МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический
университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)
Чистопольский филиал «Восток»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ

по дисциплине

ОБРАБОТКА ИЗОБРАЖЕНИЙ

Индекс по учебному плану: **Б1.В.ДВ.08.02**
Направление подготовки: **09.03.01 Информатика и вычислительная техника**
Квалификация: **Бакалавр**
Профиль подготовки: **Вычислительные машины, комплексы, системы и сети**
Типы задач профессиональной деятельности: **проектный, производственно-технологический**

Рекомендовано УМК ЧФ КНИТУ-КАИ

Чистополь
2023 г.

Лабораторная работа №1

Преобразование изображений

Алгоритм преобразования в черно-белое изображение

Каждый пиксель изображения формируется при помощи сочетания четырех каналов: ARGB (Alpha, Red, Green, Blue), альфа-канала, красного, зеленого и синего.

Альфа-канал отвечает за прозрачность пикселя (100% — пиксель полностью непрозрачный, 0% — полностью прозрачный).

Сочетание значений RGB каналов определяет цвет пикселя.

Каждый канал несёт в себе 8 бит информации (1 байт), соответственно интенсивность канала может меняться в диапазоне от 0 до 255. Полностью пиксель занимает в памяти 32 бита (4 байта).

Для того, чтобы **преобразовать цветное изображение в черно-белое**, нужно найти среднее арифметическое значение R, G и B каналов пикселя и затем это значение присвоить RGB каналам этого же пикселя (то есть оно будет одинаковое). Альфа-канал оставляем без изменений.

Такую операцию нужно проделать с каждым пикселем изображения.

Программа для преобразования цветного изображения в черно-белое

Приступим к написанию приложения Windows Forms в среде разработки Visual Studio на языке программирования C#.

Создадим новый проект программы и разместим на форме 3 кнопки (Button) и два контейнера под изображения PictureBox. Кнопки понадобятся для: открытия изображения, преобразования в черно-белое (в оттенки серого) и сохранения преобразованной картинки на компьютер.

Свойства кнопок Text изменим на:

Открыть

Ч/Б

Сохранить

Имена кнопок (Name), используемые при разработке, изменим на:

openButton

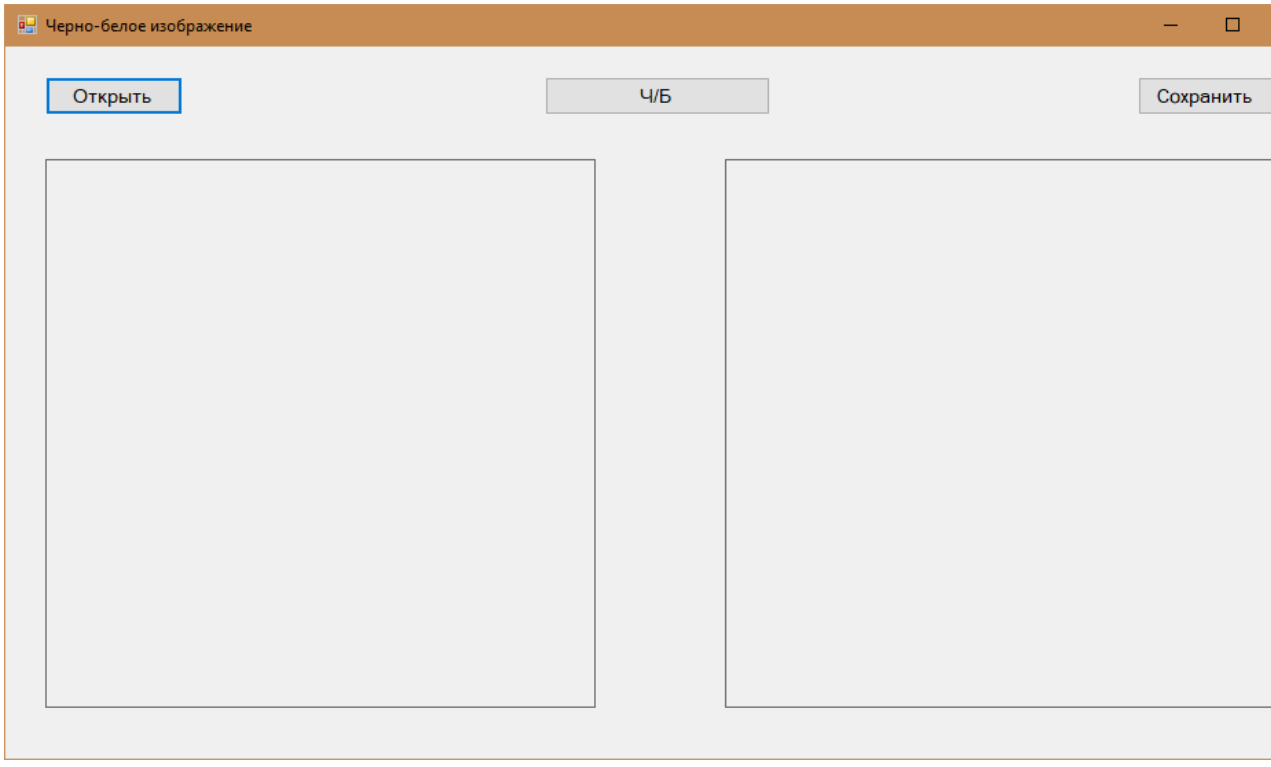
grayButton

saveButton

Размеры обоих PictureBox (Size) сделаем равными 400 x 400 пикселей.

Чтобы вокруг границ PictureBox отображалась рамка в виде сплошной тонкой линии, установим значение свойства BorderStyle в положение FixedSingle.

Также у PictureBox свойство SizeMode сделаем равным Zoom. Данная настройка позволит автоматически масштабировать отображаемые изображения в соответствии с размером PictureBox, при этом сохраняя соотношения сторон исходной картинки.



Перейдём к написанию кода. Сначала закодируем обработку нажатий кнопок «Открыть» и «Сохранить»:

// кнопка Открыть

```
        private void openButton_Click(object sender, EventArgs e)
        {
            // диалог для выбора файла
            OpenFileDialog ofd = new OpenFileDialog();
            // фильтр форматов файлов
            ofd.Filter = "Image Files(*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG|All files (*.*)|*.*";
            // если в диалоге была нажата кнопка ОК
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    // загружаем изображение
                    pictureBox1.Image = new Bitmap(ofd.FileName);
                }
                catch // в случае ошибки выводим MessageBox
                {
                    MessageBox.Show("Невозможно открыть выбранный файл", "Ошибка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }

        // кнопка Сохранить
        private void saveButton_Click(object sender, EventArgs e)
        {
            if (pictureBox2.Image != null) // если изображение в pictureBox2 имеется
```

```
        {
            SaveFileDialog sfd = new SaveFileDialog();
            sfd.Title = "Сохранить картинку как...";
            sfd.OverwritePrompt = true; // показывать ли "Перезаписать файл" если пользователь указывает имя файла, который уже существует
            sfd.CheckPathExists = true; // отображает ли диалоговое окно предупреждение, если пользователь указывает путь, который не существует
            // фильтр форматов файлов
            sfd.Filter = "Image Files(*.BMP)|*.BMP|Image Files(*.JPG)|*.JPG|Image Files(*.GIF)|*.GIF|Image Files(*.PNG)|*.PNG|All files (*.*)|*.*";
            sfd.ShowHelp = true; // отображается ли кнопка Справка в диалоговом окне
            // если в диалоге была нажата кнопка ОК
            if (sfd.ShowDialog() == DialogResult.OK)
            {
               try
               {
                  // сохраняем изображение
                  pictureBox2.Image.Save(sfd.FileName);
               }
               catch // в случае ошибки выводим MessageBox
               {
                  MessageBox.Show("Невозможно сохранить изображение", "Ошибка",
                      MessageBoxButtons.OK, MessageBoxIcon.Error);
               }
            }
        }
```

}

Подробнее про открытие и сохранение картинки из PictureBox можно прочитать в соответствующих статьях на нашем сайте:

Как загрузить картинку в PictureBox C#

Сохранить изображение из PictureBox C#

Теперь напишем код кнопки, преобразующей цветное изображение в оттенки серого:

```
// кнопка Ч/Б
private void grayButton_Click(object sender, EventArgs e)
{
  if (pictureBox1.Image != null) // если изображение в pictureBox1 имеется
    {
      // создаём Bitmap из изображения, находящегося в pictureBox1
      Bitmap input = new Bitmap(pictureBox1.Image);
      // создаём Bitmap для черно-белого изображения
      Bitmap output = new Bitmap(input.Width, input.Height);
      // перебираем в циклах все пиксели исходного изображения
      for (int j = 0; j < input.Height; j++)
        for (int i = 0; i < input.Width; i++)
        {
          // получаем (i, j) пиксель
          UInt32 pixel = (UInt32)(input.GetPixel(i, j).ToArgb());
          // получаем компоненты цветов пикселя
          float R = (float)((pixel & 0x00FF0000) >> 16); // красный
          float G = (float)((pixel & 0x0000FF00) >> 8); // зеленый
          float B = (float)(pixel & 0x000000FF); // синий
          // делаем цвет черно-белым (оттенки серого) - находим среднее арифметическое
```

```
                R = G = B = (R + G + B) / 3.0f;
                // собираем новый пиксель по частям (по каналам)
                UInt32 newPixel = 0xFF000000 | ((UInt32)R << 16) | ((UInt32)G
<< 8) | ((UInt32)B);
                // добавляем его в Bitmap нового изображения
                output.SetPixel(i, j, Color.FromArgb((int)newPixel));
            }
        // выводим черно-белый Bitmap в pictureBox2
        pictureBox2.Image = output;
    }
}
```

Отдельно следует сказать про хранение значения пикселя в программе и про получение значений отдельных каналов.

Пиксель мы храним в структуре UInt32. Она предназначена для хранения беззнакового целого 4-х байтового (32 бита) числа. Диапазон значений такого числа от 0 до 232 — 1; такой диапазон идеально подходит для хранения значения пикселя — он туда помещается полностью, поскольку занимает также 32 бита.

Для получения значения конкретного канала используется побитовое умножение значения пикселя на битовую маску для соответствующего цвета с последующим сдвигом вправо на нужное количество бит (строки 17-19).

Сборка пикселя из каналов воедино происходит с использованием операций сдвига влево (для каждого канала) и последующим логическим сложением всех компонент (строка 23). Альфа-канал примем равным 0xFF000000 (FF16 = 25510).

Проверим работу программы:


Черно-белое изображение
Открыть
Ч/Б
Сохранить
CODE
VS
.RU

Лабораторная работа №2

Моделирование процесса искажения и восстановления изображений

Создать программную систему для проведения распознавания изображений в условиях наличия смещений и шумов методом двумерных корреляций. Исследовать влияние шумов на результат распознавания.

Порядок выполнения работы:

- подготовить эталонные изображения.
- оценить меру их непохожести.

На основе эталонных изображений путем геометрических преобразований (смещений) и генерации гауссова шума создать входные изображения.

Провести распознавания входных изображений с помощью двумерной корреляции при различном уровне шума.

Получить график зависимости вероятности правильного распознавания от величины сигнал/шум.

Теоретические сведения и пояснения к работе

Распознавание изображений путем двумерной корреляции заключается в нахождении наиболее похожей пары изображений: эталон B_{0_i} – входное изображение B, i=1,…,n_0. Под максимальным сходством понимается определение минимального (в общем случае экстремального) значения некоторой меры сходства $\rho(B, B_{0_i})$. Распознавание в условиях геометрических искажений осуществляется на основании следующего правила:

$$\min_{i\in\{1,\ldots,k\}} \min_{g\in G} \rho(B, gB_{0_i}) \le \delta_B, \qquad (4.1)$$

где G – группа предполагаемых искажений (в данном случае группа смещений);

δ_B – некоторая пороговая величина, введение которой объясняется наличием на входном изображении помех.

