ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАНА
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИИ

К защите
Зав. Кафедрой «ПОИТ»
Проф. Нишонов А.Х.
_____
«___»_____2013 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему: Разработка программного обеспечения и алгоритма сравнения изображении

Выпускник _____ __Махмудов А.А.__
___
(имзо)
Руководитель _____ __Рахматуллаев З.М._
___
(имзо)
Рецензент _____ _____
___ ___
(имзо)
Консультант по БЖД _____ _____
___ ___
(имзо)

Ташкент 2013

СОДЕРЖАНИЕ

МАЗМУННОМА

Битирув малакавий ишининг мақсади тасвирларни солиштириш алгоритми ва дастурий таъминотини яратишдан иборат. Ушбу ишда тасвирларни солиштириш алгоритми ва унинг турлари ҳамда геометрик модел бўйича тасвирларни синтезлаш ва тасвирларни сиқиш сифатини текшириш кўрилган. Дастурий таъминотни ишлаб чиқишда Delphi дастурлаш тили воситаларидан фойдаланилган.

АННОТАЦИЯ

Целью выпускной квалификационной работы является разработкой программного обеспечения и алгоритма сравнения изображении. В работе рассмотрено алгоритм сравнения изображении и его виды, а также применение при синтезе изображений по геометрической модели, а также при контроле качества сжатия. Программное обеспечние реализовано с использованием средств языка программирование Delphi.

SUMMARY

The purpose of the final qualifying work is the development of software and image comparison algorithm. In this paper we considered the image comparison algorithm and its types, and application in the synthesis of images on the geometric model, and the quality control of compression. Software; realized with the use of the programming language Delphi.

ВВЕДЕНИЕ

На сегодняшний день основной из задач сфере информационных технологий является подготовка квалифицированных кадров в области обработки изображение. В данной работе рассматривается вопрос разработки алгоритм и программного обеспечения цифровой обработки изображений, сравнения изображении. Эти алгоритмы используется в задачах сравнение изображений, идентификация, поиск изображения, чтобы облегчить трудности и экономит время работы.

Как отметил Президент Республики Узбекистан И.А.Каримов, одним из важнейших приоритетов выхода из мирового кризиса является дальнейшее развитие производственной и социальной инфраструктуры как важнейшего фактора модернизации страны и увеличения занятости населения.[1].

Это вызвано рядом причин:

- развитие инфраструктуры создает необходимые благоприятные условия для размещения новых предприятий и развития экономики в целом, расширяет доступ к освоению богатых минерально-сырьевых ресурсов страны;
- развитая система производственной инфраструктуры, прежде всего автомобильных и железных дорог, эффективное их функционирование служит важнейшим условием и фактором снижения общих издержек производства, что повышает конкурентоспособность производимой продукции и в целом нашей экономики;
- развитие социальной инфраструктуры, обеспечение населения чистой питьевой водой, системой энергообеспечения, строительство объектов социальной сферы в конечном итоге направлено на повышение качества жизни населения;
- развитие инфраструктуры является обширной сферой приложения труда, что позволяет создать новые рабочие места и обеспечить

занятость населения, особенно молодежи, повысить уровень доходов и благосостояния людей. [1].

Вместе с тем, решение научных и инженерных задач при работе с визуальными данными требует особых усилий, опирающихся на знание специфических методов, поскольку традиционная идеология одномерных сигналов и систем мало пригодна в этих случаях. Сравнивать изображения начали в середине 60-х годов, то есть почти сразу, как появилась компьютерная графика. Изображения тогда были бинарные, в лучшем случае полутоновые. Поэтому к ним стали относиться как к дискретным двумерным сигналам, и перенесли на них стандартные в области обработки сигналов метрики, такие как среднеквадратичная ошибка (MSE), отношение сигнал-шум (SNR), пиковое отношение сигнал-шум (PSNR).

Попытки измерять цвет предпринимались еще в прошлом веке. Существенные результаты были достигнуты к 1931 году, когда Международная Комиссия по Освещению (CIE) приняла стандарт измерения цвета. Так появилось стандартное цветовое пространство CIE XYZ. Оно позволяло, задав всего 2 числа, однозначно определить оттенок цвета, а, добавив еще одно число, яркость, однозначно определить любой цвет. Однако оно не давало возможности определить, насколько похожи два заданных цвета, и работы по поиску цветовых пространств, удовлетворяющих этим условиям, были продолжены. Они завершились принятием той же комиссией в 1971 году пространств CIE Luv и CIE Lab, равномерных с точки зрения восприятия.

Работы по изучению и описанию физиологии человеческого зрения идут уже довольно долго. Среди прочих наиболее интересной является работа Mannos и Sakrison [2], которые в 1974 году изучили зависимость восприятия изменения яркости, от однородности фона, и предложили ее математическое описание – функцию чувствительности контраста. Спустя примерно 10 лет в распознавании изображений было применено двумерное Габоровское преобразование, которое, как оказалось, хорошо моделирует

преобразование сигнала, происходящее в коре головного мозга. Его существенным недостатком является то, что для него отсутствует эффективная схема вычисления. Поэтому следующий шаг был сделан в 1987 году, когда Watson [3] предложил преобразование, у которого базисные функции были похожи на Габоровские, но допускали эффективную реализацию.

Основные задачи исследования**.** Современная область применения сравнения изображений очень велика. Изображения приходится сравнивать в системах распознавания образов. Это может потребоваться при обработке запросов к базам данных, содержащим изображения, при синтезе изображений по геометрической модели (так называемый рендеринг), для автоматического управления этим процессом. Также сравнивать изображения надо для автоматического управления сжатием изображений, когда алгоритм позволяет контролировать качество сжатого изображения.

Требования к алгоритмам сравнения сильно различаются, в зависимости от области применения. Так в базах данных, в системах Query by Example (запрос по примеру), сравнение может быть не очень точным, но должно проводиться за 10^{-3}с. – 10^{-4}с., кроме того алгоритм должен быть нечувствителен к сдвигам изображения. В системах управления синтезом изображений время не столь существенно, зато очень важна точность сравнения, при этом алгоритм может быть чувствителен к сдвигам и другим линейным преобразованиям изображений, так как они нехарактерны для этой задачи. В системах управления сжатием требования к алгоритму будут примерно такие же, как и в случае синтеза, только алгоритм должен быть нечувствителен к малым сдвигам изображения.

