МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

СОВРЕМЕННЫЕ СИСТЕМЫ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Методические рекомендации к лабораторным работам для студентов специальности 1-40 80 02 «Системный анализ, управление и обработка информации» очной и заочной форм обучения

Могилев 2020

УДК 004.9
ББК 32.97
С23

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «02» сентября 2020 г., протокол № 1

Составитель канд. техн. наук, доц. А. В. Шилов

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации к лабораторным работам предназначены для студентов специальности 1-40 80 02 «Системный анализ, управление и обработка информации».

Учебно-методическое издание

СОВРЕМЕННЫЕ СИСТЕМЫ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Ответственный за выпуск	А. И. Якимов
Корректор	А. А. Подошевко
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, Могилев.

Содержание

Введение

Системы технического зрения призваны и во многих случаях уже решают задачи по дополнению или даже замене человека в областях деятельности, связанных со сбором и анализом зрительной информации. Уровень их использования в прикладных областях является одним из наиболее ярких и наглядных интегральных показателей уровня развития высоких технологий в самых различных отраслях промышленности.

Целью учебной дисциплины состоит в формировании у обучающихся знаний, умений, навыков, необходимых при построении современных систем компьютерного зрения, осваивать новые и применять существующие алгоритмы обработки, распознавания и отслеживания объектов на изображениях и в видеопотоке.

Целью данных методических рекомендаций является приобретение обучающимися практических навыков в области современных систем компьютерного зрения, понимания принципов аппаратного и программного построения таких систем, умения пользоваться средствами для получения и обработки двумерных и трехмерных изображений различных объектов, знания основных библиотек для работы с изображениями, их обработки и алгоритмов распознавания, выделение особенностей и сопоставление изображений, отслеживание движения.

В обработке изображений главное внимание уделяется преобразованию одних изображений в другие, в то время как распознавание изображений сосредоточено на формировании выводов на основе изображений и явном получении необходимых для этого описаний сцен

Задачей компьютерного зрения заключается в формировании представления объектов и сцен реального мира на основе анализа изображений, полученных с помощью датчиков. Для решения поставленной задачи применяют методы математического анализа, теории вероятности и аналитической геометрии. Компьютерное зрение, объединяя эти методы, является самостоятельной научной дисциплиной и реализуется в программном обеспечении современных математических пакетов и библиотеках обработки изображений в различных областях для создания описаний сцен по изображениям.

1 Обработка демонстрационного изображения, базовые операции с изображениями

Цель работы: обработать демонстрационное изображение, изучить базовые операции с изображениями: коррекции яркости, операция свертки, фильтр Гаусса, медианный фильтр, повышение резкости, растяжение, сжатие, вращение и деформация изображения.

Изображение служит для представления информации в визуальном виде. Эффективность восприятия этой информации человеком зависит от многих факторов. На современном этапе развитие технической и медицинской диагностики неразрывно связано с визуализацией внутренних структур объекта [1]. Существует много различных видов визуализации. Возникают новые методы, но они не заменяют уже существующие, а лишь дополняют их. Разные методы визуализации основываются на разнообразных физических взаимодействиях электромагнитного излучения с материалами, средами, биотканями и, как следствие, обеспечивают измерение разных физических свойств этих объектов.

Основные методы получения изображений, которые представляют интерес для технической и медицинской диагностики – это рентгенография, компьютерная томография, получение изображений с помощью короткоживущих радиоизотопов, ультразвуковая диагностика, магнитно-резонансная томография, формирование изображений дистанционного зондирования земли, мультиспектральных изображений, гиперспектральных изображений.

1.1 Геометрические преобразования изображения

Геометрические преобразования изображений рассмотрим на примере преобразования фотографии.

К наиболее распространенным функциям геометрических преобразований относится кадрирование изображений (imcrop), изменение размеров (imresize) и поворот изображения (imrotate).

Суть кадрирования состоит в том, что функция imcrop позволяет с помощью мыши в интерактивном режиме вырезать часть изображения и поместить ее в новое окно просмотра (рисунок 1.1).

а) *б*)

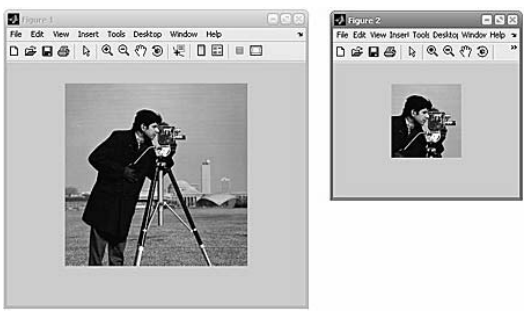


Рисунок 1.1 – Изображение исходное (*а*) и после применения операции кадрирование (*б*)

Функция изменения размеров изображения imresize позволяет, используя специальные методы интерполяции, изменять размер любого типа изображения (рисунок 1.2).

```
L=imread('cameraman.tif');
imshow(L);
imcrop;
```

а) *б*)

Рисунок 1.2 – Изображение исходное (*а*) и после изменения размеров изображения (*б*)

В пакете Image Processing Toolbox существует функция imrotate, которая осуществляет поворот изображения на заданный угол (рисунок 1.3).

```
L1=imrotate(L,35,'bicubic');
figure,imshow(L1)
```

а) *б*)

Рисунок 1.3 – Изображение исходное (*а*) и после поворота (*б*)

Таким образом, приведенные выше функции позволяют поворачивать, вырезать части, масштабировать, т. е. работать с целым массивом изображения.

1.2 Преобразования изображения: яркость и контраст

Обработка изображений применяется во многих сферах деятельности человека. Развитие информационных технологий способствует повышению качественного уровня анализа данных. Также прогресс компьютерной техники влияет на быстродействие и достоверность такой обработки. Приведем

несколько наглядных примеров из тех областей, где применяется обработка изображений и, в частности, улучшение яркости и контрастности изображения падающего снега.

Сначала считываем файл snowflakes.png, который представляет собой изображение хлопьев снега (рисунок 1.4).

```
I=imread('snowflakes.png');
figure, imshow(I)
```

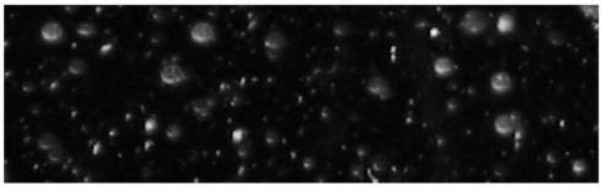


Рисунок 1.4 – Улучшение контрастности изображения

Эффективность решения любой задачи по обработке изображений, в большой мере, зависит от качества исходных данных. В данном случае, когда необходимо идентифицировать объекты и анализировать их размер, то высокое качество исходного изображения является одним из необходимых условий. Для решения задачи повышения контрастности изображения (рисунок 1.5) в системе Matlab можно использовать функцию adapthisteq. Эта функция реализует контрастно-ограниченную адаптивную эквализацию (выравнивание) гистограммы. Для коррекции яркостей изображения используется функция imadjust, которая позволяет управлять диапазоном яркостей изображения.

```
claheI=adapthisteq(I, 'Divisions', [10 10]);
claheI=imadjust(claheI);
imshow(claheI);
```

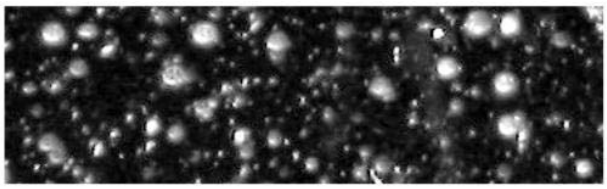


Рисунок 1.5 – Определение яркости поверхности на улучшенном изображении

При решении задач по обработке изображений, например, в гранулометрии результат измерения интенсивностей элементов исследуемого изображения представляется в виде функции. Эта функция привязана к размерам исходного изображения. Каждому объекту изображения присваиваются определенные параметры, основной из которых – размер. На основании этих параметров производится построение гистограммы распределения объектов по определенным признакам. Вторым параметром, на основании которого осуществляется распределение, является интенсивность (яркость) объектов изображения.

```
    for counter=0:22
    remain=imopen(claheI, strel('disk', counter));
    intensity_area(counter+1)=sum(remain(:));
    end
    figure,plot(intensity_area, 'm - *'), grid on;
    title('Функция зависимости суммы пикселей раскрытого изображения от
их радиуса');
    xlabel('радиус раскрытия (в пикселях)');
    ylabel('значение суммы пикселей раскрытых объектов
(интенсивность)');
```

Распределение объектов изображения, полученное в результате гранулометрической обработки, характеризуется также рядом параметров. Одним из них является первая производная, которая вычисляется при помощи функции diff (рисунок 1.6).

```
intensity_area_prime=diff(intensity_area);
plot(intensity_area_prime, 'm-*'), grid on;
title('Granulometry (Size Distribution) of Snowflakes');
set(gca, 'xtick', [0 2 4 6 8 10 12 14 16 18 20 22]);
xlabel('радиус снежинок (в пикселях)');
ylabel('зависимость суммы пикселей снежинок от радиуса');
```

Отметим минимум, который встречается на полученном графе. Этот минимум характеризует наиболее мелкие объекты (снежинки) на изображении (рисунок 1.7). Отрицательная минимальная точка свидетельствует о повышении кумулятивной интенсивности снежинок в зависимости от их радиуса.

```
open5=imopen(claheI, strel('disk', 5));
open6=imopen(claheI, strel('disk', 6));
rad5=imsubtract(open5, open6);
imshow(rad5, []);
```

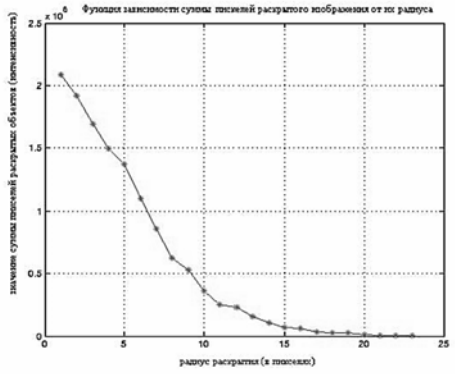


Рисунок 1.6 – Вычисление первой производной распределения

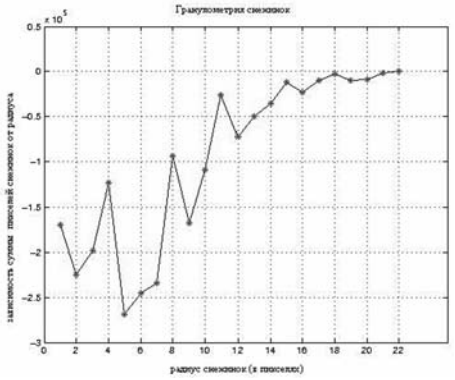


Рисунок 1.7 – Формирование объектов (снежинок) с учетом вычисленного радиуса

Результат обработки представлен на рисунке 1.8.

Рисунок 1.8 – Изображение после обработки

Практическое задание

1 Разработать программу для геометрического преобразования произвольного изображения в среде Matlab.

2 Разработать программу, которая улучшает контраст и яркость произвольного изображения в среде Matlab.

2 Представление изображений

Цель работы: подавление шумов на изображениях.

Решению задачи фильтрации шумов посвящено очень много работ, существуют различные методы и алгоритмы. В этой работе рассмотрим только некоторые подходы и возможности их реализации в системе Matlab.

Для этого разобъем процесс разработки на несколько шагов.

1 *Считывание исходного изображения.*

Считаем исходное изображение (рисунок 2.1) из файла в рабочее пространство Matlab и отобразим его на экране монитора.

```
L=imread('kinder.bmp');
figure, imshow(L);
```

2 *Формирование зашумленных изображений.*

В системе Matlab (Image Processing Toolbox) существует возможность формирования и наложения на изображение трех типов шумов. Для этого используется встроенная функция imnoise, которая предназначена, в основном, для создания тестовых изображений, используемых при выборе и исследовании методов фильтрации шума. Рассмотрим несколько примеров наложения шума на изображения.

