Первый шаг — это сглаживание. Оно используется, когда во избежание появления ложных границ требуется уменьшить количество шумов на изображении. Для этого часто используется размытие фильтром Гаусса.

В зависимости от условий, в которых применяется детектор, необходимо выбирать параметр σ, который обеспечивает наилучшее подавление шума. Чрезмерное увеличение этого параметра приведет к усилению усреднения вплоть до равномерно черного цвета всех пикселей.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}\right); 1 \le i, j \le (2k+1)$$

$\sigma -$ степень размытия

$i, j -$ размер ядра фильтра

2. Второй шаг — это поиск градиентов. Контур отмечается там, где значение градиента изображения имеет наибольшее значение. На этом этапе используется оператор Собеля, который использует две маски размерности 3*3:

$$G_{x=} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix};$$

$$G_{y=} \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}.$$

Величина градиента и его направление определяются соответственно по формулам:

$$G = \sqrt{G_x^2 + G_y^2} \quad ;$$

$$\theta = arctan\frac{G_y}{G_x} .$$

Полученные значения направлений градиента округляются до одного из четырех углов: 0, 45, 90, 135 градусов.

3. Третий шаг - это подавление немаксимумов. Пикселями границ объявляются пиксели, в которых достигается локальный максимум градиента в направлении вектора градиента. Этот шаг позволяет не разрывать саму границу.

Направление градиента есть направление максимального возрастания функции. На этом основана процедура подавления немаксимумов. При этой процедуре для каждой точки рассматривается отрезок длиной в несколько пикселей, ориентированный по направлению градиента и с центром в рассматриваемом пикселе. Пиксель считается максимальным тогда и только тогда, когда длина градиента в нем максимальна среди всех длин градиентов пикселей отрезка. Граничными можно признать все максимальные пиксели с длинами градиента больше некоего порога.

Градиент яркости в каждой точке перпендикулярен границе, поэтому после подавления немаксимумов жирных линий не остается: на каждом перпендикулярном сечении жирной линии останется один пиксель с максимальной длиной градиента.

4. Четвертый шаг - двойная пороговая фильтрация (гистерезисом). Возможные границы определяются на основе нижних и верхних порогов. Если значение градиента в каком-то месте на просматриваемом фрагменте превысит верхний порог, то этот элемент остается также допустимой границей и в тех местах, где значение градиента падает ниже этого порога, до тех пор, пока она не станет ниже нижнего порога. Если же на всем фрагменте нет ни одной точки со значением большим верхнего порога, то он удаляется. При таком подходе фрагмент границы обрабатывается как целое, что способствует удалению слабых границ:

$$g_{NH}(x,y) = \begin{cases} g_N(x,y), & g_N(x,y) \geq T_H \\ 0 & g_N(x,y) < T_H \end{cases};$$

$$g_{NL}(x,y) = \begin{cases} g_N(x,y), & g_N(x,y) \geq T_L \\ 0 & g_N(x,y) < T_L \end{cases};$$

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y) \quad .$$

T_H–верхний порог, T_L–нижний порог

Ненулевые пиксели изображений $g_{NH}(x,y)$ и $g_{NL}(x,y)$ являются сильными и слабыми и контурными точками.

5. Трассировка области неоднозначности. Итоговые границы определяются путем подавления всех краев, несвязанных с сильными границами. Таким образом задача сводится к выделению групп пикселей, получивших на предыдущем шаге промежуточное значение и отнесению их к границе, при условии, что они соединены с одной из установленных границ, в противном случае подавлению их. Добавление пикселя к группе происходит, если он соприкасается с ней по одному из 8-ми направлений.

2.2.2 Определение периметра и площади контура

Площадь области определяется формулой Грина. Пусть в плоскости *Oxy* задана область *R*, ограниченная замкнутой, кусочно-непрерывной и гладкой кривой *C*. Предположим, что в некоторой области, содержащей R, задана непрерывная векторная функция (i,j – единичные векторы по осям *x* и *y*): $F = P(x,y)i + Q(x,y)j$.