В случае запросов к базам данных, для сравнения изображений могут использоваться достаточно грубые алгоритмы. Их достоинство заключается в их простоте. Они позволяют делать порядка 10.000 сравнений в секунду. Так в системе QBIC фирмы IBM, представленной в работе [4], для поиска изображений используются критерии типа процентного содержания

заданных цветов в изображении, близости гистограмм содержания цветов, и приблизительного расположения цветов на картинке. Иногда используются более сложные алгоритмы сравнения, использующие дискретное преобразование Фурье, или вейвлет-преобразование. В таком случае в базе данных вместе с изображением хранится его представление, а для поиска изображений создается представление-запрос, и ищутся представления наиболее похожие на него. Такая система описывается в статье Jacobs и др.[5]. В ней представление строится на основе 60 наибольших по модулю вейвлет-коэффициентов, которые затем квантуются до значений 0, 1 или -1. Эта система позволяет использовать в качестве запроса к базе данных сделанный в графическом редакторе набросок изображения, либо отсканированный уменьшенный вариант картинки.

Целью выпускной квалификационной работы является разработать программного обеспечения и алгоритма сравнения изображении.

Исходя из требований, предъявляемых к методам сравнения, при синтезе изображений по геометрической модели, а также при контроле качества сжатия, была поставлена следующая задача. Требовалось создать метод сравнения изображений, который обеспечивал бы правильные результаты для любого монитора, позволял указать области, в которых отличия будут заметны, а также выдавал обобщенную характеристику, показывающую насколько похожи сравниваемые изображения.

Поскольку при сжатии изображений, и синтезе изображений по геометрической модели результаты обычно рассматриваются на мониторе, в решении не рассматривался случай сравнения изображений на отражающих поверхностях кроме информативных точек1.

Подходы к решению: В современных работах по сравнению изображений основной упор делается либо на учет ошибок цветопередачи [5,9], либо на учет зависимости восприятия цвета от однородности окружающей области [8,9]. В моей работе делается попытка объединить эти

[1] Определения точек места глаз и брови человека по образу.

направления и применить функцию чувствительности контраста для вычисления порогов различимости цветов, использующихся в специальных цветовых пространствах.

Идея метода состоит в том, чтобы сначала с помощью двумерного дискретного преобразования Фурье и функции чувствительности контраста получить карту порогов различимости цветов, учитывающую неоднородность исходных изображений, затем поточечно сравнить исходные изображения в пространстве Luv, и на основе карты порогов и поточечного сравнения сделать вывод о том, заметит ли человек отличия.

1. МОДЕЛИРОВАНИЕ ЧЕЛОВЕЧЕСКОГО ВОСПРИЯТИЯ

1.1. Цветопередача

Для корректной работы с цветом прежде всего надо учесть зависимость измеряемых результатов от конкретного отображающего устройства. Для этого необходим способ задачи цвета, не зависящий от устройства. Стандартный случай, когда цвет задается как сочетание яркостей красного, зеленого и синего люминофоров, не подходит, потому что спектры излучения люминофоров в мониторе могут не совпадать даже у мониторов из одной партии; к тому же спектры могут меняться со временем. Единственно приемлемым подходом в такой ситуации является использование стандартных цветовых пространств, например таких как CIE XYZ. Это вызывает необходимость периодически производить калибровку монитора, но позволяет получать правильный результат на любом дисплее. В этом случае можно говорить об отображении пикселя с заданными цветовыми координатами.

После представления цвета стандартным образом необходимо научиться определять, сможет ли человек отличить два заданных цвета друг от друга. Для этого необходимо, чтобы в цветовом пространстве были заданы функция и пороговая модель, позволяющие определить расстояние между цветами, а также каково должно быть расстояние, чтобы человек отличил заданные цвета. Таким требованиям удовлетворяют пространства CIE Luv и CIE Lab. Они получаются из пространства XYZ с помощью несложных и, что важно, обратимых нелинейных преобразований. К тому же они являются равномерными по восприятию, то есть такими, что одинаковые изменения каждой из координат приводят к одинаково заметным изменениям цвета. Поэтому расстояние между цветами в них задается формулами

$$\Delta E_{Luv} = \sqrt{\Delta L^2 + \Delta u^2 + \Delta v^2} \qquad \textbf{(1)}$$

$$\Delta E_{Lab} = \sqrt{\Delta L^2 + \Delta a^2 + \Delta b^2}$$

где Δ – разность соответствующих компонент цвета.

Необходимость введения двух пространств связана с различными схемами цветопередачи. В случае излучающих поверхностей (мониторы) используется аддитивное цветовоспроизведение, когда нужный цвет создается с помощью нескольких источников света (триады люминофоров), имеющих разные спектры излучения. При использовании отражающих поверхностей (печать на бумаге) работает субтрактивная схема цветовоспроизведения, когда поверхность освещается белым светом, а нанесенные на нее слоями краски поглощают некоторые части спектра, и поверхность имеет цвет, соответствующий непоглощенной части.

Пространство Luv применяется для задания и сравнения цветов, которые будут рассматриваться на излучающих поверхностях, а Lab – на отражающих поверхностях. Поскольку при постановке задачи было решено ограничиться случаем излучающих поверхностей, то в дальнейшем в работе будет рассматриваться лишь пространство Luv.

Для перехода из пространства RGB в пространство Luv необходимо сначала посчитать координаты цвета в пространстве XYZ, а потом с их помощью получить координаты в пространстве Luv. Для этого надо один раз посчитать так называемую тройственную матрицу[1], а затем умножать на нее вектор (R,G,B).

Методика вычисления тройственной матрицы, а также переход из пространства RGB в пространство Luv описан в приложениях 1 и 2.

Помимо расстояния между цветами в пространстве Luv заданы два порога. В случае если расстояние между двумя цветами меньше 1, то эти цвета невозможно отличить, если расстояние больше 3, то отличие хорошо заметно, если же расстояние больше 1, но меньше 3, то эти цвета можно отличить только при определенных условиях.

К сожалению, указанные значения порогов верны лишь для идеального случая, когда на фоне одного цвета рассматривается квадрат другого цвета.

[1] В английской литературе – tristimulus matrix.

Практика показывает, что чувствительность человеческого глаза к ошибкам цветопередачи в точке зависит от многих факторов, и среди прочих от «однородности» окружающей точку области. В случае черно-белых изображений эту зависимость описывает функция чувствительности контраста.