2.1 Добавление к изображению импульсного шума (по умолчанию плотность шума равна доле искаженных пикселей (рисунок 2.2)):

```
L2=imnoise(L,'salt&pepper', 0.05);
figure, imshow(L2);
```

2.2 Добавление к изображению гауссовского белого шума (по умолчанию математическое ожидание равно 0, а дисперсия – 0,01 (рисунок 2.3)):

```
L1=imnoise(L,'gaussian');
figure, imshow(L1);
```

Рисунок 2.1 – Исходное изображение

Рисунок 2.2 – Зашумленное изображение (импульсный шум)

2.3 Добавление к изображению мультипликативного шума (по умолчанию математическое ожидание равно 0, а дисперсия 0,04 (рисунок 2.4)):

```
L3=imnoise(L,'speckle',0.04);
figure, imshow(L3);
```

Рисунок 2.3 – Зашумленное изображение (гауссовский шум)

Рисунок 2.4 – Зашумленное изображение (мультипликативный шум)

3 *Использование медианного фильтра для устранения импульсного шума.*

Одним из эффективных путей устранения импульсных шумов на изображении является применение медианного фильтра. Наиболее эффективным вариантом является реализация в виде скользящей апертуры.

```
for i=1+n1:N+n1;
    disp(i)
    for j=1+m1:M+m1;
    if j==1+m1;
                        D=0;
                        for a=-n1:n1;
                        for b=-m1:m1;
                           D(n1+1+a,m1+1+b)=Lr(i+a,j+b);
                        end;
                        end;
                end;
            if j>1+m1;
             for a=-n1:n1;
               D(n1+1+a,m+1)=Lr(i+a,j+m1);
             end;
              D=D(1:n,2:m+1);
```

```
            end;
            Lvyh(i,j)=median(D(:));
end;
end;
Lvyh=Lvyh(n1+1:N+n1, m1+1:M+m1);
figure, imshow(Lvyh);
```

Для наглядного сравнения приведем три изображения вместе: исходное, зашумленное и восстановленное (рисунок 2.5).

а) *б*) *в*)

а – исходное изображение; *б* – зашумленное изображение (импульсный шум); *в* – восстановленное изображение

Рисунок 2.5 – Восстановление изображения, искаженного импульсным шумом, с применением метода медианной фильтрации

Восстановленное изображение лишь незначительно отличается от исходного изображения и значительно лучше, с точки зрения визуального восприятия, зашумленного изображения.

4 *Подавление шумовой составляющей с использованием операции сглаживания.*

Существует класс изображений, для которых подавление шумовой составляющей возможно реализовать с помощью операции сглаживания (метод низкочастотной пространственной фильтрации). Этот подход может применяться к обработке изображений, содержащих области большой площади с одинаковым уровнем яркости. Отметим, что уровень шумовой составляющей должен быть относительно небольшим (рисунок 2.6).

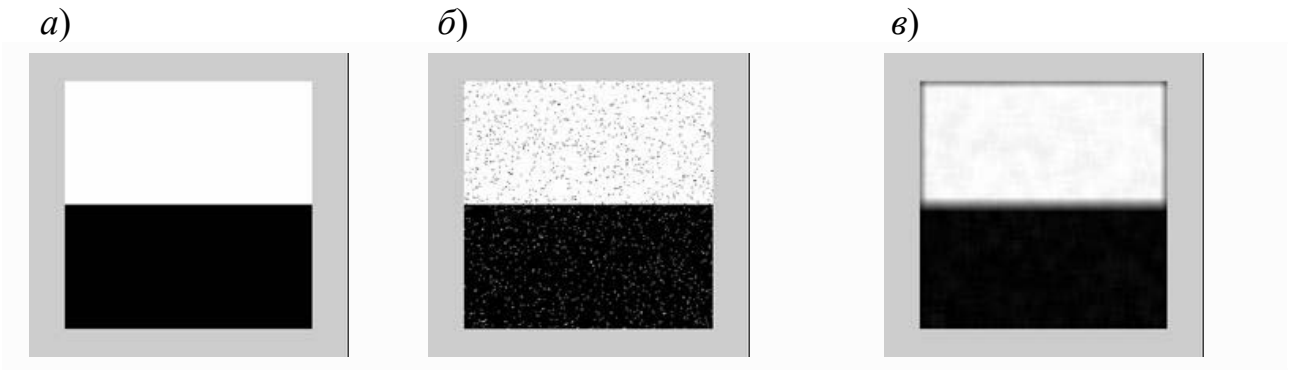
```
F=ones(n,m); % n и m размерность скользящей апертуры
Lser=filter2(F,Lroshyrena,'same')/(n*m);
```

Недостаток этого метода, в отличие от метода медианной фильтрации, состоит в том, что он приводит к размыванию границ объектов изображения.

5 *Пороговый метод подавления шумов.*

Элементы изображения, которые были искажены шумом, заметно отличаются от соседних элементов. Это свойство легло в основу многих методов подавления шума, наиболее простой из которых, так называемый пороговый

метод. При использовании этого метода последовательно проверяют яркости всех элементов изображения.



а – исходное изображение; *б* – зашумленное изображение (импульсный шум); *в* – восстановленное изображение

Рисунок 2.6 – Восстановление изображения, искаженного импульсным шумом с применением операции сглаживания

Если яркость данного элемента превышает среднюю яркость локальной окрестности, тогда яркость данного элемента заменяется на среднюю яркость окрестности (рисунок 2.7).

```
  for i=1+n1:N+n1;
    disp(i)
    for j=1+m1:M+m1;
    if j==1+m1;
                          D=0;
                          for a=-n1:n1;
                          for b=-m1:m1;
                             D(n1+1+a,m1+1+b)=Lr(i+a,j+b);
                          end;
                          end;
                  end;
             if j>1+m1;
              for a=-n1:n1;
                D(n1+1+a,m+1)=Lr(i+a,j+m1);
              end;
               D=D(1:n,2:m+1);
             end;
             LS=mean(mean(D));
             if abs(Lr(i,j)-LS)>10/255; % Установка порога
                Lvyh(i,j)=LS;
              else
                Lvyh(i,j)=Lr(i,j);
             end;
end;
end;
Lvyh=Lvyh(n1+1:N+n1,m1+1:M+m1,:);
figure, imshow(Lvyh);
```

а) *б*) *в*)

а – исходное изображение; *б* – зашумленное изображение (импульсный шум); *в* – восстановленное изображение

Рисунок 2.7 – Восстановление изображения, искаженного импульсным шумом, с применением порогового метода подавления шумов

6 *Низкочастотная фильтрация с использованием шумоподавляющих масок.*

В шаге 4 было рассмотрено применение операции сглаживания для устранения шума. Рассмотрим примеры низкочастотной фильтрации с использованием других шумоподавляющих масок. Это могут быть следующие маски:

– маска 1

$$F = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix};$$

– маска 2

$$F = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Маски для подавления шума представлены в виде нормированного массива для получения единичного коэффициента передачи, чтобы при подавлении шума не было искажений средней яркости. На рисунках 2.8–2.11 представлены результаты обработки зашумленного изображения масками 1 и 2.

```
F=(1/10)*[1 1 1; 1 2 1; 1 1 1];
Lvyh=filter2(F,L,'same')/(3*3);
figure, imshow(Lvyh);
F=(1/16)*[1 2 1; 2 4 2; 1 2 1];
Lvyh=filter2(F,L,'same')/(3*3);
figure, imshow(Lvyh);
```

Это были примеры подавления импульсных шумов. Рассмотрим аналогичные примеры подавления гауссовского и мультипликативного шумов.

Рисунок 2.8 – Результат восстановления зашумленного импульсным шумом изображения с применением маски 1

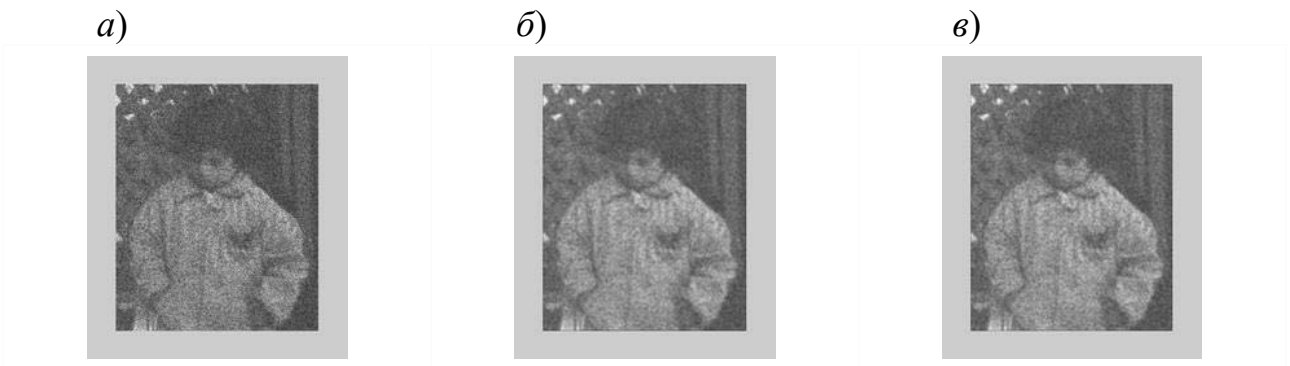
Рисунок 2.9 – Результат восстановления зашумленного импульсным шумом изображения с применением маски 2

а) *б*) *в*)

а – зашумленное гауссовским шумом изображение; *б* – восстановленное изображение с применением маски 1; *в* – восстановленное изображение с применением маски 2

Рисунок 2.10 – Результат восстановления зашумленного гауссовским шумом изображения с применением масок 1 и 2

а) *б*) *в*)



а – зашумленное мультипликативным шумом изображение; *б* – восстановленное изображение с применением маски 1; *в* – восстановленное изображение с применением маски 2

Рисунок 2.11 – Результат восстановления зашумленного мультипликативным шумом изображения с применением масок 1 и 2

Отметим, что универсальных методов нет и к обработке каждого изображения следует подходить индивидуально. Если речь идет о медианной и низкочастотной фильтрации, то качество обработки во многом зависит от удачного выбора размеров локальной апертуры [2–5].

Практическое задание

Разработать программу для обработки изображения путем подавления шумов.

3 Выделение особенностей и сопоставление изображений. Выравнивание освещенности

Цель работы: изучение технологий распознавания объектов. Изучение сопоставления изображений.

Существуют различные подходы к решению задачи распознавания объектов. В этом материале будут рассмотрены некоторые методы структурного распознавания. Суть структурного распознавания состоит в представлении интересующих объектов в виде символьных строк, деревьев, графов и других дескрипторов. Также важным моментом является формирование критериев распознавания на основе выбранного описания объектов.

В отличие от других подходов, особенностью структурного метода распознавания является то, что он работает только с информацией, представленной в символьном виде. Следует отметить, что система Matlab имеет целый ряд специализированных функций для обработки символьных строк. Однако, кроме различных функций обработки, для решения задач распознавания необходимо иметь меру схожести символьных строк, которая бы работала аналогично формулам вычисления расстояний между *n*-мерными векторами.

3.1 Определение меры схожести символьных строк

Одна из таких мер была предложена Тзе и Янгом в 1981 г. Наиболее простую меру схожести строк можно представить в виде аналитического выражения

$$R = \frac{\alpha}{\max(|a|,|b|) - \alpha}, \tag{3.1}$$

где a, b – строки;

α – число совпадений между строками a и b.

Рассмотренная мера равна бесконечности при полном совпадении строк и равна нулю, когда таких совпадений не существует.

Далее рассмотрим задачу распознавания объектов на основе сопоставления строк.

Пример **–** Считаем изображения трех различных объектов – круга (рисунок 3.1, *а*), квадрата (рисунок 3.1, *б*) и треугольника (рисунок 3.1, *в*).

```
L=imread('im1.bmp');
L=L(:,:,1);
L=double(L)./255;
iptsetpref('ImshowBorder','tight');
figure, imshow(L);
```

а) *б*) *в*)

Рисунок 3.1 – Изображение круга (*а*), квадрата (*б*), треугольника (*в*)

Чтобы продемонстрировать эффективность рассмотренной выше меры схожести для распознавания объектов, сначала границы представленных объектов необходимо приблизить ломаной линией с различными параметрами (различным размером ячейки приближения).

Приближение ломаной линией минимальной длины.

Если речь идет о приближении, то сразу возникает вопрос о точности такого приближения. Наиболее точным приближение будет тогда, когда число отрезков ломаной и число точек границы будут совпадать. Однако на практике суть такого приближения состоит в том, чтобы с помощью наименьшего количества отрезков приблизить форму объекта так, чтобы она была узнаваема. Для этого можно использовать функцию minperpoly [6].

Сформируем результаты приближения контуров объектов функцией minperpoly с размерами ячейки (2, 5 и 11 пикселей).

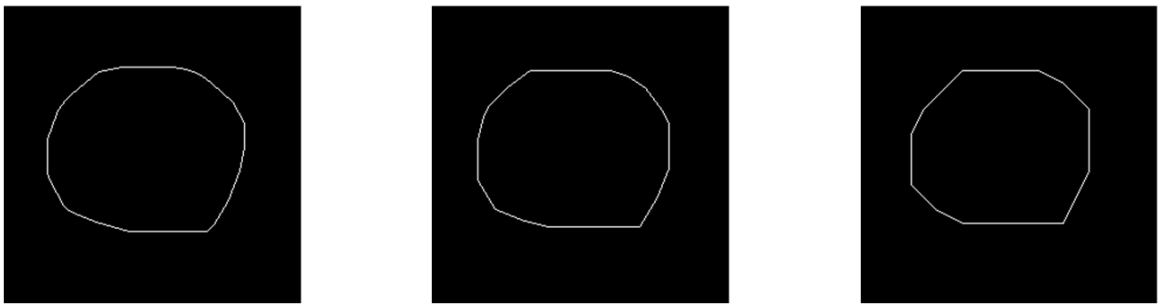
Для первого объекта (круга).

```
cellsize=2;5; 11;
[x,y]=minperpoly(L,cellsize);
L1=zeros(256);
figure, imshow(L1);
line([y;y(1)],[x;x(1)],'color',[1 1 1]);
```

Аналогично приближаем контуры двух других объектов (квадрата и треугольника) с различным размером ячеек.