С непрерывными частными производными первого порядка ∂P/∂y,∂Q/∂x. Тогда справедлива формула Грина:

$$\iint_R \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dxdy = \oint_C Pdx + Qdy,$$

где символ $\oint C$ указывает, что кривая (контур) *C* является замкнутой, и обход при интегрировании вдоль этой кривой производится против часовой стрелки. Если $Q=x$, $P=-y$, то формула Грина принимает вид:

$$S = \iint_R dxdy = 1/2 \oint_C xdy - ydx,$$

где S — это площадь области R, ограниченной контуром C.

Контур является кривой, поэтому для нахождения её длины используется формула Ньютона-Лейбница.

Пусть Γ - кривая, заданная своим непрерывно дифференцируемым векторным представлением $\boldsymbol{r} = \boldsymbol{r}(t)$, $a \leq t \leq b$; тогда она спрямляема, и если $s = s(t)$ - ее переменная длина дуги, отсчитываемая от начала, то функция $s(t)$ дифференцируема и $s'(t) = |\boldsymbol{r}'(t)|$. По формуле Ньютона-Лейбница для длины $S = s(b)$ кривой имеем формулу:

$$S = s(b) = \int_a^b s'(t)dt = \int_a^b |r'(t)|'dt.$$

2.2.3 Определение дескриптора компактности

Площадь области определяется как число содержащихся в ней пикселей. Периметр области есть длина ее границы. Хотя площадь и периметр иногда и применяются в качестве дескрипторов, это относится преимущественно к тем случаям, когда размеры интересующих областей не меняются. Чаще эти дескрипторы используются при вычислении меры компактности области, которая определяется как отношение квадрата периметра к площади. Немного отличающийся (скалярным множителем) дескриптор компактности — это коэффициент округлости, который определяется как отношение площади области к площади круга (максимально компактная фигура) с таким же периметром. Поскольку площадь круга с периметром P равна $P^2/4\pi$, коэффициент округлости Rc задается формулой:

$$R_C = \frac{4\pi A}{P^2},$$

где A — площадь рассматриваемой области, а P — ее периметр. Такая характеристика принимает значение 1 для круглой области и $\pi/4$ — для квадратной. Компактность является безразмерной величиной (и поэтому инвариантна к однородным изменениям масштаба). С точностью до

погрешностей, возникающих при повороте дискретных областей, компактность также инвариантна к ориентации объекта.

2.2.4 Определение инвариантных моментов, центра масс и геометрического центра

Момент – это характеристика контура, объединённая (суммированная) со всеми пикселями контура. Момент (*p,q*) определяется как:

$$m_{pq} = \sum I(x,y)x^p y^q,$$

где *p* – порядок *x*, *q* – порядок *y*, где порядок означает, так сказать, мощность, на которой соответствующий компонент взят в сумме с другими отображенными.

В структуре появились ещё и центральные моменты, которые вычисляются по следующей формуле:

$$\mu_{pq} = \Sigma I(x,y)\left(x - x_{avg}\right)^p \left(y - y_{avg}\right)^q,$$

где $x_{avg} = m10/\,m00$ и $y_{avg} = m01/\,m00$, что является центром масс контура.

Центральные моменты можно нормализовать следующим образом:

$$\eta_{pq} = \frac{\mu_{pq}}{m_{00}^{(p+q)/2+1}}.$$

Для нахождения инвариантных моментов, комбинируя различные нормализованные центральные моменты, нужно создавать инвариантные функции, представляющие различные аспекты изображения, которые не зависят от масштабов, вращения и отражения:

$$h_1 = \eta_{20} + \eta_{02};$$

$$h_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2;$$

$$h_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2;$$

$$h_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2;$$

$$h_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} -$$

$$- \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2];$$

$h_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} +$
$+ \eta_{03})$;

$h_7 = (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] - (\eta_{30} -$
$- 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$.

2.2.5 Определение схожести объектов со стандартными фигурами

Для выявления схожести с конкретной фигурой используется вычисление среднеквадратических отклонений со всеми стандартными фигурами данного объекта. В качестве параметров выбираются инвариантные моменты стандартных фигур и вычисленные раннее инвариантные моменты всех объектов, расположенных на изображении.