1.2. Функция чувствительности контраста

В качестве определения контрастности будем использовать определение по Майкельсону (Michaelson), согласно которому контрастность периодического сигнала L (в простейшем случае – яркости изображения) равна

$$C = \frac{L_{\max} - L_{\min}}{L_{\max} + L_{\min}} = \frac{\Delta L_{пиковое}}{L_{среднее}} \approx \frac{\Delta L_{пиковое}}{L_{фон}}.$$

Функция чувствительности контраста (ФЧК) позволяет определить воспринимаемую глазом контрастность изображения в зависимости от частоты рассматриваемого сигнала. Она показывает отношение воспринимаемой контрастности к реально присутствующей. Мы воспользуемся ФЧК, описанной Mannos и Sacrison в статье [2], как одной из наиболее часто используемых в современных работах. Она вычисляется по формуле:

$$CSF(f) = 2.6 \cdot (0.0192 + 0.144 \cdot f) \cdot \exp\left(-(0.144 \cdot f)^{1.1}\right), \qquad \textbf{(2)}$$

где f – пространственная частота изображения.

Пространственная частота периодического сигнала равна числу периодов наблюдаемых под углом в 1˚. Поскольку изображение, как правило, не является периодическим сигналом, то для вычисления пространственной частоты в точке приходится рассматривать лишь некоторую окрестность точки, представлять ее в виде суммы периодических сигналов, и анализировать набор этих сигналов и их амплитуды. График ФЧК приведен на рисунке 1.

Наилучшее восприятие контраста человеческим глазом соответствует частоте около 7,9 периодов/градус и соответствует чувствительности 0.98. Поскольку максимальное значение ФЧК не равно 1, то при использовании ее в качестве весовой функции, ее приходится нормировать. Нормированная ФЧК изображена на рисунке 1 пунктирной линией.

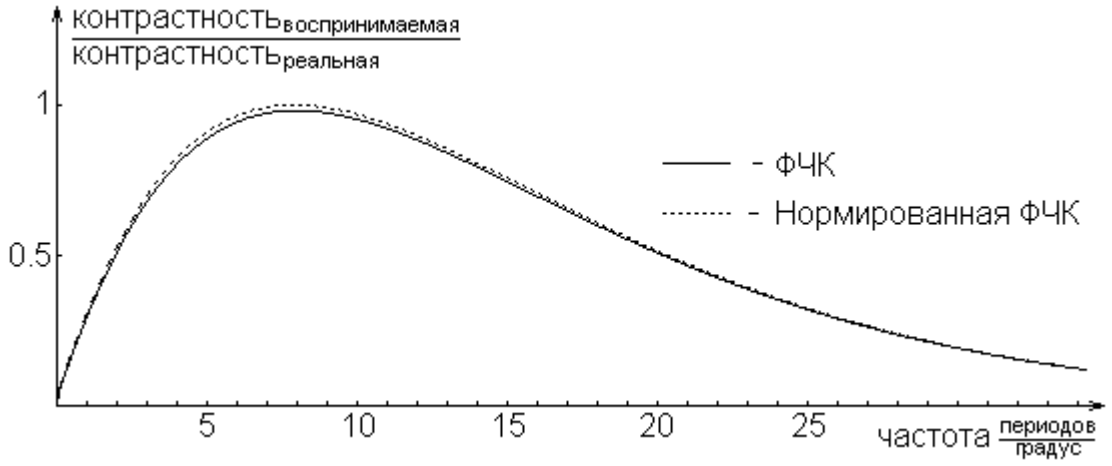


Рисунок 1. Функция чувствительности контраста.

1.3. Вычисление пространственной частоты

Поскольку при определении контрастности использовалась модель периодического сигнала, то для вычисления пространственной частоты было бы логично представить изображение в виде суммы простейших периодических сигналов. Для получения такого разложения надо использовать преобразование, для которого эти функции были бы базисными.

Таким образом, естественно использовать дискретное преобразование Фурье (ДПФ). Поскольку ДПФ не локально – нужно применять его не ко всему изображению, а на каждом шаге рассматривать лишь некоторую окрестность, и, на основании полученных результатов, вычислять пространственную частоту для 4-центральных точек (поскольку окрестность – квадрат со сторонами четной длины).

Исходя из определения функции чувствительности контраста, на вход преобразования Фурье (ПФ) должен подаваться массив яркостей изображения. Но чтобы можно было применить ФЧК для вычисления порогов различимости цветов, нужно чтобы пространственная частота отражала также неоднородности вызванные флуктуациями компонент u и v в пространстве Luv. Поэтому предлагается применить ПФ не к массиву яркостей точек, а к массиву величин

$$E_{Luv} = \sqrt{L^2 + u^2 + v^2} \ .$$

Для того чтобы посчитать пространственную частоту, надо оценить вклад каждой из частот в неоднородность изображения. Вклад каждой частоты определяется ее амплитудой, а также чувствительностью глаза к этой частоте. Таким образом можно записать формулу для пространственной частоты F_{sp} изображения в точке:

$$F_{sp} = \frac{1}{N} \sum_{i,j} \frac{f_{r_{i,j}}}{l} \cdot cpd \cdot A_{ij} \cdot W_{ij} \ , \qquad \textbf{(3)}$$

где

- $f_{r_{ij}}$ – радиальная частота, соответствующая коэффициенту X_{ij} двумерного преобразования Фурье;
- A_{ij} – действительная часть (т.е. амплитуда соответствующего сигнала) коэффициента X_{ij} двумерного ДПФ;
- W_{ij} – весовой коэффициент, учитывающий вклад каждой частоты в воспринимаемую неоднородность изображения;
- l – длина стороны квадрата, в котором рассматривается ДПФ;
- cpd [пиксел/градус] – коэффициент для перевода частот к размерности [период/градус];
- коэффициент N получается из условия $\sum_{i,j} A_{ij} \cdot W_{ij} = 1$

Радиальная частота, соответствующая коэффициенту двумерного преобразования Фурье, вычисляется по формуле:

$$f_r = \sqrt{f_h^2 + f_v^2} ,$$

где f_v и f_h – вертикальная и горизонтальная частоты, соответствующие этому же коэффициенту.

Весовой коэффициент W_{ij} описывает чувствительность глаза к возмущению, создаваемому определенной частотой ДПФ, и имеет вид:

$$W_{ij} = NormCSF\left(\frac{f_{r_{i,j}}}{l} \cdot cpd\right)\cdot, \qquad \textbf{(4)}$$

где NormCSF – нормированная функция чувствительности контраста.

1.4. Вычисление порогов различимости цветов

После вычисления пространственной частоты в точке, можно посчитать порог различимости цветов в этой точке.

