Краевой конкурс учебно-исследовательских и проектных работ учащихся
«Прикладные вопросы математики»

Прикладные вопросы математики

Распознавание образов

Байдин Дмитрий Алексеевич,
МОУ «Лицей №1» г. Перми, 11 кл.
Анферов Сергей Дмитриевич,
преподаватель информатики
МОУ «Лицей №1» г. Перми

Введение

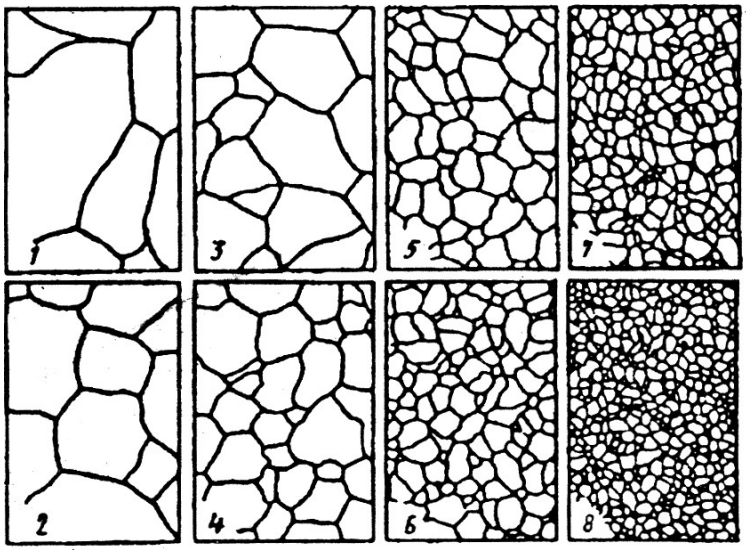
С задачей распознания образов человек сталкивается с самого момента рождения. Все поступающие из окружающего мира сигналы необходимо распознать, отсортировать по степени важности и обработать. Человеческий мозг способен с легкостью распознавать отдельные слова, выделять объекты, отличать их друг от друга. Поступающее из окружающего мира изображение мгновенно дробится на десятки смысловых частей: дерево, куст, человек, птица, солнце, облака. В тоже время, каждая из этих частей состоит из более мелких, таких как: листья, ветки, руки, ноги, глаза. Чем человек руководствуется? Какие алгоритмы, какие процессы происходят в его голове?

Все большее распространение получили системы автоматизированного ввода информации через различные типы, а также цифровые фото- и видеокамеры. При этом по разрешающей способности такие системы ввода вполне приближаются к зрению человека или животных. Тем не менее, возможности интеллектуального анализа изображений с помощью компьютеров оставляют желать лучшего. Создание искусственных систем распознавания образов остаётся сложной теоретической и технической проблемой. Таким образом, успехи по распознаванию букв и цифр в документах и текстах впечатляют, также как и другие значительные достижения по анализу изображений специального вида (например, распознавание треков ядерных частиц, идентификация автомобилей-нарушителей по фотоснимкам, анализ и распознавание сигналов в медицине и геологии).

Но еще не создано универсальных систем распознания, приближенных по уровню к интеллекту человека.

Эта работа посвящена распознаванию образов в применении к материаловедению, для нахождения размера зерна металла, в частности. К разрабатываемому комплексу предъявляются следующие требования:

Способность выделять однотипные объекты, разбивать их на группы и статистически описывать это множество.



Алгоритмы распознавания образов

На данный момент существует несколько известных алгоритмов распознавания. Перечислю некоторые из них:

1 Алгоритм скелетизации.

Это метод распознавания одинарных бинарных образов, основанный на построение скелетов этих образов и выделения из скелетов ребер и узлов. Далее по соотношению ребер, их числу и числу узлов строится таблица соответствия образам. Так, например, скелетом круга будет один узел, скелетом буквы П - три ребра и два узла, причем ребра относятся как 2:2:1. В программировании данный метод имеет несколько возможных реализаций.

2 Нейросетевые структуры.

Направление было очень модным в 60е-70е годы, впоследствии интерес к ним немного поубавился, т.к. солидное число нейронов требует солидные вычислительные мощности, которые обычно отсутствуют на простеньких мобильных платформах. Однако надо иметь в виду того, что нейросети иногда дают весьма интересные результаты, за счет своей нелинейной структуры, более того некоторые нейросети способны распознавать образы инвариантные относительно поворота без какой либо внешней предобработки. Так, например сети на основе неокогнейтронов способны выделять некоторые характерные черты образов, и распознавать их как бы образы не были повернуты

3 Инвариантрые числа.