Во-первых, нормализуются все параметры рассматриваемого объекта по формуле:

$$\overline{x_i} = x_i / \sqrt{\sum_{i=1}^{n} x_i^2},$$

где $\overline{x_i}$ – нормализованный параметр, x_i – нормализуемый параметр, i – текущий параметр, n – количество параметров.

Далее необходимо высчитать среднеквадратическое отклонение относительно каждой стандартной фигуры по формуле:

$$\sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i^{\Delta} - \overline{x_i})^2,$$

где n – количество параметров, $\overline{x_i}$ – нормализованный параметр, x_i^{Δ}- параметр стандартной фигуры.

В конце мы сравниваем эти отклонения. Относительно какой стандартной фигуры будет наименьшее отклонение, на такую фигуры и наиболее похож рассматриваемый объект.

3. Спецификация программы

3.1. Реализация и алгоритм работы программы

Загрузка исходного изображения:

image= imread("example3.jpg", IMREAD_UNCHANGED)

imread - загрузка изображения из файла,

“example3.jpg" - путь к файлу (относительная адресация),

IMREAD_UNCHANGED - загрузка изображения без изменений.

Вывод исходного изображения:

imshow("example", image)

"example" - название окна,

image - изображение для показа.

Смена цветового пространства на чёрно-белое:

cvtColor(image, gray, CV_BGR2GRAY);

image – исходное изображение,

gray – итоговое изображение,

CV_BGR2GRAY – новое цветовое пространство.

Размытие изображения с использованием фильтра Гаусса:

GaussianBlur(gray, gray, Size(3, 3), 0);

gray – исходное изображение,

gray – итоговое изображение,

Size(3, 3) – размер маски,

0 -Стандартное отклонение ядра Гаусса в направлении Y.

Выделение границ изображения:

Canny(gray,edged, 10, 250);

gray — одноканальное изображение для обработки (градации серого),

edged — одноканальное изображение для хранения границ, найденных функцией, 10 — порог минимума, 250 — порог максимума.

Возвращение структурирующего элемента указанного размера и формы для морфологических операций.

kernel= getStructuringElement(MORPH_RECT, Size(7, 7));

MORPH_RECT - прямоугольный структурирующий элемент,

Size(7, 7) – размер структурирующего элемента.

Морфологические операции:

morphologyEx(edged,closed, MORPH_CLOSE, kernel);

edged – исходное изображение,

closed – итоговое изображение,

MORPH_CLOSE – операция закрытия,

kernel – структурирующий элемент, используемый для расширения.

Определение площади контура:

area = abs(contourArea(cont, true));

area – значение площади,

cont – текущий контур,

true - функция возвращает значение области со знаком, в зависимости от ориентации контура (по часовой стрелке или против часовой стрелки).

Определение периметра контура:

peri = arcLength(cont , true);

peri – значение периметра;

cont – текущий контур;

true – контур закрыт.

Определение коэффецента округлости:

round = 4 * 3.14159265358979323846*area / (peri*peri);

round – значение коэффицента;

area – текущая площадь;

peri – текущий периметр.

Определение моментов контура до 3 порядка:

Moments moment = moments(cont, false);

Moment – список моментов;

Cont – текщий контур.

Нахождение опоясывающего прямоугольника:

rect = boundingRect(cont);

rect – значения опоясывающего прямоугольника;

cont – текущий контур.

Вычисление геометрического центра:

prect.x = int(rect.x + rect.width / 2);

prect.y = int(rect.y + rect.height / 2);

prect.x prect.y – координаты геометрического центра;

rect.x rect.y – координаты окружающего прямоугольника;

rect.width rect.height – длина и ширина прямоугольника.

Координаты центра масс

pmass.x = int(moment.m10/moment.m00);

pmass.y = int(moment.m01 / moment.m00);

pmass.x pmass.y – координаты центра масс;

moment.m10 – момент первого порядка по x;

moment.m01 – момент первого порядка по y;

moment.m00 – момент нулевого порядка.