Поскольку ФЧК показывает отношение воспринимаемой контрастности к реальной, то для того, чтобы определить, отличит ли человек два цвета, нужно найти расстояние между этими цветами, умножить его на значение ФЧК и полученное число сравнить с порогами 1 и 3. Отсюда следует, что в точке (i,j) в качестве первого порога нужно использовать величину $T_{ij,}$ а в качестве второго порога – 3 T_{ij}

$$T_{ij} = \frac{1}{CSF\left(F_{sp\ ij}\right)}, \qquad \textbf{(5)}$$

где $F_{sp\ ij}$ – значение пространственной частоты;

Однако из-за отличий между изображениями, значения пространственной частоты в точке для каждого из изображений могут отличаться, и поэтому могут отличаться пороги различимости цветов. Чтобы избежать этого случая, при выборе порогового значения для данной точки,

выбирается меньший из порогов, который соответствует большей чувствительности глаза

После получения порогов и поточечного сравнения цветов, можно сформировать карту видимых ошибок, в которой каждой точке изображения соответствует расстояние между цветами, если оно больше порога T_{ij}, и 0 в противном случае.

1.5. Статистическая обработка и визуализация карты видимых ошибок.

После получения карты видимых ошибок нужно собрать представленные в ней данные, вычислить по ним некоторую обобщенную характеристику и с ее помощью определить, насколько отличаются предложенные изображения.

В качестве такой характеристики предлагается использовать среднее значение карты видимых ошибок:

$$LuvDiff = \frac{1}{H \cdot W} \sum_{1 \le i \le W} \sum_{1 \le j \le H} \Delta E_{Luv\,i,j} \cdot I_{i,j}$$, где

H, W – высота и ширина изображения соответственно;

$\Delta E_{Luv\,i,j}$ – расстояние между цветами в точке (i,j); см. формулу 1

$I_{i,j}$ вычисляется по формуле:

$$I_{i,j} = \begin{cases} 1 \ : \ \Delta E_{Luv\,i,j} > T_{i,j} \\ 0 \ : \ \Delta E_{Luv\,i,j} \le T_{i,j} \end{cases}$$

T_{ij} – значение порога в точке (i,j), вычисляемое по формуле 5.

Эксперименты показали, что разница между изображениями становится хорошо заметна, когда характеристика LuvDiff превышает значение 0.8-0.9 для черно-белых изображений, и 1.2-1.4 для цветных.

Для того, чтобы определить области, в которых отличия между изображениями будут заметны, карту видимых ошибок надо визуализировать. Для этого предлагается следующая операция.

Исходя из свойств пространства Luv, карта разбивается на 3 области, каждая из которых закрашивается своим цветом. Первая область, где расстояние между цветами не позволяет отличить их, закрашивается черным цветом, вторая, где при определенных условиях можно заметить разницу – оттенками зеленого, и третья, где разница хорошо заметна – оттенками красного.

Помимо характеристики LuvDiff, степень отличия изображений неплохо характеризуют такие величины, как процент точек, закрашенных черным, красным или зеленым цветом.

2. ОСОБЕННОСТИ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

2.1. Общие сведения

На основе предложенного алгоритма были написаны программа сравнения изображений, и программа тестирования алгоритмов сжатия изображений с потерями качества. Программы компилировался в среде Borland Delphi 7 Professional Edition. В программах для вычисления дискретного преобразования Фурье, а также весовых коэффициентов в **формулах 3, 4** использовались функции библиотеки Intel Signal Processing Library версии 4.1.

Кроме того, в целях оптимизации программы сравнения изображений алгоритм был модифицирован. За счет этого время работы программы было сокращено на 20%, а объем используемой памяти уменьшен на 10%. После этого, при тестировании на платформе Intel Pentium-IV 2,4 256M Ram под управлением Windows XP, изображения размером 200x300 пикселов сравнивались за 5-6 секунд, при этом использовалось около 256 Мб памяти.

2.2. Схема алгоритма

На основе решений, описанных в параграфе 1, был предложен алгоритм. Его схема изображена на **рисунке 2**. На вход алгоритма поступают 2 изображения. Для каждого изображения, используя преобразование Фурье и функцию чувствительности контраста, строится карта порогов различимости цветов (Индивидуальная карта порогов). Затем эти карты объединяются в совместную карту порогов, при этом из двух порогов выбирается тот порог, который обеспечивает большую чувствительность. Одновременно с этим оба изображения переводятся в пространство Luv, и поточечно сравниваются по **формуле 1**. Далее карта ошибок сравнивается с совместной картой порогов, и значения, превосходящие порог различимости цветов T_{ij}, попадают в карту видимых ошибок. После этого производится визуализация и статистическая обработка карты видимых ошибок.