Меры сходства между изображениями может определяться по одной из следующих формул:

$$\rho(B, B_0) = \sum_{(x,y)\in D} |B(x,y) - B_0(x,y)|, \qquad (4.2)$$

$$\rho(B, B_0) = 1 - \frac{\sum_{(x,y)\in D} \left(B(x,y) - \overline{B}(x,y)\right)\left(B_0(x,y) - \overline{B}_0(x,y)\right)}{\sqrt{\sum_{(x,y)\in D} \left(B(x,y) - \overline{B}(x,y)\right)^2 \left(B_0(x,y) - \overline{B}_0(x,y)\right)^2}}, \qquad (4.3)$$

$$\rho(B, B_0) = \frac{\sum_{(x,y)\in D} |B(x,y) - B_0(x,y)|}{N}, \qquad (4.4)$$

где $\overline{B}$, $\overline{B}_0$ – средние значения яркостей изображений B и B_0 соответственно;

N – количество точек области D.

В качестве эталонных изображений в работе предлагается рассматривать сегментированные полутоновые изображения, отцентрированные по центру тяжести.

Эталонные изображения должны соответствовать условию непохожести:

$\rho(B_0^i, B_0^j) > \delta_0$, ($\forall$ i, j =1,…, n_0; кроме i=j)

где n_0 – количество эталонных изображений;

δ_0 – некоторая пороговая величина.

Величина сигнал/шум определяется следующим образом

$$\mu_A = \frac{\overline{B}(x,y)}{\delta},$$

где $\overline{B}(x,y)$ – средняя яркость изображения, δ – среднеквадратическое отклонение гауссова шума (случайная помеха, распределенная по нормальному закону с нулевым математическим ожиданием и задаваемой величиной дисперсии).

Вероятность правильного распознавания при заданном уровне шума определяется путем многократного повторения экспериментов с данным уровнем шума для разных изображений путем деления количества правильных результатов на количество всех экспериментов.

Лабораторная работа №3
Фрактальное сжатие

Цель задания

Необходимо написать программу, которая умеет сжимать изображение и распаковывать его, используя классический алгоритм фрактального сжатия, описанный на лекциях.

Задача - достичь как можно большего сжатия при использовании сетки фиксированного размера (4x4, 8x8 и 16x16 по параметру) и как можно лучшего соотношения степени сжатия к качеству для варианта с квадродеревом.

Мерой потерь будет служить PSNR - метрика обратно логарифмически пропорциональная среднеквадратичному отклонению пикселов.

Замер будет проводиться в двух номинациях:

- Сжатие с использованием сетки 4x4, 8x8 и 16x16 (допустим ТОЛЬКО классический алгоритм, который был рассмотрен на лекции).
- Сжатие с использованием квадродерева и указанием качества.

Для повышения соотношения степени сжатия и качества рекомендуется использовать следующие методы:

1. Использование перевода в YUV и дискретизацию в 4 раза цветовых компонент U & V.
2. Использование квадродерева - 16x16 - 8x8 - 4x4 с дроблением очередного блока, если его приближение (например), хуже среднего значения для всех остальных блоков. Другой вариант - просто по порогу, пропорционально пересчитываемому для каждого уровня квадродерева.
3. Использование арифметического сжатия для дожатия афинных коэффициентов (в первую очередь сдвига и контрастности).

Первая часть задания

Программа должна уметь получать на вход BMP файл и по опции -c - сжимать его, по опции -d распаковывать его.

Чтение BMP из файла проще возложить на систему - посмотрите в MSDN "LoadImage":HBITMAP image = (HBITMAP) LoadImage (0,"image.bmp",IMAGE_BITMAP,0,0,LR_LOADFROMFILE);

Пример простой программы, которая работает с BMP также входит в комплект VS.

По умолчанию должен использовать фрактальный алгоритм разбиением изображения на сетки 4x4, 8x8 и 16x16.