Из геометрии образов можно выделить некоторые числа, инвариантные относительно размера и поворота образов, далее можно составить таблицу соответствия этих чисел конкретному образу(почти как в алгоритме скелетезации). Примеры инвариантных чисел - число эйллера, экцентриситет, ориентация (в смысле расположения главной оси инерции относительно чего-нибудь тоже инвариантного).

4 Поточечное процентное сравнение с эталоном.

Здесь должна быть некоторая предобработка, для получения инвариантности относительно размера и положения, затем осуществляется сравнение с заготовленной базой эталонов изображений - если совпадение больше чем какая-то отметка, то считаем образ распознанным.

Содержательная постановка задачи

Целью данной работы является создание прикладной программы способной выделять и идентифицировать объекты в двумерном пространстве. Программа будет выделять на картинке область однородной цветовой гаммы, который в будущем будет называться объект. С малым числом объектов задача должна решаться в реальном времени. Алгоритм распознавания должен быть инвариантен относительно размера образов и их положения (поворота) и зависеть лишь от формы объекта. Таким образом, программе в качестве входных данных будет предоставлено двумерное изображение, выходными же данными будет статистическое описание множества объектов, а также геометрические характеристики каждого объекта.

Концептуальная постановка задачи

При рассмотрении данной проблемы были введены следующие определения и допущения:

Объектом в этой задаче называется непрерывная область однородного цвета (в области одного объекта цвет резко не меняются), с четкими границами.

Рассматривается что, объекты не накладываются друг на друга, то есть мы рассматриваем лишь однослойное изображение. Между границами проведены четкие границы, так как при смазанных границах программа может пропустить их и посчитать несколько объектов в качестве одного. При рассмотрении задания было введено предположение, что на изображении нет бликов, так как в противном случае это значительно усложняет работу программы. Так же количество пикселей принадлежащих к объекту должно быть больше некой константы, определяемой пользователем, если количество пикселей меньше константы, то объект становится мусором.

Мусором называется объект имеющий количество пикселей меньше заданной величины.

При решении задачи о нахождении центра масс, каждая точка объекта имеет одинаковую массу.

Математическая постановка задачи

Перед началом решения задачи распознавания образов к изображению были применены графические фильтры, в частности, размытие по Гауссу было реализовано в программе.

При решении задачи для оценки похожести (цветовой однородности области) была использована функция сравнения цветов.

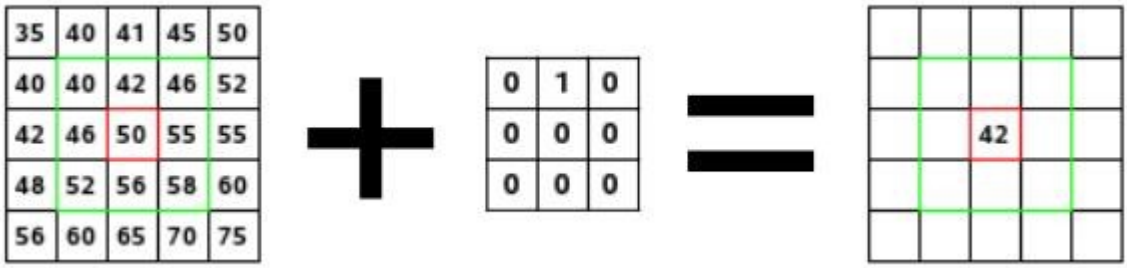
Для последующей классификации полученных объектов было необходимо каким-то образом охарактеризовать их, для этой цели служат формулы описания формы объекта.

Графические фильтры:

Для повышения цветовой однородности изображения используется следующий алгоритм:

Преобразования происходит следующим образом. Каждый элемент исходного изображения умножается на центральное значение матрицы ядра. Кроме этого на соответствующие значения умножаются окружающие его элементы (при размере ядра 3x3 их будет 8), после чего результаты суммируются и принимаются как преобразованное значение.