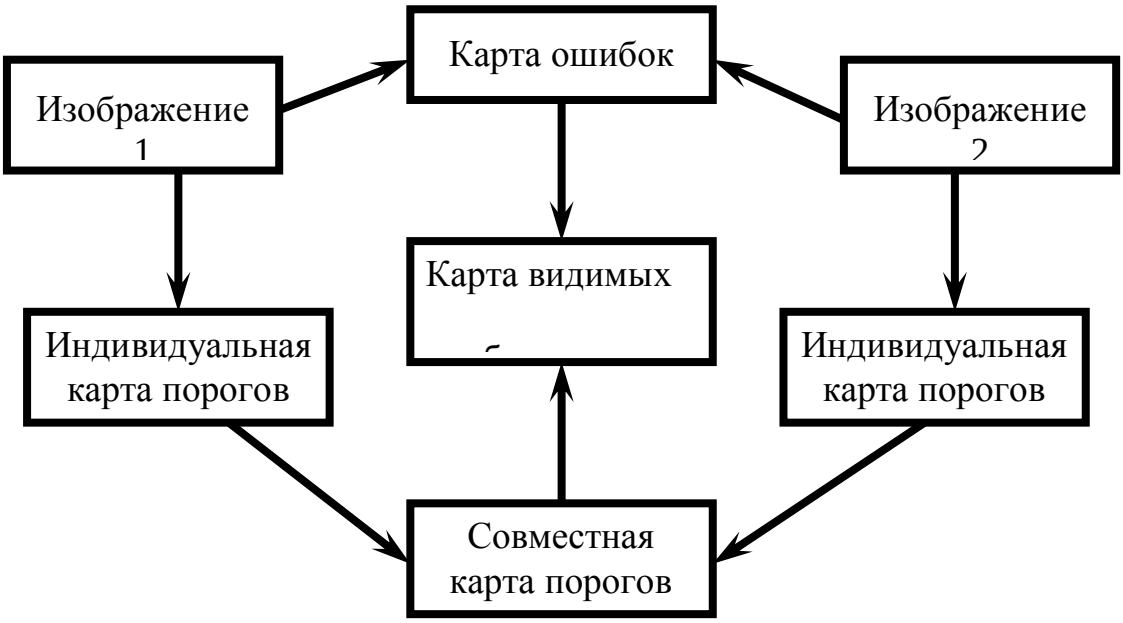


Рисунок 2. Схема алгоритма сравнения изображений

2.3. Преобразование Фурье

Одним из свойств дискретного преобразования Фурье является отсутствие для него эффективной вычислительной схемы, а многократное вычисление преобразования Фурье, необходимое для вычисления пространственной частоты, вызывает очень большие накладные расходы. Поэтому решено было разбить исходное изображение на блоки $2^n x 2^n$, чтобы к каждому блоку можно было применить быстрое преобразование Фурье (БПФ). Для БПФ существуют эффективные, так называемые пирамидальные схемы вычисления. Размер блока выбирается так, чтобы минимальное возможное значение функции чувствительности контраста было не больше ½, а максимальное – равно 1.

Другой проблемой, возникающей при использовании разновидностей преобразования Фурье, является проблема краевых эффектов, когда для подсчета пространственной частоты на краях изображения нужны значения за границей изображения. В качестве решения предлагается использовать четное продолжение сигнала.[13,14] Нечетное или нулевое продолжения (или

продолжение константой) не подходят, потому что предполагается, что сравниваемые картинки были «вырезаны» из некоторого большого изображения.

2.4. Вычисление пространственной частоты

Анализируя **формулу 4** можно заметить, что пространственные частоты $\frac{f_{r_{i,j}}}{l} \cdot cpd$, а следовательно и коэффициенты W_{ij}, зависят только от размеров блока, к которому применяется БПФ. Исходя из этого, можно оптимизировать алгоритм, создав два специальных массива, вычислить их значения перед сравнением Единственная сложность возникает с первым элементом, который соответствует нулевой частоте в преобразовании Фурье блока, которая, в свою очередь, может не совпадать с нулевой частотой преобразования всего изображения. Поэтому первый элемент массива частот инициализируется значением $Freqs[0][0] = 0.5 \cdot Freqs[0][1]$, а массив коэффициентов – $Matrix[0][0] = NormCSF(Freqs[0][0])$

Переход из цветового пространства CIE XYZ в пространство CIE Luv

Для перехода в пространство Luv достаточно знать не сами координаты x, y, z, а промежуточные значения X, Y и Z. В любом случае их можно восстановить по формулам:

$$X = x \cdot \frac{Y}{y}$$
$$Y = Y$$
$$Z = z \cdot \frac{Y}{y}$$

Далее переход в пространство Luv осуществляется по формулам:

$$u' = \frac{4X}{X + 15Y + 3Z}$$
$$v' = \frac{9Y}{X + 15Y + 3Z}$$

$$L^* = \begin{cases} 116\left(\frac{Y}{Y_n}\right)^{1/3} - 16 & : \left(\frac{Y}{Y_n}\right) > 0.008856 \\ 903.3\left(\frac{Y}{Y_n}\right) & : \left(\frac{Y}{Y_n}\right) \le 0.008856 \end{cases}$$

$$u^* = 13L^*(u' - u'_n)$$
$$v^* = 13L^*(v' - v'_n)\ ,$$

где u'_n, v'_n, Y_n – соответствующие значения белого цвета для данного монитора. В случае если для воспроизведения цвета используется модель гамма-коррекции, то вместо отношения $\frac{Y}{Y_n}$ следует брать $\left(\frac{Y}{Y_n}\right)^{\gamma}$

Подробнее о цветовых пространствах, их свойствах и переходах в различные пространства смотри книгу Девида Тревиса [11].

Переход из цветового пространства RGB в пространство CIE XYZ

Переход в пространство XYZ осуществляется умножением вектора RGB на тройственную матрицу T и последующим нормированием полученных значений.[12,14,15]

Для вычисления тройственной матрицы надо:

1. Получить цветовые координаты каждого из люминофоров монитора (либо в документации к профессиональным мониторам, либо непосредственно измерить с помощью колориметра, либо взять значения соответствующие стандарту NTSC) и записать их в виде матрицы:

$$C = \begin{pmatrix} x_r & x_g & x_b \\ y_r & y_g & y_b \\ z_r & z_g & x_b \end{pmatrix}$$

2. Посчитать координаты белой точки $(x_w \quad y_w \quad z_w)$ монитора, увеличив яркость каждого люминофора до максимума

3. Посчитать тройственные значения[1] белого цвета X_w, Y_w, Z_w по формулам:

$$X_w = \frac{x_w}{y_w}$$
$$Y_w = 1$$
$$Z_w = \frac{z_w}{y_w}$$

и с их помощью вычислить вектор V:

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} = C^{-1} \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix},$$

где C^{-1} – матрица, обратная к C.

4. Посчитать тройственную матрицу T:

$$T = C \cdot \begin{pmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{pmatrix}$$

Теперь переход в пространство XYZ осуществляется по формулам:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = T \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

$$x = \frac{X}{X+Y+Z}$$
$$y = \frac{Y}{X+Y+Z},$$
$$z = \frac{Z}{X+Y+Z}$$

где x, y, z – хроматические координаты в пространстве XYZ, Y – яркость цвета, а R, G, B – соответствующие координаты в пространстве RGB, $0 \le R, G, B \le 1$

[1] В английской литературе – tristimulus values.