Программа должна быть консольным приложением и запускаться так:

```
compress -c image.bmp out_file.fr -s 8
compress -d out_file.fr out_image.bmp
```

Т.е. тестовый файл test.bat вида:

```
compress -c image1.bmp out_file1.fr -s 8
compress -d out_file1.fr out_image1.bmp
compress -c image2.bmp out_file2.fr -s 8
compress -d out_file2.fr out_image2.bmp
compress -c image3.bmp out_file3.fr -s 8
compress -d out_file3.fr out_image3.bmp
compress -c image4.bmp out_file4.fr -s 8
compress -d out_file4.fr out_image4.bmp
compress -c image5.bmp out_file5.fr -s 8
compress -d out_file5.fr out_image5.bmp
```

(обратите внимание - минус перед параметром означает, что это опция)

должен сжимать 5 тестовых изображений (image1.bmp, ... image5.bmp). Для оценки степени сжатия используются размеры каждого сжатого

изображения, а для оценки качества - разница между исходным и распакованным изображениями.

Вторая часть задания

Для повышения соотношения качества к размеру (и набора дополнительных баллов) нужно реализовать работу с квадродеревом и управление качеством.

С ним ваша программа должна уметь выполнять тестовый файл test_qual.bat вида:

```
compress -c image1.bmp out_file1.fr -q 10
compress -d out_file1.fr out_image1.bmp
compress -c image2.bmp out_file2.fr -q 10
compress -d out_file2.fr out_image2.bmp
compress -c image3.bmp out_file3.fr -q 10
compress -d out_file3.fr out_image3.bmp
compress -c image4.bmp out_file4.fr -q 10
compress -d out_file4.fr out_image4.bmp
compress -c image5.bmp out_file5.fr -q 10
compress -d out_file5.fr out_image5.bmp
```

При этом q - параметр качества должен изменяться от 1 до 100 (включительно).

Для тестов будут использованы 4-6 значений, условно 10, 30, 50, 70, 85, 100.

Лабораторная работа №4

Выделение признаков изображений, распознавание образов

В последнее время стала актуальной задача нахождения лиц людей на статических изображениях и в видеопоследовательности. Большая часть методов поиска человеческих лиц на первом этапе производит обработку изображения при помощи цветовой сегментации.

Методы поиска лиц людей на изображении обычно делятся на два этапа:

Предобработка изображения – выделение пикселов близких по цвету к коже человека посредством цветовой сегментации.

Постобработка изображения, с использованием результатов полученных на первом этапе и априорного знания структуры человеческого лица.

От того насколько точно на этапе предобработки будут определены области на изображении, относящиеся к коже, в значительной мере зависит точность и достоверность конечного результата.

Основными проблемами при определении цвета кожи являются:

- зависимость цвета от условий съемки (например, освещения);
- зависимость получаемых цветовых значений объектов от конкретной камеры;
- многообразие оттенков кожи.

Задача цветовой сегментации кожи на изображении заключается в оценке близости цвета каждого пиксела к оттенку кожи. Определить понятие «оттенок кожи» строго математически не представляется возможным, так как оно основано на человеческом восприятии цвета.

На данный момент существует целый ряд подходов к решению этой задачи. Суть всех методов заключается в описании некоторой модели

распределения цвета кожи, с помощью которой и оценивается принадлежность цвета коже.

Разработка модели цвета кожи в общем случае разбивается на три этапа:

- накопление тренировочных (обучающих) знаний, используя пробные изображения, на которых самостоятельно указывают область «кожи» и «не-кожи». По этим данным накапливается статистика оттенков кожи и в ряде методов гистограмма оттенков, к коже не относящихся («не-кожи»);
- обработка полученной статистики и выбор параметров модели цвета кожи, для последующего использования. Выбор критериев оценки принадлежности пикселей к области кожи;
- обработка входящих данных, используя накопленные знания и построенную цветовую модель.

Существующие методы разбиты на три группы:

Априорные методы

К группе априорных относятся методы, которые в качестве модели цвета кожи используют явно заданный многогранник в заданном цветовом пространстве, т.е. заранее задается некоторый жесткий набор правил.