Вот простой графический пример:



(40*0)+(42*1)+(46*0)+(46*0)+(50*0)+(55*0)+(52*0)+(56*0)+(58*0) = 42

Для нормализации результата используются дополнительные переменные: div (делитель) (div = 0 выставлять нельзя!) и offset (коэффициент). Они работают очень просто: результат преобразования делится на div и к нему прибавляется offset. [1]

Сравнение цвета:

Сравнение цвета происходит в пространстве Lab, так как в этом случае используется система представления цвета приближенная к человеческому ощущению. Для этого цвет из RGB был переведен в Lab палитру по следующим формулам:

Шаг 1. Переход из RGB в вещественной нормировке в коническое пространство LMS, т.к. Lab - его модификация:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 1.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Шаг 2. Перевод LMS в Lab:

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0.5774 & 0 & 0 \\ 0 & 0.4082 & 0 \\ 0 & 0 & 0.7071 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \lg L \\ \lg M \\ \lg S \end{bmatrix}$$ [2]

Разница цвета считается по формуле:

$$Eps = \sqrt{(L_0 - L_1)^2 + (a_0 - a_1)^2 + (b_0 - b_1)^2}$$

Описание формы объекта:

При вычислении морфометрических признаков объекта используются понятия механики твердого тела. В частности, это относится к длинам осей инерции объекта. Направления в теле, совпадающие с полуосями эллипсоида инерции, называют главными осями инерции.

Пусть N - количество пикселей, относящихся к объекту. Все множество пикселей p(x, y), относящихся к объекту, обозначим Q. Тогда координаты центра масс объекта вычисляются как:

$$x_c = \frac{1}{N} \sum_{p(x,y) \in \Omega} x \qquad y_c = \frac{1}{N} \sum_{p(x,y) \in \Omega} y$$

Вычислим несколько вспомогательных величин:

$$U_x = \frac{1}{12} + \frac{1}{N} \sum_{p(x,y) \in \Omega} (x - x_c)^2$$

$$U_y = \frac{1}{12} + \frac{1}{N} \sum_{p(x,y) \in \Omega} (y - y_c)^2$$

$$C = \sqrt{(U_x - U_y)^2 + 4 \cdot U_{xy}{}^2}$$

Тогда длины максимальной и минимальной осей инерции вычисляются как:

$$A_{\max} = 2\sqrt{2} \cdot \sqrt{U_x + U_y + C}$$

$$A_{\min} = 2\sqrt{2} \cdot \sqrt{U_x + U_y - C}$$

Длины главных осей инерции используются для вычисления эксцентриситета и ориентации объекта.

Эксцентриситет определяется с помощью соотношения:

$$E = \frac{2 \cdot \sqrt{(0.5 \cdot A_{max})^2 - (0.5 \cdot A_{min})^2}}{A_{max}}$$

Ориентация определяется как угол в градусах между максимальной осью инерции и осью X. Если $U_y > U_x$, то ориентация O вычисляется с помощью формулы:

$$O = \frac{180}{\pi} \cdot arctg\left(\frac{U_y - U_x + C}{2 \cdot U_{xy}}\right)$$

в противном случае O вычисляется как:

$$O = \frac{180}{\pi} \cdot arctg\left(\frac{2 \cdot U_{xy}}{U_x - U_y + C}\right)$$

Пусть, Area – площадь объекта, то есть количество пикселей, FilledArea – полная площадь, количество пикселей объекта плюс площадь дыр в объекте, RectangleArea – площадь прямоугольника, в который вписан объект, ConvexArea – площадь выпуклого многоугольника, в который вписан объект, тогда коэффициент заполнения – Extent, Convex – коэффициент выпуклости и коэффициент дырчатости – Solidity считаются по следующим формулам:

$$\text{Extent} = \frac{\text{Area}}{\text{FilledArea}}$$

$$Convex = \frac{Area}{ConvexArea}$$

$$\text{Solidity} = \frac{\text{Area}}{\text{RectangleArea}}$$ [3]

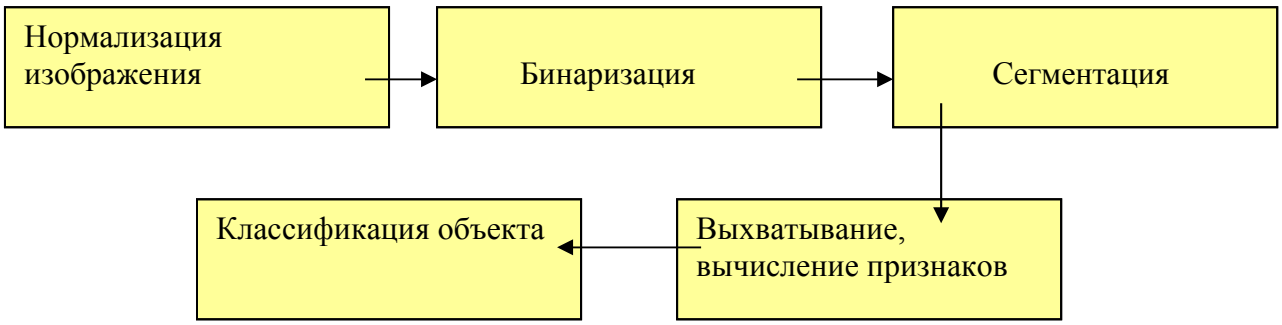
Алгоритм поворота изображения, пусть:
O - центр поворота,
M - точка исходного изображения.