Вычисление Hu моментов

HuMoments(moment, huMoments);

Moment – моменты текущего контура;

huMoments – Hu моменты текущего контура.

Приведение всех Hu моментов текущего контура в сопоставимый масштаб:

huMoments[i] = -1 * copysign(1.0, huMoments[i]) * log10(abs(huMoments[i]));

copysign - Возвращает значение с величиной 1 и знаком huMoments[i];

log10 – десятичный логарифм.

Создание окна изображения

namedWindow("example", WINDOW_AUTOSIZE);

“example”- название окна;

WINDOW_AUTOSIZE – автоматическое определение масштаба окна.

Вывод окна с изображением:

imshow("example", image);

“example” – название окна;

Image – переменная изображения;

Mat gray, edged, kernel, closed, closed1, centers – переменные изображений;

vector< vector<Point> > contours – контейнер для контуров;

vector<Point> cont – контейнер точек текущего контура;

double area – переменная площади;

double peri – переменная периметра;

double round – переменная коэффициента округления;

Point prect – точка геометрического центра;

Point pmass – точка центра масс;

double huMoments[7] – массив Hu моментов.

Расчёт среднеквадратического отклонения:

sqrt(auf) – квадратный корень числа;

pow(huMoments[0], 2) – возведение числа в квадрат;

double auf = pow(huMoments[0], 2) + pow(huMoments[1], 2) +
pow(huMoments[2], 2) + pow(huMoments[3], 2) + pow(huMoments[4], 2) +
pow(huMoments[5], 2) + pow(huMoments[6], 2);

double delitel = sqrt(auf) - расчёт делителя для формулы нормализации;

huMoments[i] = abs(huMoments[i] / sqrt(auf)) – нормализация параметров рассматриваемого объекта;

double masKrug[], double masKvadrat[], double masTreugolnik[] – набор параметров стандартных фигур;

double pervieTriKrug, double pervieTriKvadrat, double pervieTriTreugolnik – переменные для расчёта суммы в формуле отклонения;

pervieTriKrug += pow(masKrug[i] - huMoments[i], 2) – расчёт слагаемых суммы для формулы отклонения;

double otklonenieKrug = pervieTriKrug/7 – расчёт отклонения;

double raznica – переменная для расчёта минимального отклонения;

krug/kvadrat/treugolnik = total – переменная, в которую записывается номер объекта (обозначающая окружность, квадрат, треугольник) при нахождении минимального отклонения.

Блок-схема решения поставленной задачи (Рис.2):