Цвет (R, G, B) причисляется к коже, если выполнены следующие соотношения:

$R > 95$ and $G > 40$ and $B < 20$ and

$\max\{R, G, B\} - \min\{R, G, B\} > 15$

and $|R - G| > 15$ and $R > G$ and $R > B$.

or

$R > 220$ and $G > 210$ and $B > 170$ and

$|R - G| \leq 15$ and

$R > B$ and $G > B$

Непараметрические методы

Основная идея таких методов состоит в том, что при оценке распределения цвета кожи на основе тренировочных данных не используется явная модель цвета кожи. В результате конструируется *вероятностная карта кожи* – каждому значению цвета в дискретизованном цветовом пространстве ставится в соответствие вероятность принадлежности этого цвета коже.

Нормализованная таблица частот (LUT)

Этот метод распознавания кожи строит нормализованную таблицу частот по обучающим данным и на основании (5) вероятность принадлежности цвета *c* коже выглядит следующим образом:

$$P_{skin}(c) = \frac{skin(c)}{K},$$

где:skin(c) – значение счетчика цвета c;

K – нормализующий коэффициент

В качестве коэффициента *K* может использоваться сумма всех значений счетчиков или только их максимальное значение . Далее вероятность принадлежности цвета *c* коже сравнивается с заданным порогом.

Параметрические методы

Моделирование с помощью нормального распределения

Предлагается моделирование распределения цвета кожи с помощью нормального распределения.

Эллиптическая модель

Эллиптическая модель была призвана улучшить результаты применения нормального распределения. В соответствии с этой модельюпиксел цвета *c* принадлежит коже, если $\Phi(c) \leq \theta$, где θ – заданный порог, а $\Phi(c)$ определяется по формулам (11), (12). Авторы заявляют, что

эллиптическая модель приближает цвет кожи лучше, чем нормальное распределение, т.к. ориентация эллипса не влияет на определение центра распределения.

Преимущества параметрических методов:

Компактная модель

Позволяют исключить яркостную компоненту

Способность обобщать и интерполировать недостающие данные

Недостатки:

Зависят от выбора цветового пространства, т.к. в модели заложено предположение о форме распределения цвета кожи

Высокий уровень ложного обнаружения

Практическая часть лабораторной работы состоит из следующих этапов:

- курсовые градиентные операторы;
- нелинейные системы контрастирования изображений, метод робертса, метод собела;
- статистический метод.
- сегментация изображений;

По заданию преподавателя выбрать изображение, провести следующие преобразования:

- курсовые градиентные операторы;
- нелинейные системы контрастирования изображений, метод робертса, метод собела;
- статистический метод.
- сегментация изображений;

Провести сравнительный анализ результатов сегментации изображения.

Лабораторная работа №5

Классификация на основе байесовской теории решений

Предобработка изображения – выделение пикселов близких по цвету к коже человека посредством цветовой сегментации.

Постобработка изображения, с использованием результатов полученных на первом этапе и априорного знания структуры человеческого лица.

Классификатор Байеса - более совершенный метод, использующий моделирование цвета кожи с помощью двух таблиц частот. Вероятность принадлежности цвета *c* коже определяется по формуле:

$$P(c \mid skin) = \frac{P(skin \mid c)P(skin)}{P(skin \mid c)P(skin) + P(\neg skin \mid c)P(\neg skin)}$$

Вероятности *P(skin|c), P(¬ skin|c)* вычисляются на основе гистограмм кожи и не-кожи. Вероятности *P(skin), P(¬ skin)* вычисляются как отношение количества пикселов цвета соответственно кожи и не-кожи на обучающих изображениях к общему количеству пикселов .

Преимущества непараметрических методов:

Простота реализации

Высокая скорость обучения

Модель не зависит от формы распределения цвета кожи в отличие от остальных методов

Высокий уровень правильного распознавания, особенно у классификатора Байеса.

Недостатки:

Требуют больших обучающих наборов

Хранение таблицы частот требует много памяти

По заданию преподавателя выбрать изображение, провести преобразования с использованием классификатора Байеса

Провести анализ результатов сегментации изображения.

Лабораторная работа №6

Распознавание образов на основе нейронных сетей

Пусть десятичные цифры 0-9 отображаются на бинарной матрице размерностью 8x8 в виде, представленном на рис. 1.