Таким образом, сформированы три набора изображений различных объектов – круга, квадрата и треугольника. А каждый набор состоит из изображений приближенного контура одного и того же объекта, но с различным размером ячеек (рисунки 3.2–3.4).

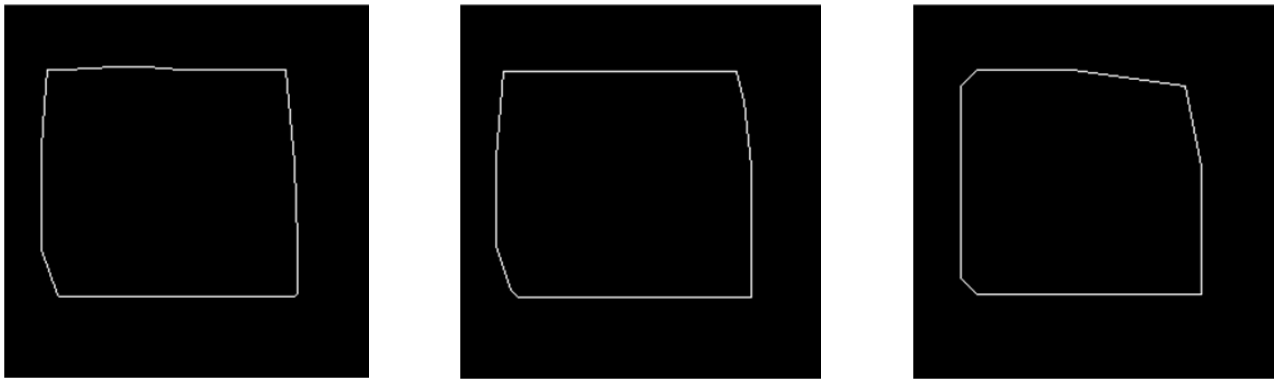
а) *б*) *в*)



а – 2 пикселя; *б* – 5 пикселей; *в* – 11 пикселей

Рисунок 3.2 – Приближение контура первого объекта (круга) с размером ячеек

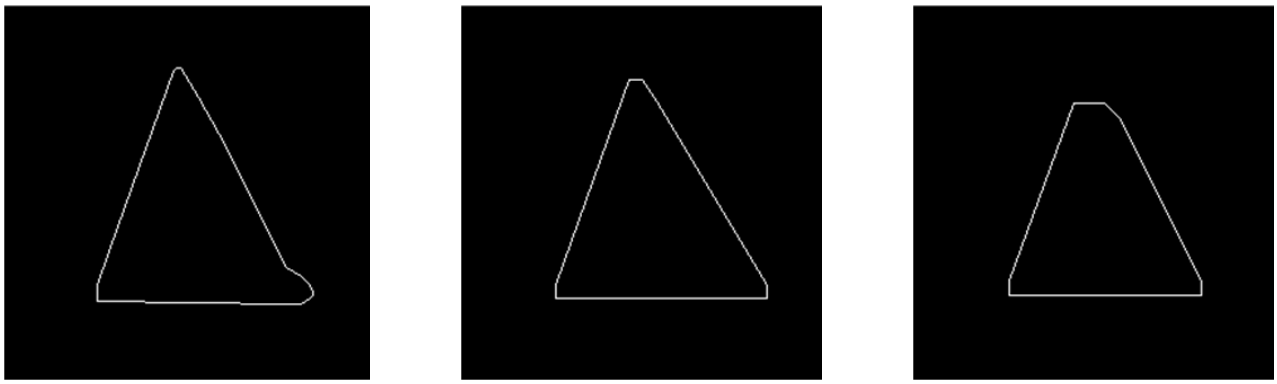
а) *б*) *в*)



а – 2 пикселя; *б* – 5 пикселей; *в* – 11 пикселей

Рисунок 3.3 – Приближение контура второго объекта (квадрата)

а) *б*) *в*)



а – 2 пикселя; *б* – 5 пикселей; *в* – 11 пикселей

Рисунок 3.4 – Приближение контура первого объекта (треугольника) с размером ячеек

На основе этих данных можем вычислить меру схожести строк изображений в пределах одного набора, а также между наборами данных.

Для этого будем использовать выражение (3.1) и реализовано в виде функции strsimilarity [1]:

```
L=imread('image1.bmp');
L=L(:,:,1);
L1=imread('image2.bmp');
L1=L1(:,:,1);
R1=strsimilarity(char(L(64,:)),char(L1(64,:)))
R2=strsimilarity(char(L(128,:)),char(L1(128,:)))
R3=strsimilarity(char(L(192,:)),char(L1(192,:)))
```

Представим полученные данные в виде таблиц 3.1–3.6.

Таблица 3.1 – Вычисление меры схожести строк для изображения круга

Тип изображения	Изображение приближения контура круга с размером ячеек 5 пикселей (рисунок 3.3)	Изображение приближения контура круга с размером ячеек 11 пикселей (рисунок 3.4)	Номер строки
Изображение приближения контура круга с размером ячеек 2 пикселя	127	20,3333	$n = 64$
	63	16,0667	$n = 128$
	2,0843	2,0843	$n = 192$

Таблица 3.2 – Вычисление меры схожести строк для изображения квадрата

Тип изображения	Изображение приближения контура квадрата с размером ячеек 5 пикселей	Изображение приближения контура квадрата с размером ячеек 11 пикселей	Номер строки
Изображение приближения контура квадрата с размером ячеек 2 пикселя	84,3333	18,6923	$n = 64$
	127	24,6000	$n = 128$
	127	35,5714	$n = 192$

Таблица 3.3 – Вычисление меры схожести строк для изображения треугольника

Тип изображения	Изображение приближения контура треугольника с размером ячеек 5 пикселей	Изображение приближения контура треугольника с размером ячеек 11 пикселей	Номер строки
Изображение приближения контура треугольника с размером ячеек 2 пикселя	127	9,6667	$n = 64$
	50,2000	127	$n = 128$
	225	17,2857	$n = 192$

Таблица 3.4 – Вычисление меры схожести строк для изображений из различных наборов, которые были приближены с размером ячейки 2 пикселя

Тип изображения	Изображение приближения контура квадрата с размером ячеек 5 пикселей	Изображение приближения контура треугольника с размером ячеек 11 пикселей	Номер строки
Изображение приближения контура круга с размером ячеек 2 пикселя	2,7647	2,3684	$n = 64$
	13,2222	1,9091	$n = 128$
	2,0118	3,0635	$n = 192$

Таблица 3.5 – Вычисление меры схожести строк для изображений из различных наборов, которые были приближены с размером ячейки 5 пикселей

Тип изображения	Изображение приближения контура квадрата с размером ячеек 5 пикселей	Изображение приближения контура треугольника с размером ячеек 11 пикселей	Номер строки
Изображение приближения контура круга с размером ячеек 5 пикселей	2,8209	2,3684	$n = 64$
	11,8000	2,1605	$n = 128$
	0,5422	0,7655	$n = 192$

Таблица 3.6 – Вычисление меры схожести строк изображений с размером ячейки 11 пикселей

Тип изображения	Изображение приближения контура квадрата с размером ячеек 5 пикселей	Изображение приближения контура треугольника с размером ячеек 11 пикселей	Номер строки
Изображение приближения контура круга с размером ячеек 11 пикселей	2,8209	1,9091	$n = 64$
	22,2727	2,4133	$n = 128$
	0,5901	0,9394	$n = 192$

Проанализируем полученные данные. Из представленных в таблицах данных видно, что в большинстве случаев значение коэффициента схожести строк изображений в пределах одного набора (см. таблицы 3.1–3.3) значительно выше значений коэффициентов схожести строк изображений из различных наборов (см. таблицы 3.4–3.6). Это свойство лежит в основе рассматриваемого структурного распознавания объектов.

3.2 Обнаружение объектов

Существует большое количество задач, одной из составляющих которых является вопрос автоматического распознавания лиц на изображениях. Подходов к решению этого вопроса существует много, однако в рамках данного материала рассмотрим метод автоматического обнаружения лиц на изображениях на основе анализа цвета.

Считаем исходное изображение (рисунок 3.5) в рабочее пространство Mathlab.

Рисунок 3.5 – Исходное изображение

Далее необходимо на изображении лица выбрать пиксели с характерным цветом, определить среднее значение их интенсивности и среднеквадратическое отклонение.

```
[x,y,z]=impixel;
r=median(z(:,1)); std_r=std(z(:,1));
g=median(z(:,2)); std_g=std(z(:,2));
b=median(z(:,3)); std_b=std(z(:,3));
```

На основе знаний о значениях интенсивностей пикселей лица и их возможных вариациях проводится сегментация изображения (рисунок 3.6).

```
%Сегментация на основе анализа цвета.
for i=1:N;
    disp(i);
    for j=1:M;
         if  (abs(L(i,j,1)-r)<std_r)&(abs(L(i,j,2)-g)<std_g)
&(abs(L(i,j,3)-b)«std_b);
         L1(i,j)=1;
         else
             L1(i,j)=0;
         end;
end;
end;
figure, imshow(L1);title('Исходное изображение после сегментации');
```

Рисунок 3.6 – Изображение после сегментации

Целью сегментации было выделение лица на изображении. Однако, вполне естественно, что на изображении присутствовали и другие объекты, значения интенсивностей пикселей которых совпали с интенсивностью пикселей лица. В результате на сегментированном изображении кроме лица выделились и другие объекты. Теперь на сегментированном изображении предстоит найти изображение нужного объекта, т. е. лица. Критерии поиска могут быть разными. Это может быть площадь, форма и др. В данном примере в качестве критерия для поиска лица выберем площадь. Исходя из выбранного ранее способа сегментации, можно предположить, что лицо занимает наибольшую площадь. Поэтому выбор критерия для поиска лица на основе площади позволит удалить другие объекты, которые меньше за площадью.

Удаление объектов, площадь которых меньше некоторой заданной величины (для удаления шума)

```
[mitka num]=bwlabel(L1,8);
feats=imfeature(mitka,'Area',8);
Areas=zeros(num);
for i=1:num;
    Areas(i)=feats(i).Area;
end;
idx=find(Areas>750);
L1=ismember(mitka,idx);
figure,imshow(L1);title('После удаления шума (небольших объектов)');
```

Удалим также и другие объекты на сегментированном изображении лица (рисунок 3.7).

```
L1=1-L1;
[mitka num]=bwlabel(L1,8);
feats=imfeature(mitka,'Area',8);
Areas=zeros(num);
for i=1:num;
    Areas(i)=feats(i).Area;
end;
idx=find(Areas>750);
L1=ismember(mitka,idx);
L1=1-L1;
figure,imshow(L1);
title('После удаления шума (небольших объектов)');
```

а) *б*)

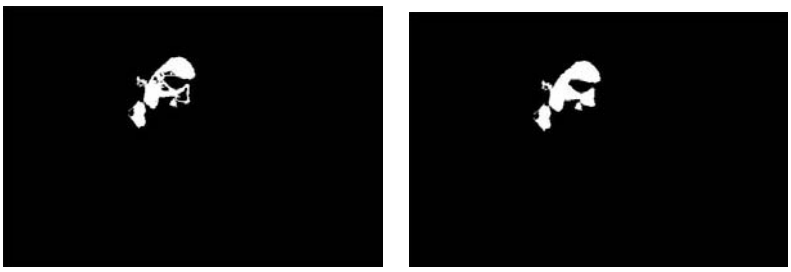


Рисунок 3.7 – Удаление объектов, площадь которых меньше некоторой заданной величины (*а*), на сегментированном изображении лица (*б*)

Здесь следует отметить, что выбранного нами критерия может быть недостаточно для достоверного определения объекта поиска, т. е. изображения лица. Такое может произойти в том случае, если на изображении присутствуют другие объекты с похожим цветом, лицо может быть в тени, быть под наклоном и т. п. Поэтому необходимо использовать также другие критерии для поиска изображения лица. Такие критерии могут базироваться на априорной информации о форме лица.

Таким образом, для повышения достоверности распознавания критерий поиска лица на изображении должен быть комплексным.