Рис.1. Блок-схема решения поставленной задачи

3.2 Результат работы программы с разными значениями порогов

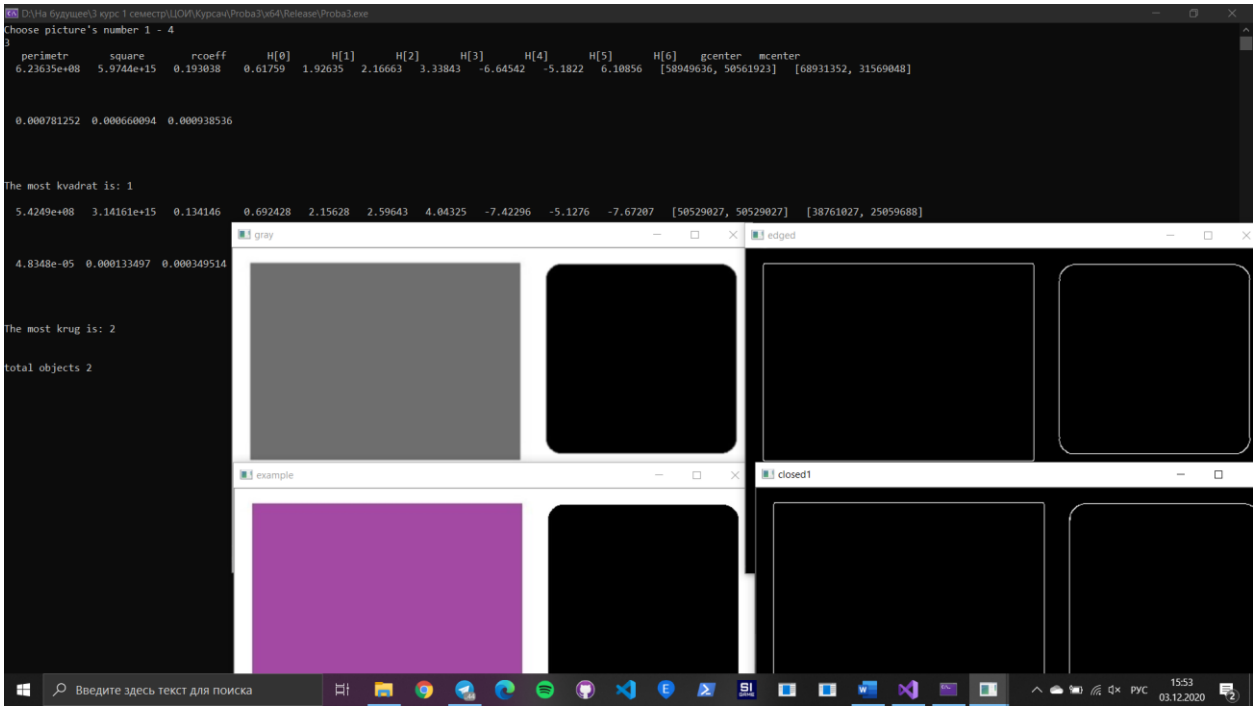


Рис.2. Результат изображения с найденными границами

Заключение

В ходе выполнения курсовой была разработана программа, находящая контуры объектов и определяющая её геометрические свойства. В ходе проделанной работы было реализовано консольное приложение, определяющее геометрические характеристики объекта: периметр, площадь, коэффициент округления, инвариантные моменты, а также центр масс и геометрический центр объекта. Приложение дает точный результат. На основании расчётов инвариантных моментов смогли настроить определение входных фигур как наиболее похожие на круг, квадрат, либо треугольник.

Conclusion

During the course work, a program was developed that finds the contours of objects and determines its geometric properties. In the course of the work done, a console application was implemented that determines the geometric characteristics of the object: perimeter, area, rounding coefficient, invariant moments, as well as the center of mass and the geometric center of the object. Debugging the application gives an approximately accurate result. The app gives an accurate result. Based on the calculations of invariant moments, we were able to configure the definition of the input figures as the most similar to a circle, square, or triangle.

4. Используемые источники

1. Документация OpenCV [Электронный ресурс]. - https://opencv.org

2. OpenCV шаг за шагом [Электронный ресурс]. - Режим доступа: http://robocraft.ru/blog/computervision/484.html

4.НОУ ИНТУИТ [Электронный ресурс] - Режим доступа: https://www.intuit.ru/studies/courses/993/163/lecture/4507?page=2

5. Шапиро Л., Компьютерное зрение / Л. Шапиро, Дж. Стокман ; пер. с англ. - 2-е изд. (эл.). - М. : БИНОМ, 2013. - 752 с.

Режим доступа: https://nashol.com/20190825113125/komputernoe-zrenie-shapiro-l-stokman-dj-2013.html

6. R. Gonzalez, Digital Image Processing / R. Gonzalez R. Woods - 3rd edition, revised and enlarged; Moscow: Technosphere, 2012 .-- 1104 p. , ISBN 978-5-94836-331-83.

http://www.technosphera.ru/files/book_pdf/0/book_311_455.pdf

7. SatoshiSuzuki KeiichiAbe Topological structural analysis of digitized binary images by border following. COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING N,32-46 (1985). Available at: https://www.sciencedirect.com/science/article/abs/pii/0734189X85900167