Для каждой точки M нужно найти угол α между отрезком OM и горизонталью и длину r отрезка OM. Теперь, чтобы повернуть изображение на угол β , нужно каждой точке M присвоить цвет точки исходного изображения с координатами x,y, где:

$$x = x_0 + r * \cos(\alpha + \beta)$$
$$y = y_0 + r * \sin(\alpha + \beta)$$

x_0, y_0 - центр поворота,
r - длина отрезка OM [4]

Описание метода решения



Нормализация изображения

В некоторых случаях исходное изображение по каким-то причинам не подходит для распознавания, например: большое количество «мусора» на изображении, наличие бликов, нечетких границ или шумов.

Для повышения однородности цвета на изображении применяется размытие по Гауссу, которое осуществляется с помощью данной матрицы:

$$\begin{Bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{Bmatrix}$$ Div=9, Offset=0

Это позволяет усреднить цветовую разницу на изображения, убрать мусор, но при этом размыть границы. Это приводит к уменьшению количества объектов, и, в некоторых случаях, более продуктивной работе программы. Для усиления эффекта необходимо произвести операцию несколько раз.

Бинаризация

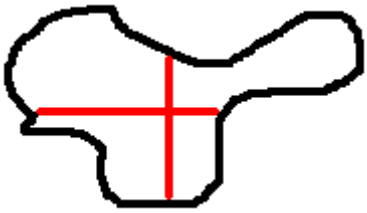
Цифровая обработка изображения. Потребность в этом шаге вызвана тем, что компьютер работает с числовой информацией намного быстрее, чем с графической. Программа преобразует цвет в массив чисел.

Так как комплекс работает в цветовом пространстве Lab, на этом шаге так же происходит перевод из RGB→Lab.

Сегментация

Далее необходимо разбить изображение на отдельные объекты. «Разбиение» начинается из левого верхнего угла и впоследствии обходит все изображение. Каждая точка проверяется на похожесть(сравнение по цвету)с соседними верхним, нижним, левым и правым пикселями. В случае успешной проверки пиксель присоединяется к текущему объекту. Далее проверяются следующие точки, пока луч не упрется в пиксель, отличающийся от текущего более чем заданная максимально допустимая разница, либо в границу изображения. Далее такая же процедура выполняется для всех точек, присоединенных к объекту на предыдущем шаге, причем ранее добавленные точки, заново не присваиваются. Процедура заканчивается, когда текущий объект больше не растет за счет добавления новых точек. Затем следует проверка на размер, является ли объект мусором. В случае успешного прохода проверки объект добавляется в массив объектов.

Начало



Шаг 1

Шаг 2

Конец

Определение формы

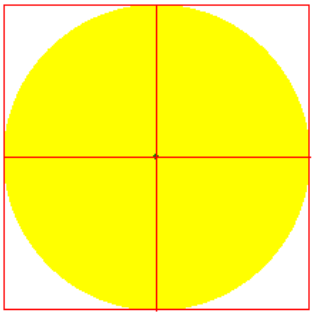
Чтобы однозначно определить форму объекта были использованы различные формулы.

1. Коэффициент выпуклости

Чтобы определить коэффициент заполнения, программа находит ориентацию объекта, затем поворачивает на угол наклона относительно центра масс. Находит экстремумы – крайние точки объекта. Описывает прямоугольник вокруг объекта. Находит отношение площади объекта к площади прямоугольника.

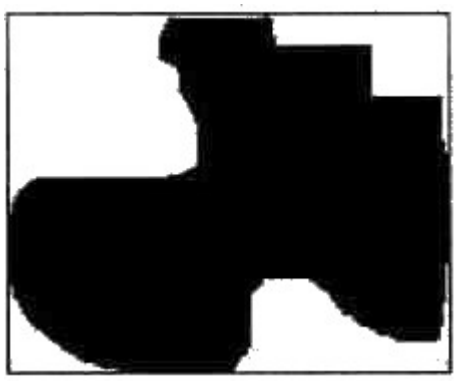
2. Эксцентриситет

3. Чтобы найти эксцентриситет, программа определяет положение центра масс, вычисляет несколько вспомогательных величин длину большой и малой оси инерции в частности.