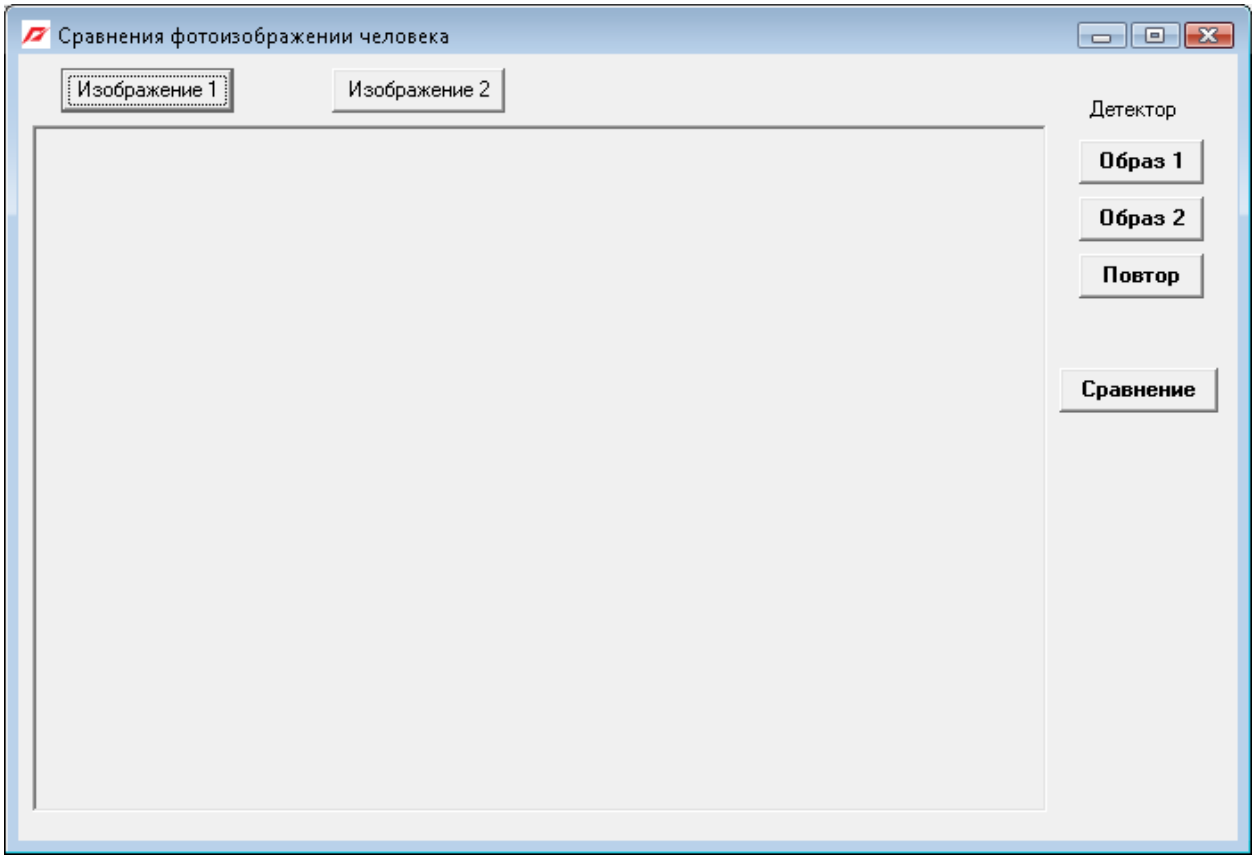
Для монитора, соответствующего стандарту NTSC матрицы C и T имеют вид:

$$C = \begin{pmatrix} 0.67 & 0.21 & 0.14 \\ 0.33 & 0.71 & 0.08 \\ 0 & 0.08 & 0.78 \end{pmatrix} \qquad T = \begin{pmatrix} 0.3639 & 0.3412 & 0.1953 \\ 0.2198 & 0.6846 & 0.0956 \\ 0.0220 & 0.01191 & 1.0748 \end{pmatrix}$$

3. Программного обеспечения сравнения двух изображений

3.1. Описание программы сравнивающих фотопортретов человека

Программа FR_New позволяет сравнивать двух изображений лиц человеков. Для сравнении нужно выбрать два изображения человека по размеру до 400x500 пикселя (в формате *.bmp). С целью сравнения изображений задача направлена сравнения изображении лиц. Окна программ представлены рис. 1.



При помощью кнопки Изображение 1 и Изображение 2 открывается два изображение в формате *.bmp.

Алгоритм сравнения работает следующем:

- Найти точек экстремум гистограмм цветов изображений;
- По этой точки определения лица человека;
- Определения геометрические значение;
- Сравнение по параметрических значений;
- Вычисления процентов подобию изображению.

Изображение 1 - кнопка для получение первую фотопортрета человека,

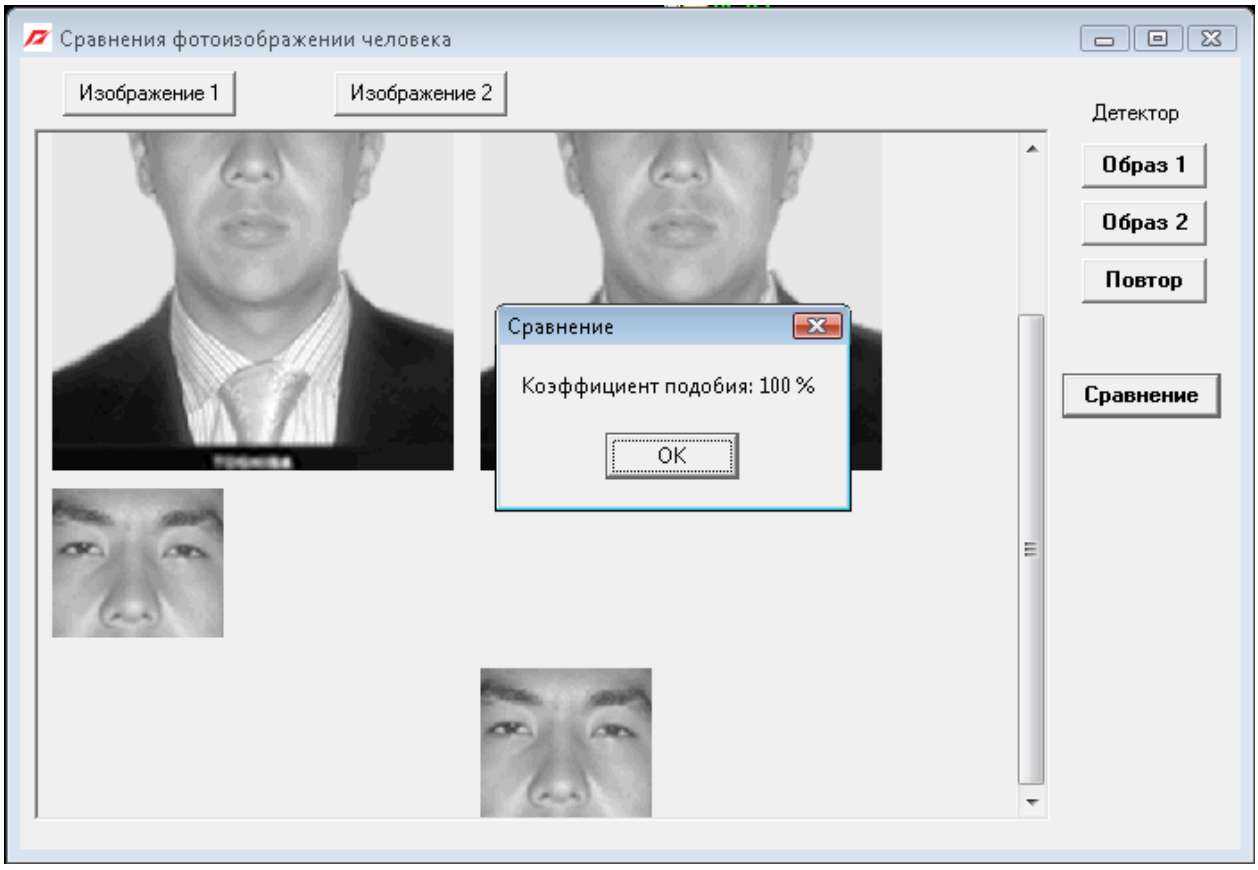
Изображение 2 -кнопка для получение вторую фотопортрета человека,