5. Приложения

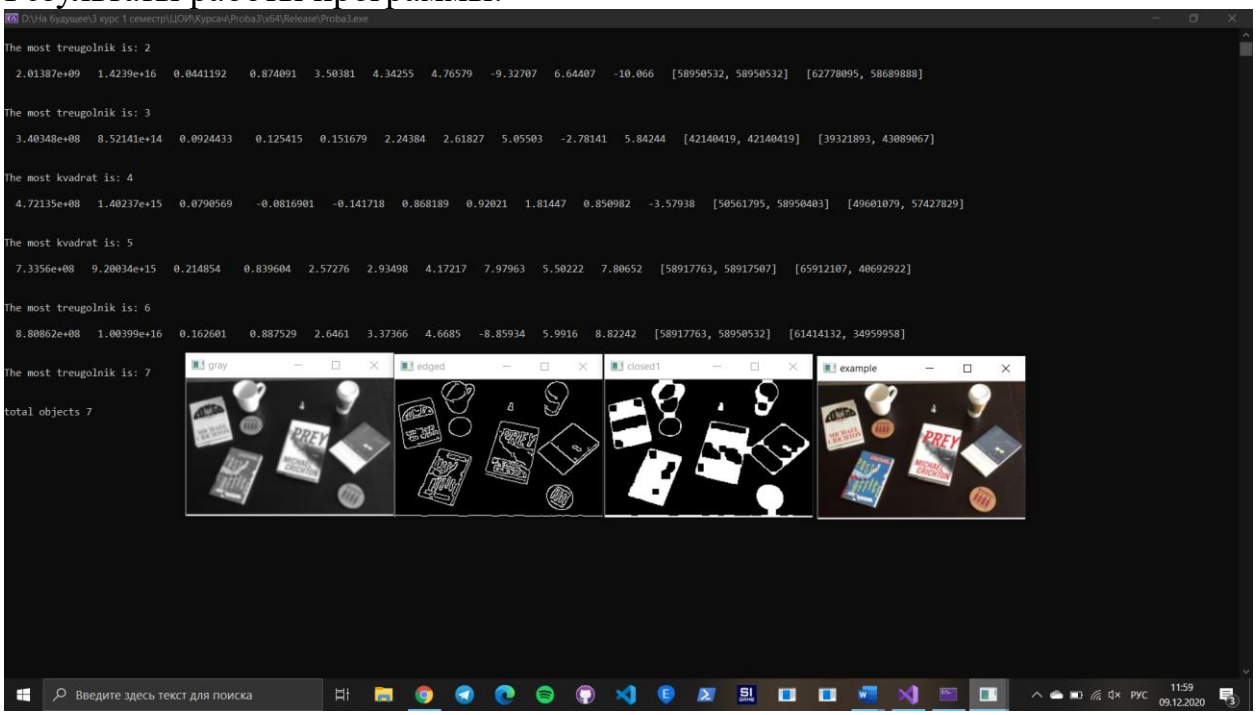
5.1. Код программы

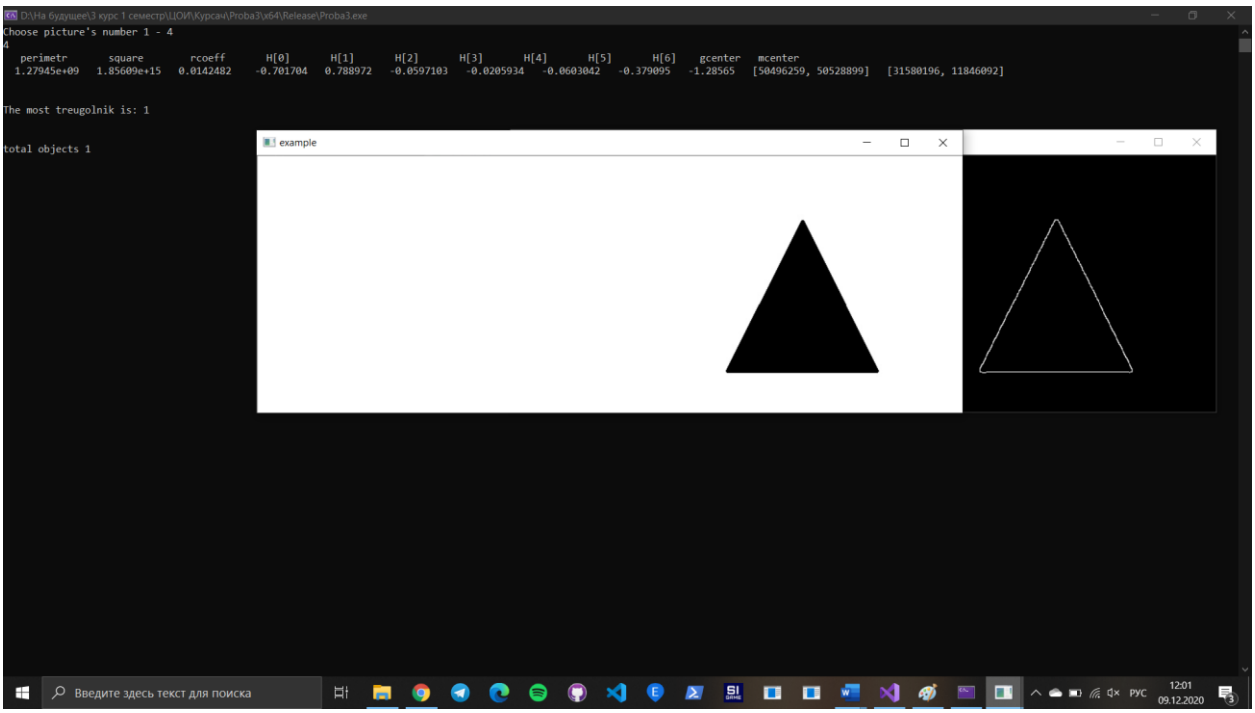
```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/core/mat.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
#include <math.h>
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdlib.h>

using namespace std;
using namespace cv;


int main(int argc, char** argv)
{
	int krug = 0;
	int kvadrat = 0;
	int treugolnik = 0;
	int total = 0;
	int pictureNumber;
	printf_s("Choose picture's number 1 - 4\n");
	cin >> pictureNumber;
	string picture;
	for (int i = 0; i < 3; i++)
	{
		if (pictureNumber == 1) picture = "example1.jpg";
		if (pictureNumber == 2) picture = "example2.jpg";
		if (pictureNumber == 3) picture = "example3.jpg";
		if (pictureNumber == 4) picture = "example4.jpg";
	}
	Mat image = imread(picture, IMREAD_UNCHANGED);
	Mat gray, edged, kernel, closed, closed1, centers;
	cvtColor(image, gray, CV_BGR2GRAY);
	GaussianBlur(gray, gray, Size(3, 3), 0);
	Canny(gray, edged, 10, 250);
	kernel = getStructuringElement(MORPH_RECT, Size(7, 7));
	morphologyEx(edged, closed, MORPH_CLOSE, kernel);
	closed1 = closed;
	centers = image;

	namedWindow("gray", WINDOW_AUTOSIZE);
	imshow("gray", gray);
	namedWindow("edged", WINDOW_AUTOSIZE);
	imshow("edged", edged);
	namedWindow("closed1", WINDOW_AUTOSIZE);
	imshow("closed1", closed1);
	vector< vector<Point> > contours;
	vector<Point> cont;

	//vector< vector<Point> > approx;
	findContours(closed, contours, CV_RETR_EXTERNAL, CV_CHAIN_CODE);
	cout << "   perimetr " << "      square " << "       rcoeff " << "      H[0] " <<
"      H[1] " << "      H[2] " << "      H[3] " << "      H[4] " << "      H[5] " << "
H[6] " << "   gcenter" << "   mcenter" << "\n";
	for (vector<Point>cont : contours)
	{
		double area = abs(contourArea(cont, true));
		double peri = arcLength(cont, true);
		double round = abs(4 * 3.14159265358979323846 * area / (peri * peri));
```

```
                cout.width(2);
                cout << " " << peri << " ";
                cout.width(2);