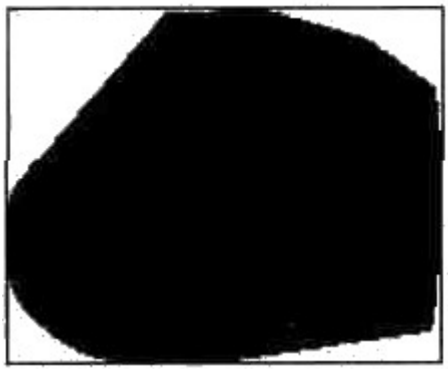
4. Коэффициент дырчатости

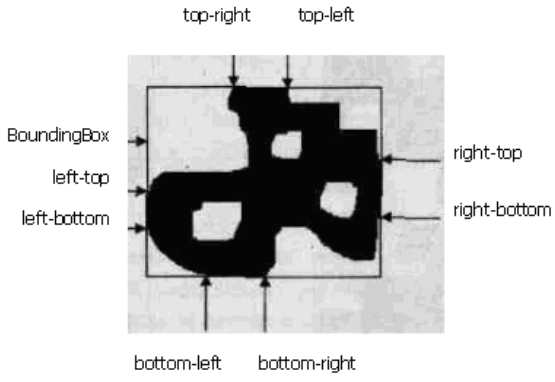
Чтобы найти коэффициент заполнения, программа находит экстремумы, описывает прямоугольник вокруг объекта, далее запускается алгоритм, похожий на алгоритм сегментации, но найденная площадь считается дыркой лишь в том случае, если луч не разу не пересекал описанный прямоугольник.



5. Коэффициент заполнения

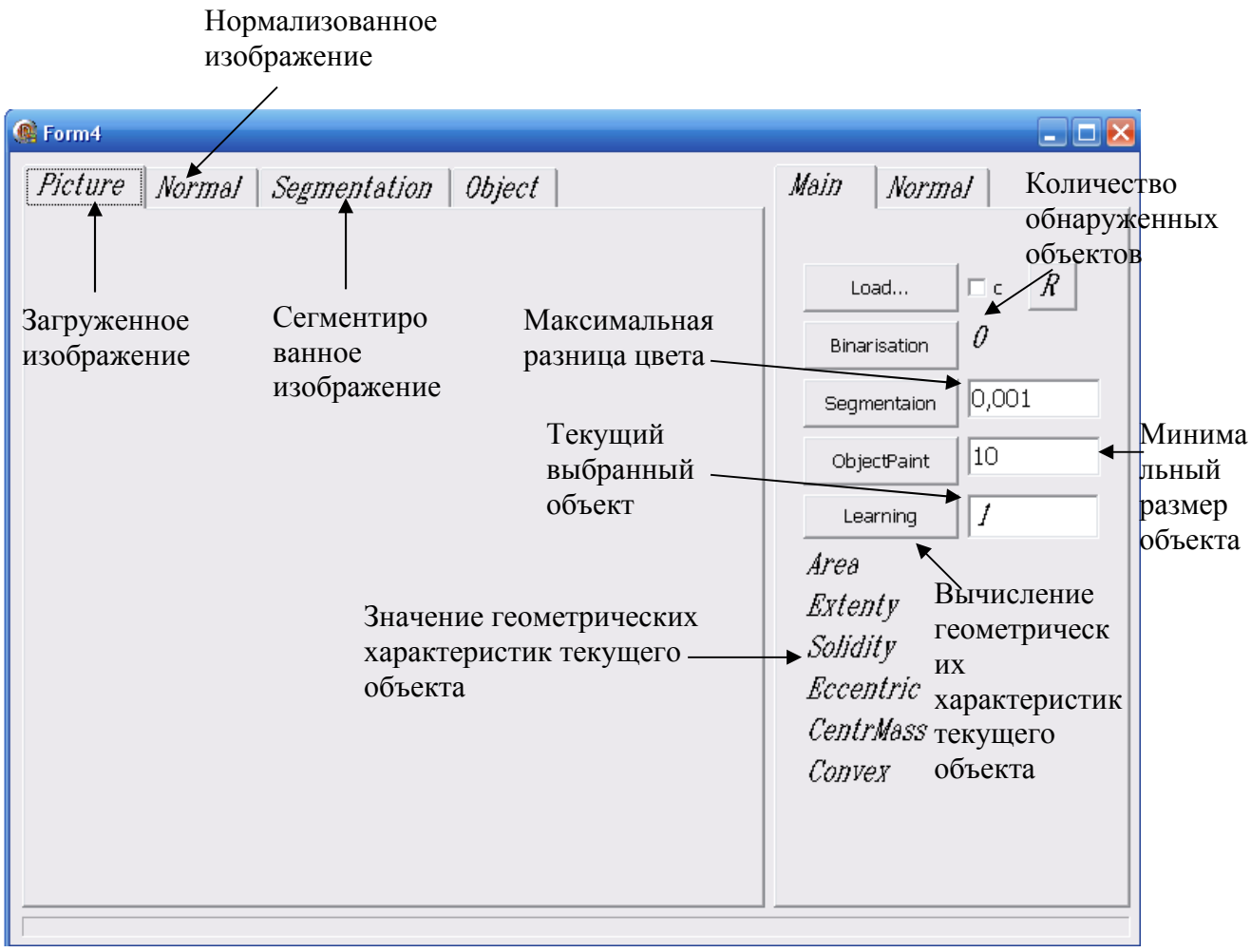
Чтобы найти коэффициент заполнения, необходимо произвести процедуру аналогичную процедуре нахождения коэффициента дырчатости, стой лишь разницей, что перед ее началом находятся крайние экстремумы (верхний левый, верхний правый, нижний левый и т.д.) и соединяются линиями, через которые не может пройти луч сегментации.