Далее проводим выделение выбранного лица, например, прямоугольником (рисунок 3.8).

```
MIN_i=N;MIN_j=M;MAX_i=0;MAX_j=0;
for i=1:N;
    disp(i);
    for j=1:M;
        if L1(i,j)==1;
            if i<MIN_i;
                MIN_i=i;
            end;
            if j<MIN_j;
                MIN_j=j;
            end;
            if i>MAX_i;
                MAX_i=i;
            end;
            if j>MAX_j;
                MAX_j=j;
            end;
        end;
    end;
end;

figure, imshow(L);
line([MIN_j MAX_j],[MIN_i MIN_i]);hold on;
line([MAX_j MAX_j],[MIN_i MAX_i]);
line([MAX_j MIN_j],[MAX_i MAX_i]);
line([MIN_j MIN_j],[MIN_i MAX_i]);hold off;
```

Рисунок 3.8 – Выделение выбранного лица

Программа, реализующая приведенный выше метод:

```
%==========ОБНАРУЖЕНИЕ ЛИЦ НА ОСНОВЕ ЦВЕТА=======

clear;
%Считывание исходного изображения
L=imread('im.jpg');
[N M s]=size(L);
L=double(L)./255;
figure, imshow(L);title('Исходное изображение');

%Выбор пикселей лица с характерным цветом
[x,y,z]=impixel;
r=median(z(:,1)); std_r=std(z(:,1));
g=median(z(:,2)); std_g=std(z(:,2));
b=median(z(:,3)); std_b=std(z(:,3));

%Сегментация на основе анализа цвета
for i=1:N;
 disp(i);
 for j=1:M;
  if  (abs(L(i,j,1)-r)<std_r)&(abs(L(i,j,2)-g)<std_g)&(abs(L(i,j,3)-
b)<std_b);
    L1(i,j)=1;
   else
     L1(i,j)=0;
   end;
end;
end;
figure, imshow(L1);title('Исходное изображение после сегментации');

%Удаление объектов, площадь которых меньше некоторой
 заданной величины (для удаления шума)
[mitka num]=bwlabel(L1,8);
feats=imfeature(mitka,'Area',8);
Areas=zeros(num);
for i=1:num;
    Areas(i)=feats(i).Area;
end;
idx=find(Areas>750);
L1=ismember(mitka,idx);
figure,imshow(L1);title('После удаления шума (небольших объектов)');
L1=1-L1;
 [mitka num]=bwlabel(L1,8);
feats=imfeature(mitka,'Area',8);
Areas=zeros(num);
for i=1:num;
    Areas(i)=feats(i).Area;
end;
idx=find(Areas>750);
L1=ismember(mitka,idx);
L1=1-L1;
figure,imshow(L1);title('После удаления шума (небольших объектов)');

%STATS = regionprops(L1,'Eccentricity');
MIN_i=N;MIN_j=M;MAX_i=0;MAX_j=0;
for i=1:N;
    disp(i);
```

```
    for j=1:M;
        if L1(i,j)==1;
            if i<MIN_i;
                MIN_i=i;
            end;
            if j<MIN_j;
                MIN_j=j;
            end;
            if i>MAX_i;
                MAX_i=i;
            end;
            if j>MAX_j;
                MAX_j=j;
            end;
        end;
    end;
end;

figure, imshow(L);
line([MIN_j MAX_j],[MIN_i MIN_i]);hold on;
line([MAX_j MAX_j],[MIN_i MAX_i]);
line([MAX_j MIN_j],[MAX_i MAX_i]);
line([MIN_j MIN_j],[MIN_i MAX_i]);hold off;
```

Практическое задание

1 Разработать программу для определения фигур на основе значений коэффициентов схожести строк изображений из различных наборов.

2 Разработать программу для распознавания лица на произвольной фотографии.

4 Трекинг и движение

Цель работы: освоить распознавание и отслеживание движения, оценку перемещения объекта в видеоряде.

Обнаружение перемещения объектов и основанного на движении отслеживания является важными компонентами многих приложений компьютерного зрения, включая распознавание активности, контроль трафика и автомобильную безопасность. Проблема основанного на движении объектного отслеживания может быть разделена на две части:

1) обнаружение движущихся объектов в каждой системе координат;

2) соединение обнаружений, соответствующих тому же объекту, в зависимости от времени.

Для обнаружения движущихся объектов используется алгоритм вычитания фона, основанный на смешанных гауссовских моделях. Морфологические операции применяются к полученной маске переднего плана для устранения шума. Анализ больших двоичных объектов обнаруживает группы связанных пикселей, которые, вероятно, соответствуют движущимся объектам.

Ассоциация обнаружения к тому же объекту базируется только на

движении. Движение каждого трека оценивается фильтром Калмана. Фильтр используется для предсказания местоположения трека в каждой системе координат и определения вероятности каждого обнаружения, присваиваемого каждой дорожке.

Обслуживание дорожки становится важным аспектом этого примера. В любой данной системе координат некоторые обнаружения могут быть присвоены трекам, в то время как другие обнаружения и треки могут остаться неприсвоенными. Присвоенные треки обновляются с помощью соответствующих обнаружений. Неприсвоенные треки отмечены как невидимые. Неприсвоенное обнаружение начинает новый трек (рисунок 4.1).

Каждый трек проводит подсчет количества последовательных систем координат, где это осталось неприсвоенным. Если количество превышает заданный порог, значит объект покинул поле представления, и трек удаляется.

```
% Create System objects used for reading video, detecting moving
objects,
% and displaying the results.
obj = setupSystemObjects();
tracks = initializeTracks(); % Create an empty array of tracks.
nextId = 1; % ID of the next track
% Detect moving objects, and track them across video frames.
while ~isDone(obj.reader)
    frame = readFrame();
    [centroids, bboxes, mask] = detectObjects(frame);
    predictNewLocationsOfTracks();
    [assignments, unassignedTracks, unassignedDetections] = ...
        detectionToTrackAssignment();
    updateAssignedTracks();
    updateUnassignedTracks();
    deleteLostTracks();
    createNewTracks();
    displayTrackingResults();
end
```

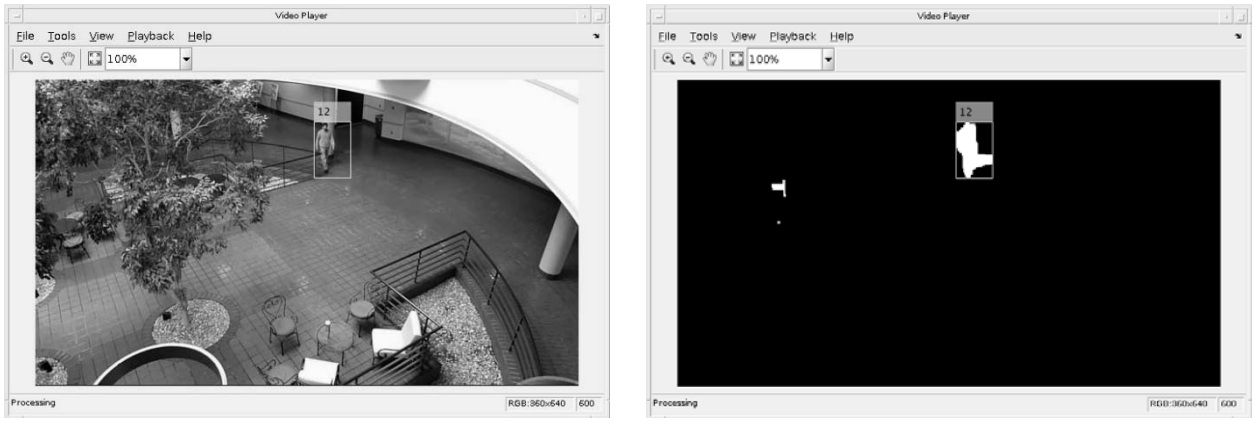


Рисунок 4.1 – Изображение исходного видеокадра и детектированного объекта

Создание системных объектов.

Создайте системные объекты (SystemObjects), используемые в чтении видеокадров, обнаружении основных объектов и отображении результатов.

```
function obj = setupSystemObjects()
    % Initialize Video I/O
        % Create objects for reading a video from a file, drawing the
tracked
        % objects in each frame, and playing the video.
        % Create a video file reader.
        obj.reader = vision.VideoFileReader('atrium.mp4');
        % Create two video players, one to display the video,
        % and one to display the foreground mask.
        obj.maskPlayer = vision.VideoPlayer('Position', [740, 400, 700,
400]);
        obj.videoPlayer = vision.VideoPlayer('Position', [20, 400, 700,
400]);
        % Create System objects for foreground detection and blob
analysis
        % The foreground detector is used to segment moving objects from
        % the background. It outputs a binary mask, where the pixel
value
        % of 1 corresponds to the foreground and the value of 0
corresponds
        % to the background.
          obj.detector = vision.ForegroundDetector('NumGaussians', 3,
...
       'NumTrainingFrames', 40, 'MinimumBackgroundRatio', 0.7);
        % Connected groups of foreground pixels are likely to correspond
to moving
      % objects.  The blob analysis System object is used to find such
groups
      % (called 'blobs' or 'connected components'), and compute their
      % characteristics, such as area, centroid, and the bounding box.
        obj.blobAnalyser = vision.BlobAnalysis('BoundingBoxOutputPort',
true,
            'AreaOutputPort', true, 'CentroidOutputPort', true, ...
            'MinimumBlobArea', 400);
    end
```

Инициация дорожки.

InitializeTracks функция создает массив дорожек, где каждая дорожка является структурой, представляющей движущийся объект в видео. Цель структуры состоит в том, чтобы обеспечить состояние отслеживаемого объекта. Состояние состоит из информации, используемой в обнаружении, чтобы отследить присвоение, завершение дорожки и отображение.

Структура содержит следующие поля:

1) id: целочисленный ID дорожки;

2) bbox: текущая ограничительная рамка объекта, используемая при отображении;

3) kalmanFilter: объект фильтр Калмана используется в основном для распознавания движения при отслеживании;

4) age: количество систем координат;

5) totalVisibleCount : общее количество систем координат, в которых дорожка была обнаружена (видимая);

6) consecutiveInvisibleCount : количество последовательных систем координат, для которых дорожка не была обнаружена (невидимая).

Зашумленные изображения имеют тенденцию приводить к коротким трекам. Поэтому пример отображает объект только после того, как он был прослежен для некоторого количества систем координат. Это происходит когда totalVisibleCount превышает заданный порог.

Когда никакие обнаружения не сопоставлены с дорожкой для нескольких последовательных систем координат, пример принимает, что объект покинул поле представления и удаляет дорожку. Это происходит когда consecutiveInvisibleCount превышает заданный порог. Дорожка может также быть удалена как шум, если она была отслежена в течение короткого времени, и отмечена как невидимая для большинства систем координат.

```
    function tracks = initializeTracks()
        % create an empty array of tracks
        tracks = struct(...
            'id', {}, ...
            'bbox', {}, ...
            'kalmanFilter', {}, ...
            'age', {}, ...
            'totalVisibleCount', {}, ...
            'consecutiveInvisibleCount', {});
    end
```

Считывание видеокадра.

Считайте следующий видеокадр из видеофайла.

```
    function frame = readFrame()
        frame = obj.reader.step();
    end
```

Обнаружение объектов.

detectObjects функция возвращает центроиды и ограничительные рамки обнаруженных объектов. Это также возвращает бинарную маску, которая имеет тот же размер как входной кадр. Пиксели со значением 1 соответствуют переднему плану, и пиксели со значением 0 соответствуют фону.

Функция выполняет сегментацию движения с помощью приоритетного детектора. Это затем выполняет морфологические операции на получившейся бинарной маске, чтобы удалить шумные пиксели и заполнить отверстия в остающихся блобах.

```
    function [centroids, bboxes, mask] = detectObjects(frame)
        % Detect foreground.
        mask = obj.detector.step(frame);
        % Apply morphological operations to remove noise and fill in
holes.
        mask = imopen(mask, strel('rectangle', [3,3]));
        mask = imclose(mask, strel('rectangle', [15, 15]));
        mask = imfill(mask, 'holes');
        % Perform blob analysis to find connected components.
        [~, centroids, bboxes] = obj.blobAnalyser.step(mask);
    end
```

Предсказание новых положений существующих дорожек.

Используйте фильтр Калмана, чтобы предсказать центроид каждой дорожки в текущей системе координат и обновить ее ограничительную рамку соответственно.

```
function predictNewLocationsOfTracks()
    for i = 1:length(tracks)
        bbox = tracks(i).bbox;
        % Predict the current location of the track.
        predictedCentroid = predict(tracks(i).kalmanFilter);
        % Shift the bounding box so that its center is at
        % the predicted location.
        predictedCentroid = int32(predictedCentroid) - bbox(3:4) / 2;
        tracks(i).bbox = [predictedCentroid, bbox(3:4)];
    end
end
```

Присвоение обнаружения дорожкам.