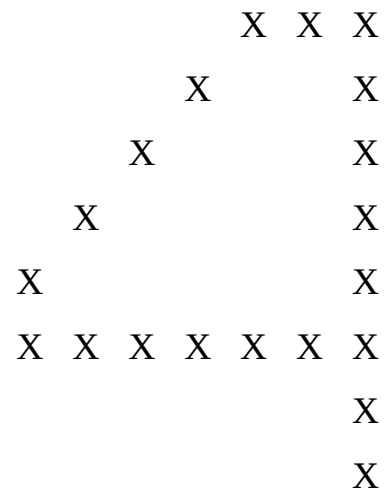


Рис 1.

Задача нейронной сети заключается в распознавании таких цифр в условиях помех. Помехами в этом случае могут являться случайные появления или пропадания отдельных пикселов в бинарной матрице, изменения формы.

Для распознавания цифр можно обучить нейронную сеть, состоящую из 64 входных нейронов и 10 выходных.

Входами нейронной сети будут являться элементы 0 или 1, в зависимости от того, черная или белая клетка используется в соответствующей битовой матрице.

Каждый выходной нейрон будет ответственен за какую либо цифру от 0 до 9. Например, первый – за 1, второй – за 2, … десятый – за 0.

Значениями выходов нейронной сети также будут являться числа 0 или 1. Активность i-ого выходного нейрона означает похожесть входного образа на i-ую цифру.

Задание на лабораторную работу

Подготовить 20 битовых масок, демонстрирующих примеры написания цифр – по 2 маски для каждой цифры. Данные маски показать преподавателю.

Подготовить обучающую выборку для обучения нейронной сети, распознающей цифры. Обучающая выборка должна включать 64 входа и 10 выходов.

Обучить в пакете Neural Network Wizard нейронную сеть для распознавания цифр.

Подготовить по 2 тестовых примера для каждой цифры. Тестовые примеры должны включать шумовые элементы (появление или пропадание пикселей в битовой маске).

Исследовать работу нейронной сети с зашумленными образами. Насколько корректно она распознает зашумленные цифры? Тестовые битовые маски и точность их распознавания показать преподавателю

Основная литература:

1. Волкова, М.А. Методы обработки и распознавания изображений. Учебно-методическое пособие по лабораторному практикуму. [Электронный ресурс] / М.А. Волкова, В.Р. Луцив. — Электрон. дан. — СПб. : НИУ ИТМО, 2016. — 40 с. — Режим доступа: https://e.lanbook.com/reader/book/91416/#1— Загл. с экрана.

2. Ростовцев, В. С. Искусственные нейронные сети : учебник / В. С. Ростовцев. — Санкт-Петербург : Лань, 2019. — 216 с. — ISBN 978-5-8114-3768-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/122180 — Режим доступа: для авториз. пользователей.

3. Вакуленко, С. А. Практический курс по нейронным сетям : учебное пособие / С. А. Вакуленко, А. А. Жихарева. — Санкт-Петербург : НИУ ИТМО, 2018. — 71 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/136500 — Режим доступа: для авториз. пользователей.

Дополнительная литература

1. Фисенко, В.Т. Компьютерная обработка и распознавание изображений. Методические указания к лабораторным работам. [Электронный ресурс] / В.Т. Фисенко, Т.Ю. Фисенко. — Электрон. дан. — СПб. : НИУ ИТМО, 2008. — 42 с. — Режим доступа: http://e.lanbook.com/book/40794 — Загл. с экрана.

2. Ежова, К.В. Моделирование и обработка изображений. [Электронный ресурс] — Электрон. дан. — СПб. : НИУ ИТМО, 2011. — 93 с. — Режим доступа: https://e.lanbook.com/reader/book/40820/#1— Загл. с экрана.

3. Гонсалес, Р. Цифровая обработка изображений. [Электронный ресурс] / Р. Гонсалес, Р. Вудс. — Электрон. дан. — Москва : Техносфера,

2012. — 1104 с. — Режим доступа: http://e.lanbook.com/book/73514 — Загл. с экрана.