                cout << "  " << area << " ";
                cout.width(2);
                cout << "  " << round << "  ";
                Moments moment = moments(cont, false);
                Rect rect = boundingRect(cont);
                Point prect;
                Point pmass;
                prect.x = int(rect.x + rect.width / 2);
                prect.y = int(rect.y + rect.height / 2);
                pmass.x = int(moment.m10 / moment.m00);
                pmass.y = int(moment.m01 / moment.m00);
                double huMoments[7];
                HuMoments(moment, huMoments);
                for (int i = 0; i < 7; i++)
                {
                        huMoments[i] = -1 * copysign(1.0, huMoments[i]) *
log10(abs(huMoments[i]));
                        cout.width(2);
                        cout << " " << huMoments[i] << " ";
                }
                cout << "  " << prect << " ";
                cout << "  " << pmass << " ";
                cout << "\n";


                //--------------ВЫЧИСЛЕНИЯ--------------
                double auf = pow(huMoments[0], 2) + pow(huMoments[1], 2) +
pow(huMoments[2], 2) + pow(huMoments[3], 2) + pow(huMoments[4], 2) + pow(huMoments[5], 2)
+ pow(huMoments[6], 2);
                double delitel = sqrt(auf);

                for (int i = 0; i < 7; i++)
                {
                        huMoments[i] = abs(huMoments[i] / sqrt(auf));
                }

                double masKrug[] = { 0.05281117763,  0.1579581782,  0.1922022284,
0.3153414458, 0.5792261401, 0.4024215814, 0.581749275 };
                double masKvadrat[] = { 0.06177768513,  0.1532395972,  0.2084010604,
0.3032979444, 0.5682376103, 0.4171379592,  0.5834230366 };
                double masTreugolnik[] = { 0.06444418185,  0.190611537,  0.230644544,
0.2940536926, 0.5611722035, 0.4106416851,  0.5799214379 };

                double pervieTriKrug = 0;
                double pervieTriKvadrat = 0;
                double pervieTriTreugolnik = 0;

                for (int i = 0; i < 7; i++)
                {