Назначение обнаружений объектов в текущем кадре существующим трекам осуществляется путем минимизации затрат. Стоимость определяется как отрицательная логарифмическая вероятность обнаружения, соответствующего дорожке.

Алгоритм включает два шага.

1 Вычислите стоимость присвоения каждого обнаружения к каждой дорожке с помощью distance метод vision.KalmanFilter система object™. Стоимость учитывает евклидово расстояние между предсказанным центроидом дорожки и центроидом обнаружения. Это также включает уверенность прогноза, который обеспечен фильтром Калмана. Результаты хранятся в матрице M × N, где M – количество дорожек; N – количество обнаружений.

2 Решите задачу присвоения, представленную матрицей стоимости использование assignDetectionsToTracks функция. Функция берет матрицу стоимости и стоимость неприсвоения любых обнаружений к дорожке.

Значение для стоимости неприсвоения обнаружения к дорожке зависит от области значений, возвращенных distance методом vision.KalmanFilter. Это значение, которое должно быть настроено экспериментально. Установка его на низком уровне не увеличивает вероятность создания нового трека и может привести к фрагментации дорожки. Установка его слишком высоко может привести к одноколейному пути, соответствующему серии отдельных движущихся объектов.

assignDetectionsToTracks использует версию венгерского алгоритма Манкреса, который для определения присвоения минимизирует общую стоимость. Это возвращает M × 2 матрицы, содержащие соответствующие индексы присвоенных дорожек и их обнаружений в двух столбцах. Это также возвращает индексы дорожек и обнаружений, которые остались неприсвоенными.

```
function [assignments, unassignedTracks, unassignedDetections] = ...
        detectionToTrackAssignment()
```

```
    nTracks = length(tracks);
    nDetections = size(centroids, 1);
    % Compute the cost of assigning each detection to each track.
    cost = zeros(nTracks, nDetections);
    for i = 1:nTracks
        cost(i, :) = distance(tracks(i).kalmanFilter, centroids);
    end
    % Solve the assignment problem.
    costOfNonAssignment = 20;
    [assignments, unassignedTracks, unassignedDetections] = ...
        assignDetectionsToTracks(cost, costOfNonAssignment);
end
```

Обновление присвоенных дорожек.

updateAssignedTracks обновляет каждую присвоенную дорожку с соответствующим обнаружением. Это вызывает коррекцию метода vision.KalmanFilter откорректировать оценку местоположения, сохраняет новую ограничительную рамку и увеличивает возраст дорожки и общего видимого количества на 1. Наконец, функция устанавливает невидимое количество на 0.

```
function updateAssignedTracks()
    numAssignedTracks = size(assignments, 1);
    for i = 1:numAssignedTracks
        trackIdx = assignments(i, 1);
        detectionIdx = assignments(i, 2);
        centroid = centroids(detectionIdx, :);
        bbox = bboxes(detectionIdx, :);
        % Correct the estimate of the object's location
        % using the new detection.
        correct(tracks(trackIdx).kalmanFilter, centroid);
        % Replace predicted bounding box with detected
        % bounding box.
        tracks(trackIdx).bbox = bbox;
        % Update track's age.
        tracks(trackIdx).age = tracks(trackIdx).age + 1;
        % Update visibility.
        tracks(trackIdx).totalVisibleCount = ...
            tracks(trackIdx).totalVisibleCount + 1;
        tracks(trackIdx).consecutiveInvisibleCount = 0;
    end
end
```

Обновление неприсвоенных дорожек.

Отметьте каждую неприсвоенную дорожку как невидимая, и увеличьте ее возраст на 1.

```
function updateUnassignedTracks()
    for i = 1:length(unassignedTracks)
        ind = unassignedTracks(i);
        tracks(ind).age = tracks(ind).age + 1;
        tracks(ind).consecutiveInvisibleCount = ...
            tracks(ind).consecutiveInvisibleCount + 1;
    end
end
```

Удаление потерянных следов.

deleteLostTracks удаляет дорожки, которые были невидимы для последовательных систем координат, а также недавно созданные дорожки, которые были невидимы для других систем координат.

```
    function deleteLostTracks()
        if isempty(tracks)
            return;
        end
        invisibleForTooLong = 20;
        ageThreshold = 8;
        % Compute the fraction of the track's age for which it was
visible.
        ages = [tracks(:).age];
        totalVisibleCounts = [tracks(:).totalVisibleCount];
        visibility = totalVisibleCounts ./ ages;
        % Find the indices of 'lost' tracks.
        lostInds = (ages < ageThreshold & visibility < 0.6) | ...
            [tracks(:).consecutiveInvisibleCount] >=
invisibleForTooLong;
        % Delete lost tracks.
        tracks = tracks(~lostInds);
    end
```

Создание новых треков.

Создайте новые треки из неприсвоенных обнаружений. Примите, что любое неприсвоенное обнаружение является запуском нового трека. На практике можно использовать другие сигналы, чтобы устранить шумные обнаружения, такие как размер, местоположение или внешний вид.

```
    function createNewTracks()
        centroids = centroids(unassignedDetections, :);
        bboxes = bboxes(unassignedDetections, :);
        for i = 1:size(centroids, 1)
            centroid = centroids(i,:);
            bbox = bboxes(i, :);
            % Create a Kalman filter object.
            kalmanFilter = configureKalmanFilter('ConstantVelocity', ...
                centroid, [200, 50], [100, 25], 100);
            % Create a new track.
            newTrack = struct(...
                'id', nextId, ...
                'bbox', bbox, ...
                'kalmanFilter', kalmanFilter, ...
                'age', 1, ...
                'totalVisibleCount', 1, ...
                'consecutiveInvisibleCount', 0);
            % Add it to the array of tracks.
            tracks(end + 1) = newTrack;
            % Increment the next id.
            nextId = nextId + 1;
        end
    end
```

Отображение результатов отслеживания.

displayTrackingResults чертит ограничительную рамку и метку ID для каждой дорожки на видеокадре и приоритетной маске, а затем отображает систему координат и их маску в соответствующих видеоплеерах.

```
function displayTrackingResults()
    % Convert the frame and the mask to uint8 RGB.
    frame = im2uint8(frame);
    mask = uint8(repmat(mask, [1, 1, 3])) .* 255;
    minVisibleCount = 8;
    if ~isempty(tracks)
        % Noisy detections tend to result in short-lived tracks.
        % Only display tracks that have been visible for more than
        % a minimum number of frames.
        reliableTrackInds = ...
            [tracks(:).totalVisibleCount] > minVisibleCount;
        reliableTracks = tracks(reliableTrackInds);
        % Display the objects. If an object has not been detected
        % in this frame, display its predicted bounding box.
        if ~isempty(reliableTracks)
            % Get bounding boxes.
            bboxes = cat(1, reliableTracks.bbox);
            % Get ids.
            ids = int32([reliableTracks(:).id]);
            % Create labels for objects indicating the ones for
            % which we display the predicted rather than the actual
            % location.
            labels = cellstr(int2str(ids'));
            predictedTrackInds = ...
                [reliableTracks(:).consecutiveInvisibleCount] > 0;
            isPredicted = cell(size(labels));
            isPredicted(predictedTrackInds) = {' predicted'};
            labels = strcat(labels, isPredicted);
            % Draw the objects on the frame.
            frame = insertObjectAnnotation(frame, 'rectangle', ...
                bboxes, labels);
            % Draw the objects on the mask.
            mask = insertObjectAnnotation(mask, 'rectangle', ...
                bboxes, labels);
        end
    end
    % Display the maskand the frame.
    obj.maskPlayer.step(mask);
    obj.videoPlayer.step(frame);
end
```

Отслеживание в этом примере только было основано на движении, учитывая, что все объекты перемещаются в прямую линию с постоянной скоростью. Когда движение объекта значительно отклоняется от этой модели, пример может произвести ошибки отслеживания. Заметьте ошибку в отслеживании человека, когда он будет закрыт деревом.

Вероятность отслеживания ошибок может уменьшаться при помощи комплексной модели движения, такой как постоянное ускорение, или при помощи нескольких фильтров Калмана для каждого объекта. Кроме того, можно

включить другие сигналы для соединения обнаружений в зависимости от времени, таких как размер, форма и цвет.

Практическое задание

Используя различные видео, обнаружить и отследить перемещающийся объект, изменяя параметры обнаружения, путем присвоения и удаления шагов.

5 Поиск и локализация классов объектов

Цель работы: освоить поиск и локализация классов объектов для распознавания объекта в видеопотоке.

Computer Vision Toolbox™ поддерживает несколько подходов для классификации изображений, обнаружения объектов и распознавания, включая:

- глубокое обучение и Сверточные нейронные сети (CNNs);
- набор признаков;
- сравнение с шаблонами;
- Blob-анализ;
- алгоритм Виолы-Джонса;
- интерактивные приложения для разметки достоверных данных.

CNN является популярной архитектурой глубокого обучения, которая автоматически изучает полезные функции непосредственно от данных изображения. Набор признаков кодирует функции изображений в компактное представление, подходящее для классификации изображений и извлечения изображений. Сравнение с шаблонами использует маленькое изображение или шаблон, чтобы найти соответствие с областями в увеличенном изображении. Blob-анализ использует сегментацию и свойства блоба идентифицировать предметы распознавания. Алгоритм Виолы-Джонса использует подобные Хаару функции и каскад классификаторов, чтобы идентифицировать объекты, включая поверхности, носы и глаза. Можно обучить этот классификатор распознавать другие объекты.

Интерактивное изображение и маркировка видео для обнаружения объектов, семантической сегментации и классификации изображений.

Используйте Image Labeler и приложение Video Labeler, чтобы интерактивно помечать достоверные данные в наборе изображений, видео или последовательности изображений. Можно пометить прямоугольные видимые области (ROIs) для обнаружения объектов, пиксели для семантической сегментации и сцены для классификации изображений. Приложение также включает алгоритмы компьютерного зрения, чтобы автоматизировать маркировку достоверных данных для использования алгоритмов с обнаружением и отслеживанием. Это также обеспечивает API и рабочий процесс, который позволяет вам импортировать свои собственные алгоритмы, чтобы автоматизировать маркировку достоверных данных.

Для создания меток в изображении применяют (Image Labeler). Для создания меток в видеопотоке необходимо применить Video Labeler [7, 8].

Маркировка пикселей для семантической сегментации.

Image Labeler, Video Labeler и Ground Truth Labeler (требует Automated Driving Toolbox™) приложения позволяют вам присвоить пиксельные метки вручную. Каждый пиксель может иметь одну метку. Метки используются, чтобы создать достоверные данные для учебных алгоритмов семантической сегментации.

Запуск пиксельной маркировки.

Сначала необходимо загрузить изображение или видео в приложение для маркировки и определения пиксельных меток прямоугольных областей (ROI) (рисунок 5.1).

а) *б*)

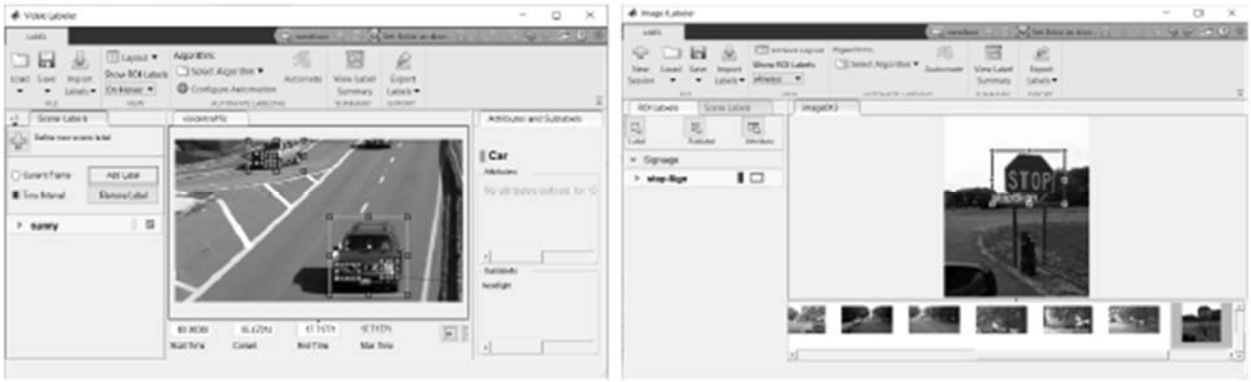


Рисунок 5.1 – Создание меток в изображении (*а*), в видеопотоке (б)

Для этого выберите пиксельное определение метки из панели **ROI Label Definition**. Во вкладке **Label Pixels** находятся инструменты для разметки вручную с помощью многоугольников, кистей или заливки. Инструменты маркировки можно использовать в любом порядке. Эта вкладка также имеет средства управления, чтобы настроить отображение изображения путем изменения масштаба и панорамирования и прозрачности меток.