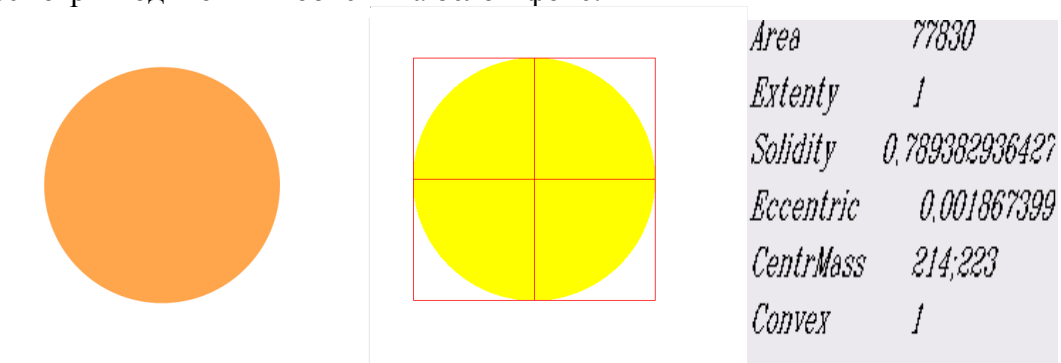
Эксперименты

Вот внешний вид разработанной программы:



Для проверки правильности работы данной программы были проведем следующие эксперименты:

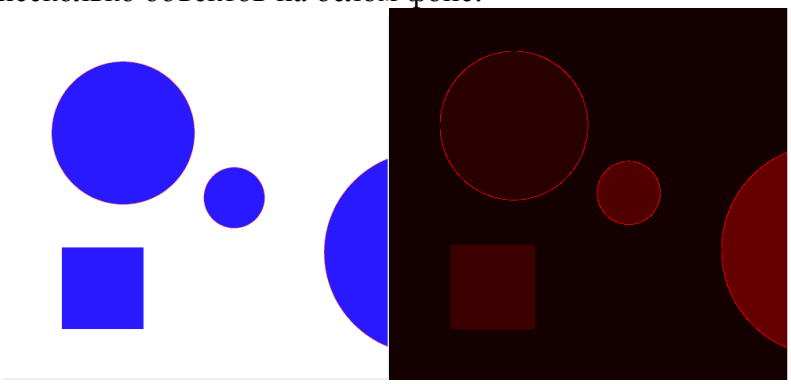
Рассмотрим одиночный объект на белом фоне:



Исходное изображение Распознанное изображение

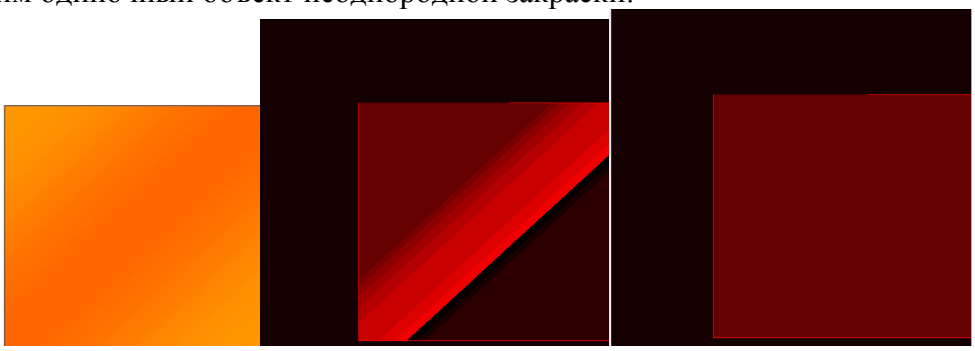
Распознавание произошло.