                        pervieTriKrug += pow(masKrug[i] - huMoments[i], 2);
                        pervieTriKvadrat += pow(masKvadrat[i] - huMoments[i], 2);
                        pervieTriTreugolnik += pow(masTreugolnik[i] - huMoments[i], 2);
                }

                double otklonenieKrug = pervieTriKrug/7;
                double otklonenieKvadrat = pervieTriKvadrat/7;
                double otklonenieTreugolnik = pervieTriTreugolnik/7;
                //--------------ВЫЧИСЛЕНИЯ--------------
                total++;

                double raznica = otklonenieKrug;
```

```
            cout << "\n\n\n  " << otklonenieKrug << "  " << otklonenieKvadrat << "  "
<< otklonenieTreugolnik << "\n\n\n  ";

            if (raznica > otklonenieKvadrat)
                  raznica = otklonenieKvadrat;

            if (raznica > otklonenieTreugolnik)
                  raznica = otklonenieTreugolnik;

            if (raznica == otklonenieKrug) {
                  krug = total;
                  cout << "\n\n" << "The most krug is: " << krug << "\n\n";
            }
            if (raznica == otklonenieKvadrat) {
                  kvadrat = total;
                  cout << "\n\n" << "The most kvadrat is: " << kvadrat << "\n\n";
            }
            if (raznica == otklonenieTreugolnik) {
                  treugolnik = total;
                  cout << "\n\n" << "The most treugolnik is: " << treugolnik << "\n\n";
            }

      }

      namedWindow("example", WINDOW_AUTOSIZE);
      imshow("example", image);
      cout << "\n" << "total objects " << total;
      waitKey(0);
      return 0;
}
```

Результаты работы программы:




D:\На будущее\3 курс 1 семестр\ЦОИ\Курсач\Proba3\x64\Release\Proba3.exe
Choose picture's number 1 - 4
4
perimetr square rcoeff H[0] H[1] H[2] H[3] H[4] H[5] H[6] gcenter mcenter
1.27945e+09 1.85609e+15 0.0142482 -0.701704 0.788972 -0.0597103 -0.0205934 -0.0603042 -0.379095 -1.28565 [50496259, 50528899] [31580196, 11846092]
The most treugolnik is: 1
total objects 1
example
Введите здесь текст для поиска
РУС
12:01
09.12.2020