Этот пример использует две стратегии метки пикселей.

1 Сначала используйте полуавтоматические инструменты, такие как **Flood Fill** и **Smart Polygon**. Затем совершенствуйте инструменты использования меток, которые предлагают прямое управление, такое как **Polygon**, **Assisted Freehand** и **Brush**.

2 Сначала пометьте удаленные объекты грубой оценкой границ объекта. Затем пометьте более близкие объекты более точными границами объекта.

Маркировка пикселей с использованием инструмента Заливка.

Инструмент Flood Fill помечает группу связанных пикселей, имеющих одинаковый цвет. На этом изображении небо является хорошим кандидатом для заливки, потому что граница яркого неба четко видна на фоне темной растительности и эстакады. При этом заливка не может изолировать растительность, потому что цвет растительности слишком похож на цвет барьеров, дороги и транспортных средств.

Для этого с помощью **Flood Fill**.

1 Выберите инструмент и метку (рисунок 5.2, *а*). Указатель превращается в банку с краской .

2 Кликните по стартовому пикселю в изображении (рисунок 5.2, *б*).

а) *б*)

Рисунок 5.2 – Создание меток с помощью инструмента Заливка

Маркировка Pixels Using Smart Polygon Tool.

Инструмент **Smart Polygon** оценивает форму предмета распознавания в многоугольнике, который вы чертите. Инструмент полезен, когда форма объекта не является простым многоугольником. Этот пример использует **Smart Polygon**, чтобы пометить растительность, которая имеет сложный контур с небом.

Пометить пиксели с помощью **Smart Polygon** можно следующим образом**.**

1 Выберите инструмент и метку. Указатель превращается в перекрестие ┼.

2 Щелкните, чтобы добавить вершины многоугольника. Полностью окружите предмет распознавания с некоторым пробелом между объектом и многоугольником.

3 Закройте многоугольник путем нажатия на первую вершину после размещения других вершин. В качестве альтернативы можно дважды кликнуть, чтобы добавить последнюю вершину и закрыть многоугольник за один шаг. После того, как вы закроете многоугольник, инструмент начертит начальную метку.

4 Отрегулируйте форму и положение многоугольника. Когда объект интереса простирается до края изображения, перетащите вершины к краю изображения, чтобы убедиться, что интеллектуальный полигон полностью окружает объект. Например, в этом примере показаны две крайние левые вершины, расположенные на левом краю изображения.

Рисунок 5.3 – Создание меток с помощью инструмента Smart Polygon

5 Используйте инструменты **Smart Polygon Editor** для изменения метки. При этом:

– выберите **Mark Foreground**, чтобы отметить области в области, которую вы хотите пометить. Приоритетные метки появляются в зеленом;

– выберите **Mark Background**, чтобы отметить области в области, которую вы не хотите помечать. Фоновые метки появляются в красном;

– выберите **Erase Marks**, чтобы удалить передний план или фоновые метки, которые больше не необходимы.

Рисунок 5.4 – Редактирование меток с помощью инструмента Smart Polygon Editor

6 Чтобы завершить метку, нажмите **Enter** или выберите новый **ROI Label Definition**. Вы больше не можете редактировать вершины многоугольника или отмечать передний план и фоновые области памяти.

Маркировка пикселей с помощью инструмента Polygon.

Инструмент **Polygon** помечает все пиксели в многоугольнике, который вы чертите. Средства управления для определения и корректировки вершин многоугольника похожи на средства управления инструмента Smart Polygon (рисунок 5.5).

Добавьте дополнительные многоугольники по структурам, таким как барьеры и дорога (рисунок 5.6). Много пикселей транспортного средства неправильно помечены. Следующий шаг показывает, как заменить ошибочные метки на правильную метку.

Рисунок 5.5 – Маркировка пикселей с помощью инструмента Polygon

Рисунок 5.6 – Дополнительные метки по структурам барьеры и дорога

Маркировка пикселей с помощью вспомогательного инструмента Freehand.

Инструмент **Assisted Freehand** позволяет вам чертить ROI, который автоматически следует за ребром предмета в основном изображении (рисунок 5.7). Можно также настроить размер и положение ROI при помощи мыши.

Рисунок 5.7 – Маркировка пикселей с помощью вспомогательного инструмента Freehand

Замена пиксельных меток.

Каждый пиксель может иметь только одну метку на один пиксель. Когда вы применяете метку к пикселю, новая метка заменяет предыдущую метку.

Этот пример использует инструмент **Smart Polygon**, чтобы пометить

пиксели, принадлежащие грузовику. Приоритетные метки присваивают метку *транспортного средства* подобластям. Фоновые метки возвращаются подобласти к своей предшествующей метке. Например, в первой паре изображений, фоновые метки возвращаются подобласти к меткам *растительности* и *небу*. Точно также во второй паре изображений, фоновые метки возвращаются подобласти к *дорожной* метке (рисунок 5.8).

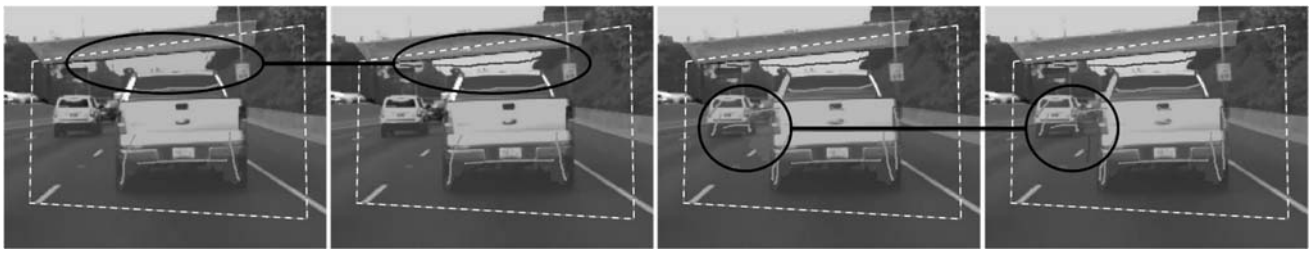


Рисунок 5.8 – Замена пиксельных меток

Граница грузовика является зубчатой, потому что **Smart Polygon** помечает целые подобласти, не отдельные пиксели. Следующий шаг показывает, как совершенствовать метки вдоль границы грузовика.

Совершенствование меток, используя кисть.

Инструмент **Brush** помечает пиксели, когда вы дистиллируете изображение с мышью. Этот пример использует **Brush**, чтобы удалить дефектные пиксели из дороги и сделать ребра грузовика более сглаженными (рисунок 5.9).

Рисунок 5.9 – Пометка при помощи инструмента Brush

Пометить пиксели с помощью **Brush**.

1 Выберите инструмент и метку. Изменения указателя в инструменте перо и квадрат указывают на размер кисти.

2 Настройте размер кисти при помощи ползунка **Brush Size**.

3 Перетащите мышь, чтобы пометить пиксели.

Инструмент **Erase** удаляет пиксельные метки, когда вы очищаете изображение мышью.

Визуализация пиксельных меток.

Можно изменить представление изображения, чтобы упростить пиксельную маркировку. **Zoom In**, **Zoom Out** и опции **Pan** позволяют вам масшта-

бировать и панорамировать изображение с мышью. Чтобы возобновить пиксельную маркировку, кликните по значку **Label**.

Ползунок **Label Opacity** настраивает непрозрачность всех пиксельных меток.

Уменьшите непрозрачность, чтобы видеть изображение более ясно. Например, уменьшите непрозрачность, чтобы облегчить нахождение границ между нижней частью автомобиля и дорогой.

Увеличьте непрозрачность, чтобы видеть сегментацию более ясно. Например, увеличьте непрозрачность, чтобы видеть, что ребро вдоль переднего бампера автомобиля должно сглаживаться. Кроме того, заметьте, что барьер и некоторые удаленные транспортные средства не пометили пиксели (рисунок 5.10).

Рисунок 5.10 – Пометка при помощи инструмента Brush

Финальное изображение выглядит следующим образом (рисунок 5.11).

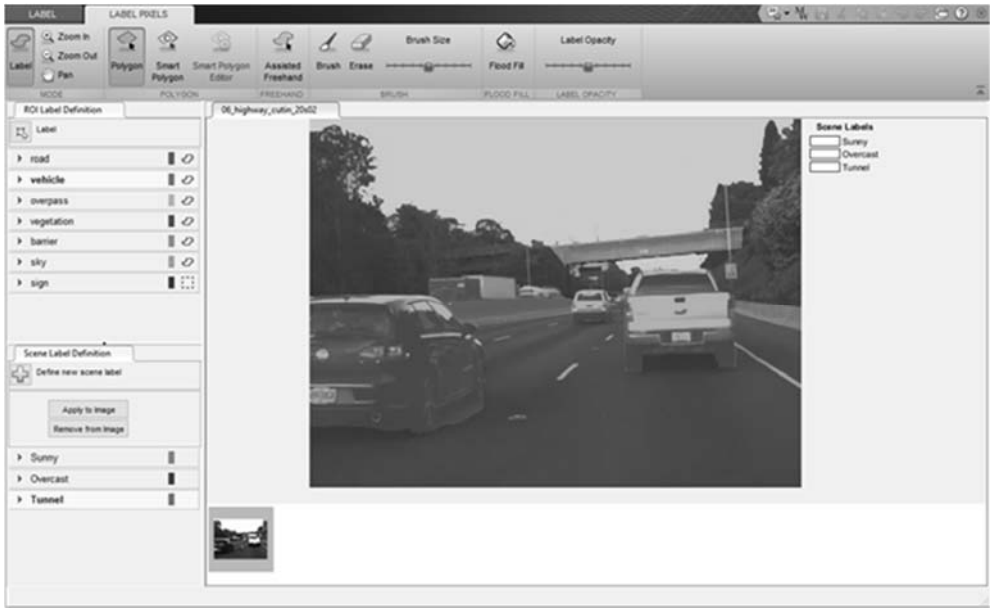


Рисунок 5.11 – Финальное изображение с размеченными областями

Инструмент **Smart Polygon** идентифицирует предмет распознавания при помощи региональной, основанной на графике, сегментации (GrabCut). Инструмент **Smart Polygon** делит изображение на подобласти и обрабатывает все подобласти, которые являются полностью или частично вне многоугольника как принадлежащий фону. Поэтому, чтобы получить оптимальную сегментацию, убедитесь, что объект, который будет помечен, полностью содержится в многоугольнике, окруженном несколькими фоновыми пикселями.

Все пиксели в подобласти имеют ту же метку. Если отметить пиксели вне многоугольника, то это не оказывает влияния на метку.

Каждый пиксель может иметь одну метку. Когда вы применяете новую метку к пикселю, то она заменяет предыдущую.

Пиксельная маркировка отключена, когда вы панорамируете и масштабируете изображение. Необходимо нажать кнопку **Label**, чтобы возобновить пиксельную маркировку.

Чтобы гарантировать, что все пиксели в изображении помечены, начните с маркировки целого изображения одной меткой. Выберите метку, которая представляет преобладающую прямоугольную видимую зону (ROI) в изображении, такую как *небо*, *дорога* или *фон*. Затем используйте инструменты маркировки, чтобы повторно пометить объекты их правильной меткой.

Чтобы заполнить все или все остающиеся пиксели, выберите метку ROI из своего списка и нажмите **Shift + Click** [8].

Практическое задание

Составить алгоритм и программу для распознавания объекта в видеопотоке.

6 Машинное обучение и классификация изображений

Цель работы: изучить подходы машинного обучения и классификации изображений.

Глубокое обучение (*deep learning*) является ответвлением машинного обучения, которое обучает компьютеры делать то, что свойственно человеческому мозгу – обучение на собственном опыте. Алгоритмы машинного обучения используют вычислительные методы, чтобы «узнать» об информации непосредственно из данных, не полагаясь на определенное уравнение как на модель. Глубокое обучение подходит для распознавания изображений, которое важно для решения таких проблем, как распознавание лиц, обнаружение движения и беспилотных технологий: автономное управление, определение маршрута, обнаружение пешеходов и автономная парковка.

Предварительно обученные сети классификации изображений были обучены на более чем миллионе изображений и могут классифицировать изображения в различных категориях объектов, таких как клавиатура, кофейная кружка, карандаш и многие животные. Сети изучили множество функций представления для широкого спектра изображений. Сеть берет изображение в

качестве входа, и затем выводит метку для объекта в изображении вместе с вероятностями определения для каждой из категорий объектов.

Изучение передачи опыта обычно используется в применении глубокого обучения. Можно взять предварительно обученную сеть и использовать ее в качестве отправной точки, чтобы изучить новую задачу. Подстройка сети с передачей опыта, обучаемой, обычно намного быстрее и легче, чем обучение сети с нуля со случайным образом инициализированными критериями. Можно быстро передать изученные функции новой задаче с помощью меньшего числа обучающих изображений [9–11].