Рассмотрим несколько объектов на белом фоне:



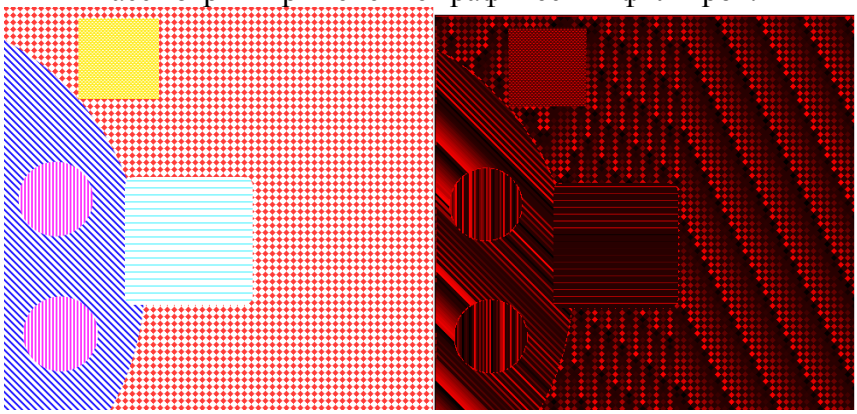
Исходное изображение, сегментированное изображение, объекты раскрашены в различные оттенки красного в зависимости от метки (принадлежности к объекту) Распознавание произошло.

Рассмотрим одиночный объект неоднородной закраски:

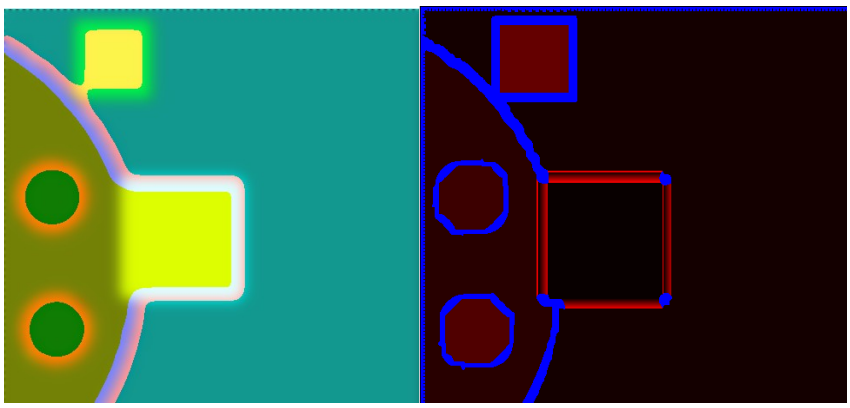


Распознавание произошло. В зависимости от чувствительности распознания несколько способов «разбиения» объекта.

Рассмотрим применение графических фильтров:

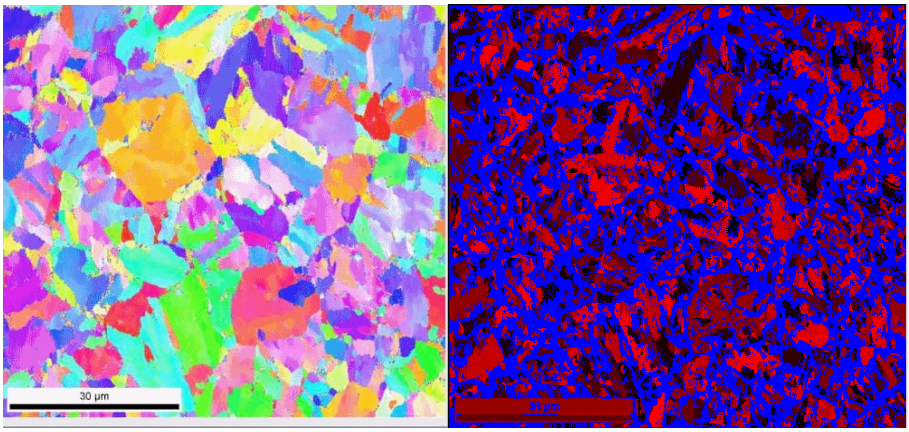


Исходное изображение, состоит из большого числа объектов (общее количество найденных объектов - 2708), чтобы уменьшить количество объектов, применим графические фильтры.