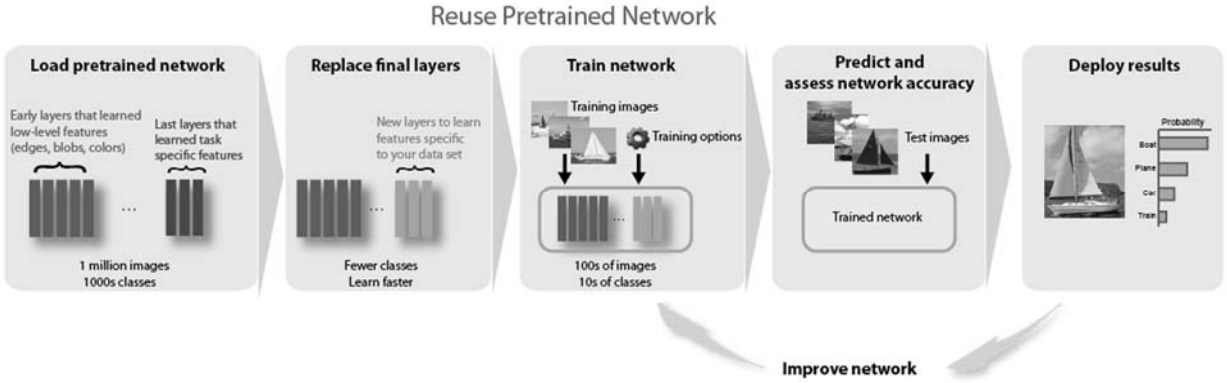


Рисунок 6.1 – Процесс обучения новой нейронной сети

Загрузка данных.

Разархивируйте и загрузите новые изображения как базу данных изображений, которая содержит только 75 изображений. Разделите данные на наборы данных для обучения и валидации. Используйте 70 % изображений для обучения и 30 % для валидации [12].

```
unzip('MerchData.zip');
imds = imageDatastore('MerchData', ...
    'IncludeSubfolders',true, ...
    'LabelSource','foldernames');
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7);
```

Загрузка предварительно обученной сети.

Загрузите предварительно обученную сеть GoogLeNet. Если Модель Deep Learning Toolbox™ *для* пакета *Сетевой поддержки GoogLeNet* не установлена, то программное обеспечение обеспечивает ссылку на загрузку.

Чтобы попробовать различную предварительно обученную сеть, откройте этот пример в Matlab® и выберите различную сеть. Например, можно попробовать squeezenet, сеть, которая еще быстрее, чем googlenet [13].

```
net = googlenet;
```

Используйте analyzeNetwork, чтобы отобразить интерактивную визуализацию сетевой архитектуры и подробной информации о сетевых слоях analyzeNetwork(net) (рисунок 6.2).

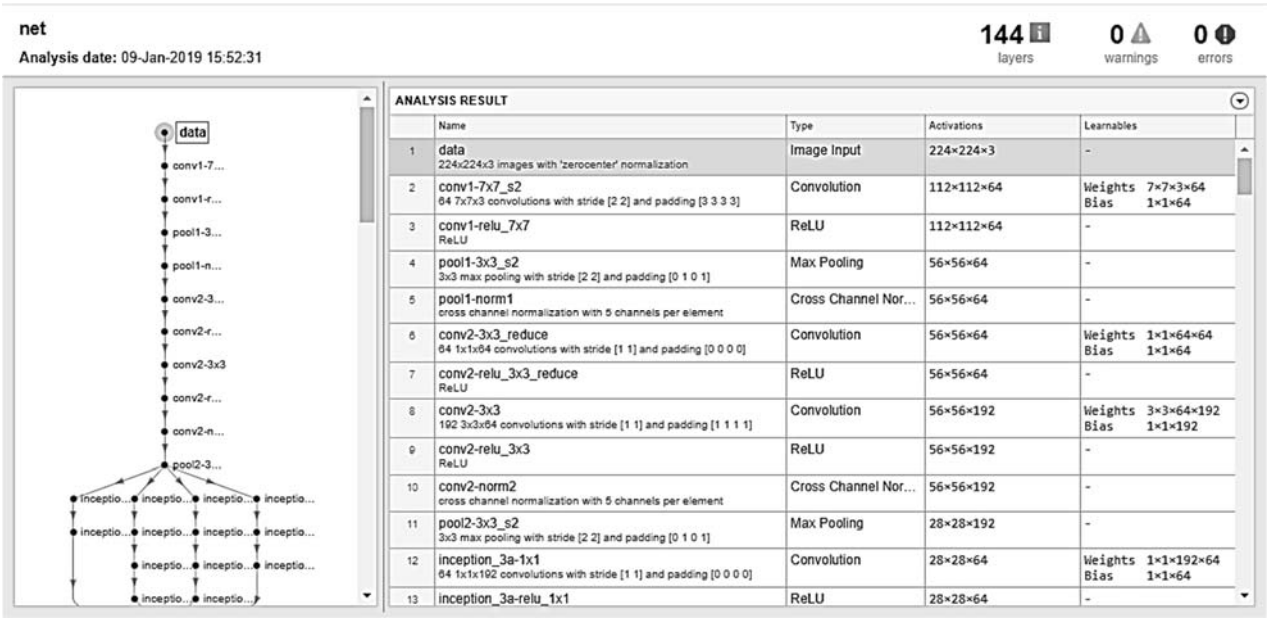


Рисунок 6.2 – Структура нейронной сети

Первый элемент свойства Layers сети является входным слоем изображений. Этот слой требует входных изображений размером $224 \times 224 \times 3$, где 3 – количество цветовых каналов.

```
net.Layers(1)
ans =
  ImageInputLayer with properties:

                  Name: 'data'
             InputSize: [224 224 3]

   Hyperparameters
    DataAugmentation: 'none'
       Normalization: 'zerocenter'
        AverageImage: [224x224x3 single]

inputSize = net.Layers(1).InputSize;
```

Замена последних слоёв.

Два слоя, loss3-classifier и output в GoogLeNet, содержат информацию о том, как сочетать функции, которые сеть извлекает в вероятности класса, значение потерь и предсказанные метки. Чтобы переобучить предварительно обученную сеть и классифицировать новые изображения, замените эти два слоя на новые слои, адаптированные к новому набору данных.

Извлеките график слоя от обучившего сеть. Если сеть является объектом SeriesNetwork, таким как AlexNet, VGG-16 или VGG-19, то преобразовывают список слоев в net.Layers к графику слоя.

```
if isa(net,'SeriesNetwork')
  lgraph = layerGraph(net.Layers);
else
  lgraph = layerGraph(net);
```

```
end
```

Имена этих двух слоев заменяют вручную, или автоматически с помощью функции findLayersToReplace.

```
[learnableLayer,classLayer] = findLayersToReplace(lgraph);
[learnableLayer,classLayer]
ans =
  1x2 Layer array with layers:
    1   'loss3-classifier'   Fully Connected    1000 fully connected
layer
    2   'output'             Classification Output   crossentropyex
with 'tench' and 999 other classes
```

В большинстве сетей последний слой с learnable-весами является полносвязным слоем. Замените этот полносвязный слой на новый полносвязный слой с количеством выходных параметров, равных количеству классов в новом наборе данных. В некоторых сетях, таких как SqueezeNet, последний learnable-слой является сверточным слоем 1 × 1 вместо этого. В этом случае замените сверточный слой на новый с количеством фильтров, равных количеству классов. Чтобы учиться быстрее в новом слое, чем в переданных слоях, увеличьте факторы темпа обучения слоя [12].

```
numClasses = numel(categories(imdsTrain.Labels));
if isa(learnableLayer,'nnet.cnn.layer.FullyConnectedLayer')
    newLearnableLayer = fullyConnectedLayer(numClasses, ...
        'Name','new_fc', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);

elseif isa(learnableLayer,'nnet.cnn.layer.Convolution2DLayer')
    newLearnableLayer = convolution2dLayer(1,numClasses, ...
        'Name','new_conv', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);
end

lgraph = replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);
```

Слой классификации задает выходные классы сети. Замените слой классификации на новый без меток класса trainNetwork автоматически устанавливает выходные классы слоя в учебное время.

```
newClassLayer = classificationLayer('Name','new_classoutput');
lgraph = replaceLayer(lgraph,classLayer.Name,newClassLayer);
```

Чтобы проверить, что новые слои соединяются правильно, постройте и увеличьте масштаб последних слоев сети (рисунок 6.3).

```
figure('Units','normalized','Position',[0.3 0.3 0.4 0.4]);
plot(lgraph)
ylim([0,10])
```

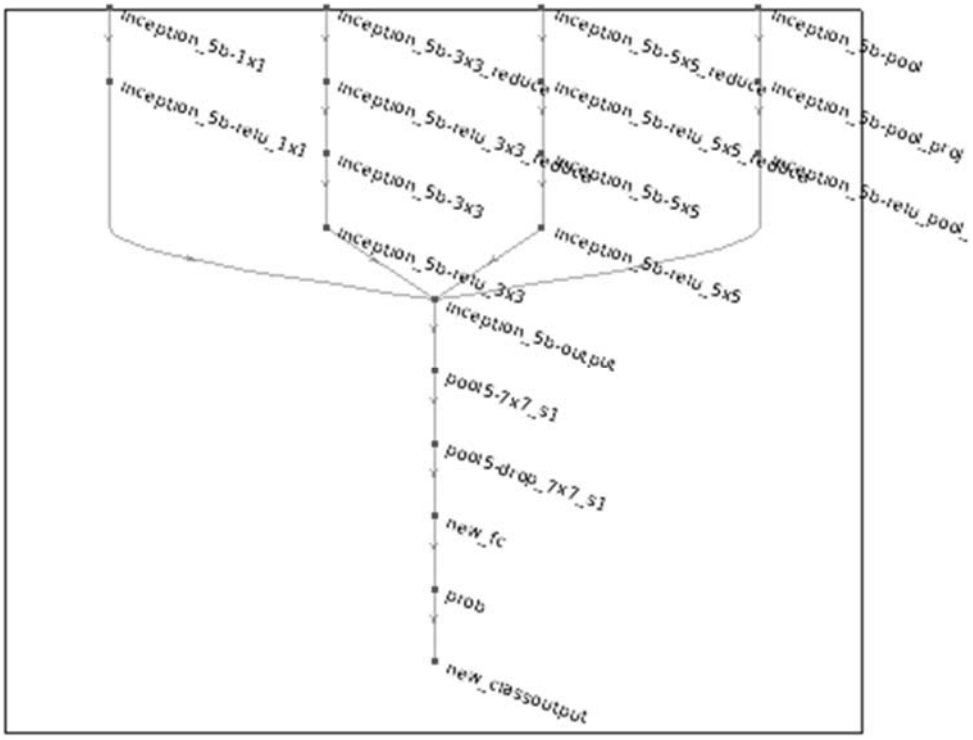


Рисунок 6.3 – Новый график слоя

Заморозка начальных слоёв.

Сеть теперь готова быть переобученной на новом наборе изображений. Опционально можно «заморозить» веса в более ранних слоях сети, обнулив темпы обучения. Во время обучения trainNetwork не обновляет параметры блокированных слоев. Поскольку градиенты блокированных слоев не должны быть вычислены, замораживание весов многих начальных слоев может значительно ускорить обучение сети. Если новый набор данных является небольшим, то замораживание более ранних сетевых слоев может также препятствовать тому, чтобы те слои полностью соответствовали новому набору данных.

Извлеките слои и связи графа слоя, которые будут замораживаться. В GoogLeNet первые 10 слоев составляют начальную основу сети. Используйте функцию поддержки freezeWeights, чтобы обнулить темпы обучения в первых 10-ти слоях. Используйте функцию поддержки createLgraphUsingConnections, чтобы повторно подключить все слои в первоначальном заказе. Новый график слоя содержит те же слои, но с темпами обучения более ранних обнуленных слоев.

```
layers = lgraph.Layers;
connections = lgraph.Connections;

layers(1:10) = freezeWeights(layers(1:10));
lgraph = createLgraphUsingConnections(layers,connections);
```

Обучение сети.

Сеть требует входных изображений размера 224 × 224 × 3, но изображения в базе данных изображений имеют различные размеры. Используйте расширенную базу данных изображений, чтобы автоматически изменить размер учебных изображений. Задайте дополнительные операции увеличения, чтобы выполнить на учебных изображениях. Случайным образом инвертируйте учебные изображения вдоль вертикальной оси и случайным образом переведите их до 30 пикселей и масштабируйте их до 10 % горизонтально и вертикально. Увеличение данных помогает препятствовать тому, чтобы сеть полностью соответствовала и запомнила точные детали учебных изображений [13].

```
pixelRange = [-30 30];
scaleRange = [0.9 1.1];
imageAugmenter = imageDataAugmenter( ...
    'RandXReflection',true, ...
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange, ...
    'RandXScale',scaleRange, ...
    'RandYScale',scaleRange);
augimdsTrain = augmentedImageDatastore(inputSize(1:2),imdsTrain, ...
    'DataAugmentation',imageAugmenter);
```

Чтобы автоматически изменить размер изображений валидации, не выполняя дальнейшее увеличение данных, используйте расширенную базу данных изображений, не задавая дополнительных операций предварительной обработки.

```
augimdsValidation = augmentedImageDatastore(inputSize(1:2),imdsValidation);
```

Задайте опции обучения. Установите InitialLearnRate на маленькое значение замедлять изучение в переданных слоях, которые уже не замораживаются. На предыдущем шаге вы увеличили факторы темпа обучения для последнего обучаемого слоя, чтобы ускорить изучение в новых последних слоях. Эта комбинация настроек темпа обучения приводит к быстрому изучению в новых слоях, медленнее учась в средних слоях и никакое изучение в ранее, блокированные слои.

Задайте номер уровня сети для обучения. При использовании обучения с переносом вы не должны обучать все уровни. Уровень обучения является полным учебным циклом на целом обучающем наборе данных. Задайте мини-пакетный размер и данные о валидации. Вычислите точность валидации уровня.

```
miniBatchSize = 10;
valFrequency = floor(numel(augimdsTrain.Files)/miniBatchSize);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',miniBatchSize, ...
    'MaxEpochs',6, ...
    'InitialLearnRate',3e-4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',augimdsValidation, ...
    'ValidationFrequency',valFrequency, ...
```

```
'Verbose',false, ...
'Plots','training-progress');
```

Обучите сеть с помощью тренировки. По умолчанию тренировочная сеть использует графический процессор, если доступны Parallel Computing Toolbox™ и CUDA®. В противном случае используется центральный процессор. Можно также задать среду выполнения при помощи пары аргументов «имя – значение» ExecutionEnvironment-trainingOptions. Поскольку набор данных является небольшим, обучение происходит быстро (рисунок 6.4) [14, 15].

```
net = trainNetwork(augimdsTrain,lgraph,options);
```

Классификация изображения валидации.

Классифицируйте изображения валидации с помощью настроенной сети и вычислите точность классификации.

```
[YPred,probs] = classify(net,augimdsValidation);
accuracy = mean(YPred == imdsValidation.Labels)
accuracy = 0.9000
```

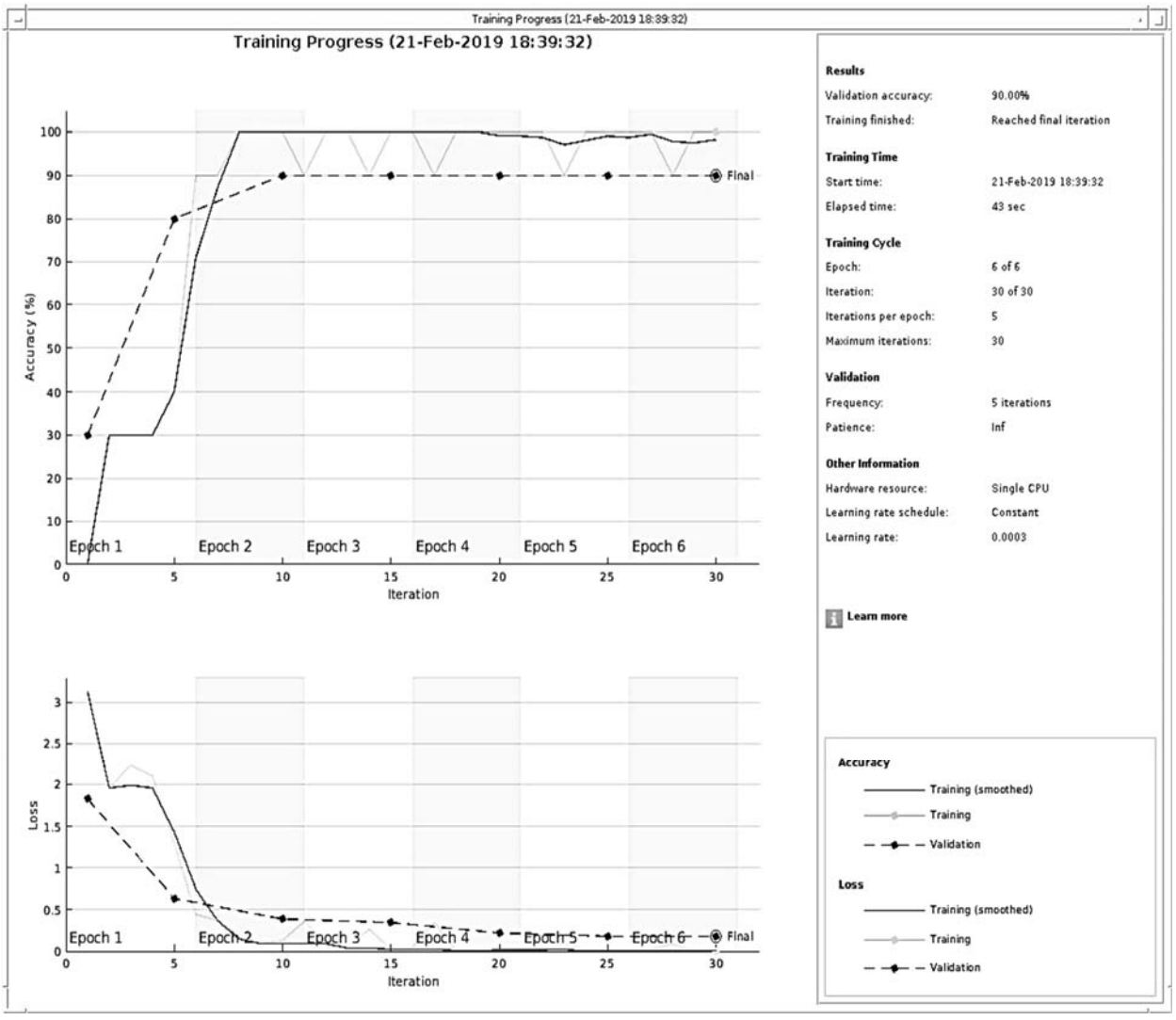


Рисунок 6.4 – График обучения нового слоя

Отобразите четыре демонстрационных изображения (рисунок 6.5) для валидации с предсказанными метками и предсказанными вероятностями изображений, имеющих метки [11].

```
idx = randperm(numel(imdsValidation.Files),4);
figure
for i = 1:4
    subplot(2,2,i)
    I = readimage(imdsValidation,idx(i));
    imshow(I)
    label = YPred(idx(i));
    title(string(label) + ", " + num2str(100*max(probs(idx(i),:)),3)
+ "%");
end
```

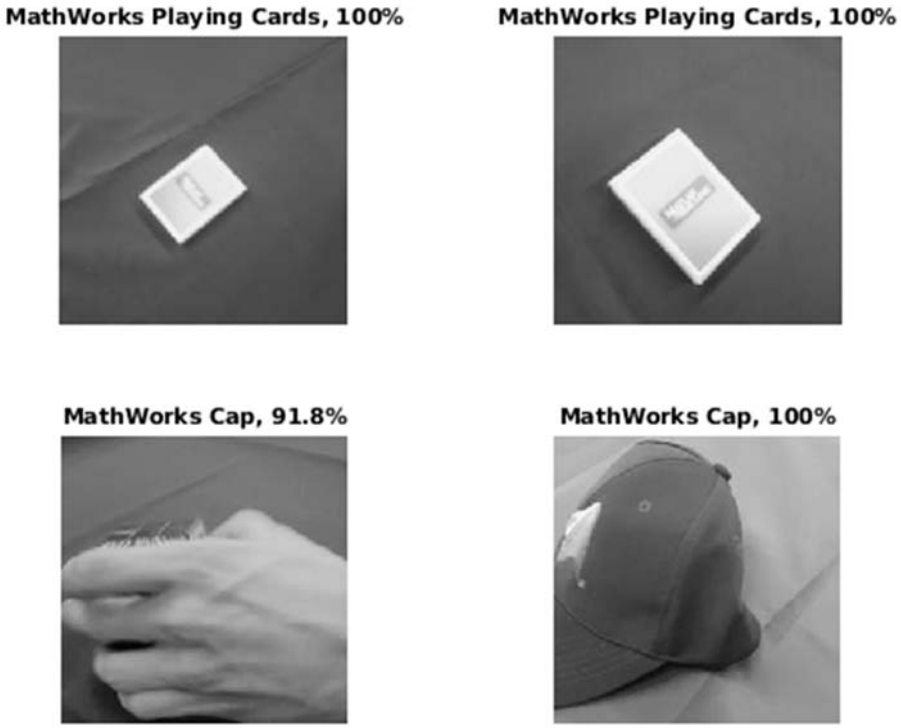


Рисунок 6.5 – Валидирующие изображения

Практическое задание

Обучить нейросеть, используя предварительно обученную, для классификации и распознавания набора изображений.

Список литературы

1 **Шапиро, Л.** Компьютерное зрение [Электронный ресурс] / Л. Шапиро, Д. Стокман. – Москва: Лаборатория знаний, 2020. – 761 с. – Режим доступа: http://www.iprbookshop.ru/89030.html. – Дата доступа: 18.10.2020.

2 **Сиденко, Л. А.** Компьютерная графика и геометрическое моделирование / Л. А. Сиденко. – Санкт-Петербург: Питер, 2012. – 224 с.

3 **Сузи, Р. А.** Язык программирования Python [Электронный ресурс]: учебное пособие / Р. А. Сузи. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ); Ай Пи Ар Медиа, 2020. – 350 с. – Режим доступа: http://www.iprbookshop.ru/97589.html. – Дата доступа: 18.10.2020.

4 **Глотова, М. Ю.** Математическая обработка информации: учебник и практикум для бакалавров / М. Ю. Глотова, Е. А. Самохвалова. – Москва: Юрайт, 2015. – 344 с.

5 **Гультяев, А.** Визуальное моделирование в среде MATLAB: учебный курс / А. Гультяев. – Санкт-Петербург: Питер, 2002. – 432 с.

6 Краткий курс теории обработки изображений [Электронный ресурс] // Экспонента. – Москва, 2020. – Режим доступа: https://hub.exponenta.ru/post/kratkiy-kurs-teorii-obrabotki-izobrazheniy734. – Дата доступа: 18.10.2020.

7 Маркировка Пикселей для Семантической Сегментации [Электронный ресурс] // Экспонента. – Москва, 2020. – Режим доступа: https://docs.exponenta.ru/vision/ug/label-pixels-for-semantic-segmentation.html. – Дата доступа: 18.10.2020.

8 **Rother, C.** GrabCut – Interactive Foreground Extraction using Iterated Graph Cuts / C. Rother, V. Kolmogorov, A. Blake // ACM Transactions on Graphics (SIGGRAPH). – 2004. – Vol. 23, № 3. – P. 309–314.

9 Отслеживание нескольких объектов на основе движения [Электронный ресурс] // Экспонента. – Москва, 2020. – Режим доступа: https://docs.exponenta.ru/R2019b/vision/examples/motion-based-multiple-object-tracking.html. – Дата доступа: 18.10.2020.

10 Отслеживание и оценка движения [Электронный ресурс] // Экспонента. – Москва, 2020. – Режим доступа: https:// docs.exponenta.ru/vision/tracking-and-motion-estimation.html. – Дата доступа: 18.10.2020.

11 Отслеживание [Электронный ресурс] // Экспонента. – Москва, 2020. – Режим доступа: https://docs.exponenta.ru/R2019b/ vision/ug/multiple-object-tracking.html. – Дата доступа: 18.10.2020.

12 Глубокое обучение в MATLAB [Электронный ресурс] // Экспонента. – Москва, 2020. – Режим доступа: https:// docs.exponenta.ru/R2019a/ deeplearning/ug/deep-learning-in-matlab.html. – Дата доступа: 18.10.2020.

13 Модель BVLC GoogLeNet [Электронный ресурс]. – Режим доступа: https://github.com/ BVLC/ caffe/ tree/master/models/bvlc_googlenet. – Дата доступа: 18.10.2020.

14 Обнаружение объектов, используя глубокое обучение YOLO v2 [Электронный ресурс] // Экспонента. – Москва, 2020. – Режим доступа: https://docs.exponenta.ru/vision/ug/train-an-object-detector-using-you-only-look-

once.html. – Дата доступа: 18.10.2020.

15 Обучите нейронную сеть для глубокого обучения классифицировать новые изображения [Электронный ресурс] // Экспонента. – Москва, 2020. – Режим доступа: https://docs.exponenta.ru/R2019a/ deeplearning/examples/train-deep-learning-network-to-classify-new-images.html. – Дата доступа: 18.10.2020.