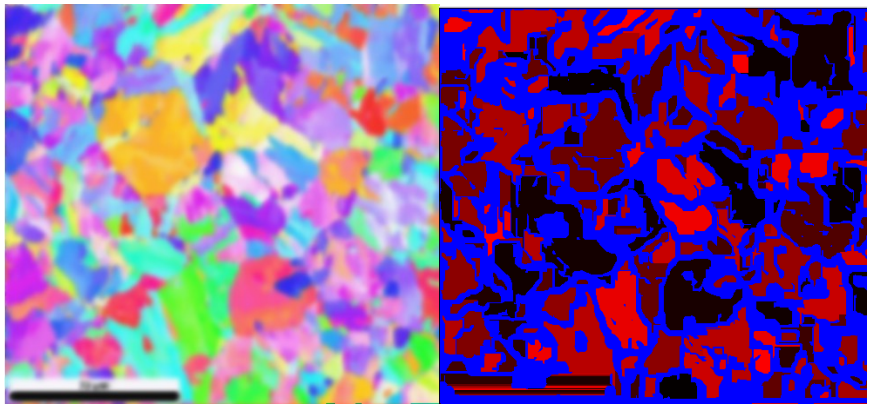
Количество объектов уменьшилось до 7. Синие пиксели – мусор.

В целом распознавание прошло успешно, но вышеприведенные примеры не имеют отношение к реальности. Рассмотрим несколько реальных фотографий. Примеры взяты отсюда.

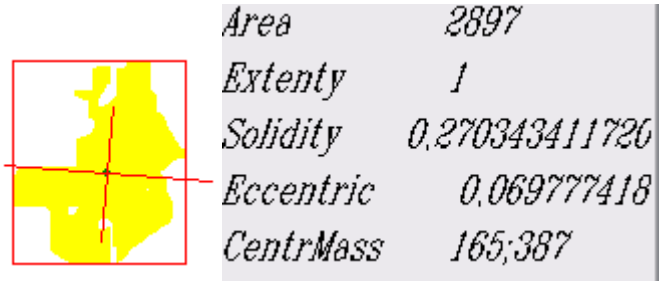


Микротекстура стали мартенситного класса

Найдено 2209 объектов. Распознавание прошло успешно. Теперь попробуем изменить максимальную разницу в цвете, минимально-допустимое количество пикселей и применить фильтры.



Распознавание прошло успешно, найдено 50 объектов.



Определение геометрических характеристик одного из объектов.

Результаты

Следует отметить, что цель данной работы достигнута, несмотря на отсутствие автономности программы. Тем не менее, данная работа может, как послужить основой для более совершенного комплекса по распознаванию объектов, так и быть использована в нынешнем виде для задач, подразумевающих выделение объектов, подсчет их количества, а также геометрических характеристик, таких как центр масс, главные оси инерции и т.д. Разработанная программа способна находить и описывать объекты на изображении. Однако для продуктивной работы программы необходимо вмешательство человека, для настройки параметров поиска объектов. Стоит отметить, что для следующих изображений подобной композиции коэффициенты сохраняются. Также изображения с нечеткими границами или бликами требуют предварительной обработки вне программы.

Планируется усовершенствование программы следующими способами:

1. Создание каталога образов, с уже вычисленными характеристиками, для последующего определения их на изображении;
2. Увеличение количества графических фильтров; для чего?
3. Усовершенствование алгоритма сегментации путем разбиения сегментации на несколько уровней.

Список литературы

1. Курсы лаборатории компьютерной графики// [Электронный ресурс]. – URL: http://www.graphicon.ru/oldgr/courses/cg/assigns/2005/hw2/index.html (дата обращения 4.03.2010)
2. Распознавание образов мобильным роботом// [Электронный ресурс]. – URL: http://www.ampersant.ru/glaz/ (дата обращения 4.03.2010)
3. Вычисление признаков объектов// [Электронный ресурс]. – URL: http://matlab.exponenta.ru/imageprocess/book3/14/imfeature.php (дата обращения 4.03.2010)
4. Графические фильтры на основе матрицы скручивания // [Электронный ресурс]. – URL: //http://habrahabr.ru/blogs/webdev/43895/ (дата обращения 4.03.2010)
5. Алгоритм поворота изображения// // [Электронный ресурс]. – URL: http://www.delphisources.ru/pages/faq/base/rotate_image.html (дата обращения 4.03.2010)