Образ 1 - кнопка детектор 1-го информативного образа,

Образ 2 - кнопка детектор 2-го информативного образа,

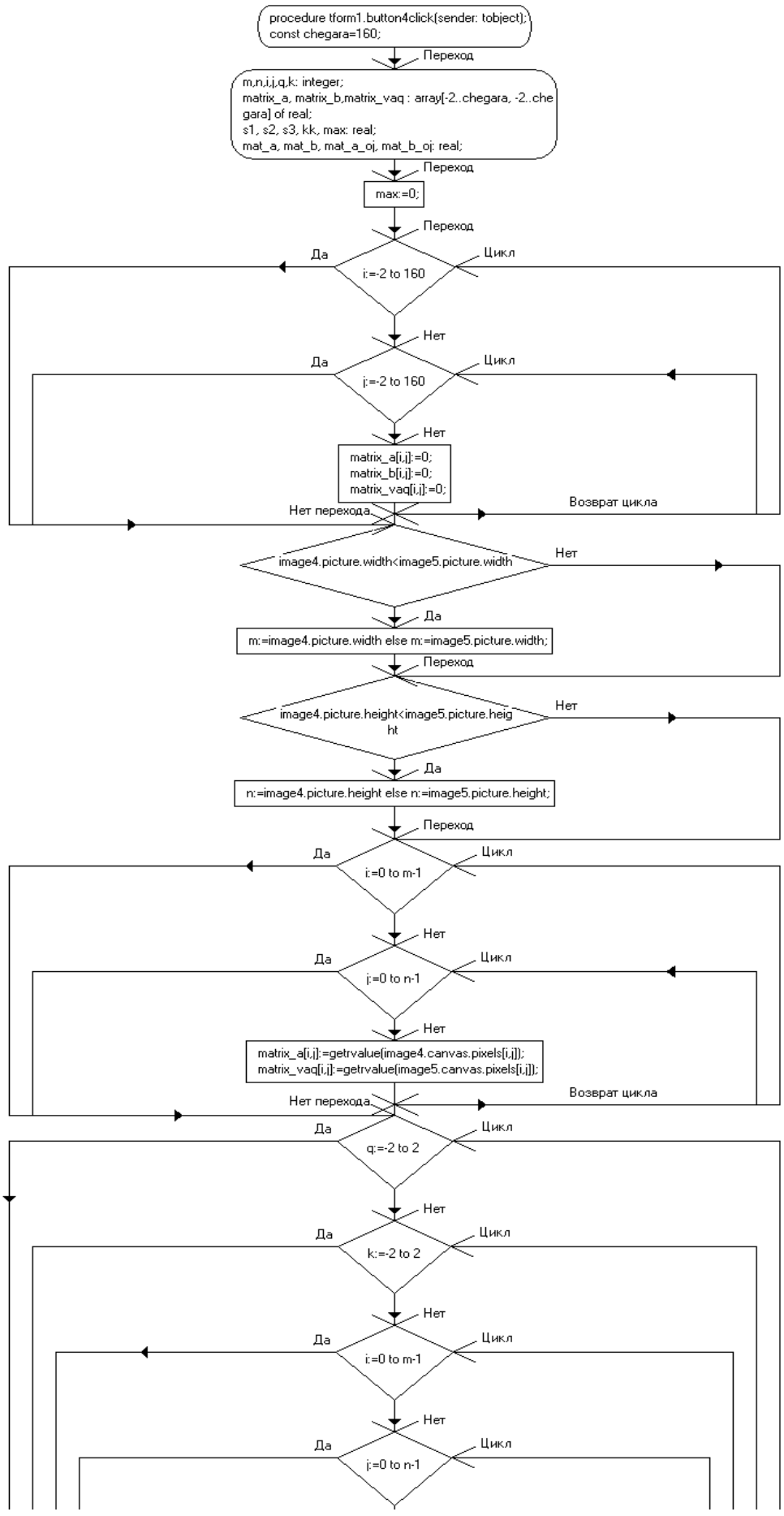
Повтор - повторение детектора,

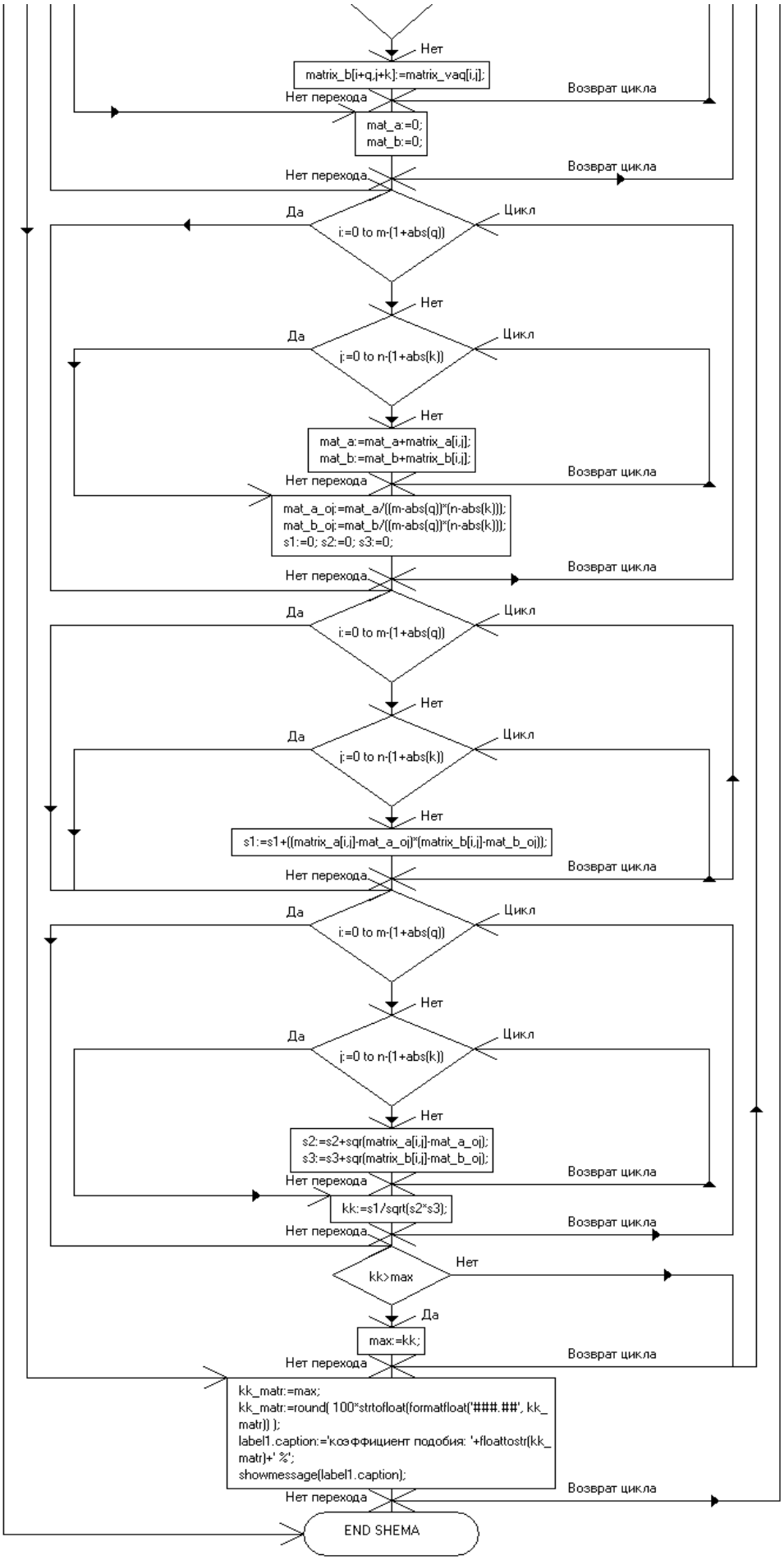
Сравнение - кнопка сравнение образов.



Результат сравнении

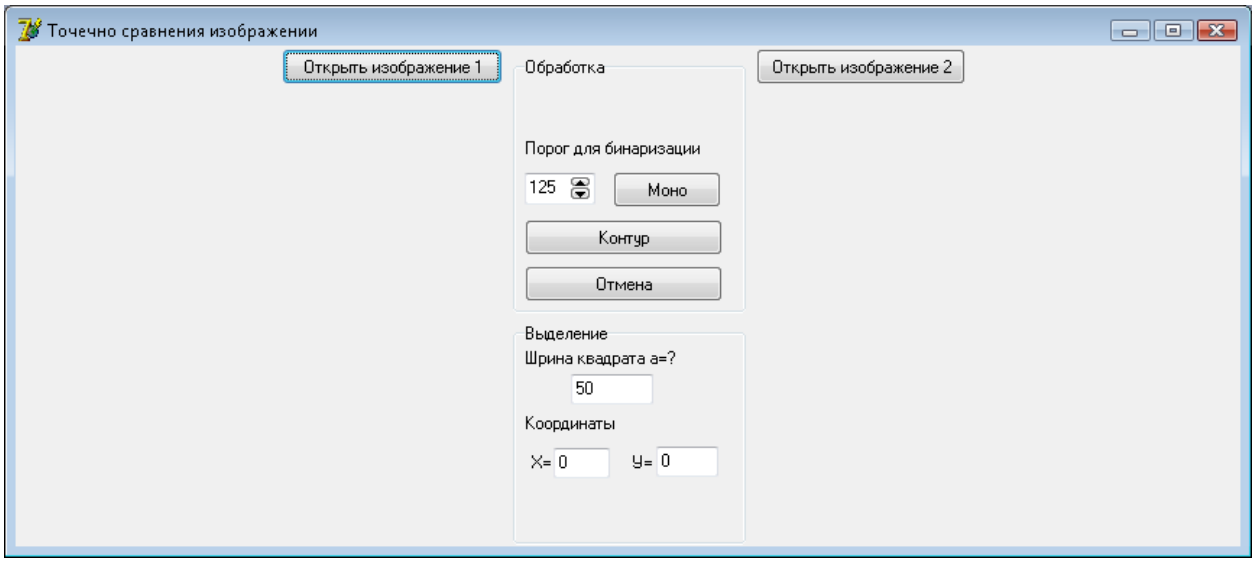
Блок-схема сравнении




Нет
matrix_b[i+q,j+k]:=matrix_vaq[i,j];
Возврат цикла
Нет перехода
mat_a:=0;
mat_b:=0;
Возврат цикла
Нет перехода
Да
i:=0 to m-(1+abs(q))
Цикл
Нет
Да
j:=0 to n-(1+abs(k))
Цикл
Нет
mat_a:=mat_a+matrix_a[i,j];
mat_b:=mat_b+matrix_b[i,j];
Возврат цикла
Нет перехода
mat_a_oj:=mat_a/((m-abs(q))*(n-abs(k)));
mat_b_oj:=mat_b/((m-abs(q))*(n-abs(k)));
s1:=0; s2:=0; s3:=0;
Возврат цикла
Нет перехода
Да
i:=0 to m-(1+abs(q))
Цикл
Нет
Да
j:=0 to n-(1+abs(k))
Цикл
Нет
s1:=s1+((matrix_a[i,j]-mat_a_oj)*(matrix_b[i,j]-mat_b_oj));
Возврат цикла
Нет перехода
Да
i:=0 to m-(1+abs(q))
Цикл
Нет
Да
j:=0 to n-(1+abs(k))
Цикл
Нет
s2:=s2+sqr(matrix_a[i,j]-mat_a_oj);
s3:=s3+sqr(matrix_b[i,j]-mat_b_oj);
Возврат цикла
Нет перехода
kk:=s1/sqrt(s2*s3);
Возврат цикла
Нет перехода
kk>max
Нет
Да
max:=kk;
Возврат цикла
Нет перехода
kk_matr:=max;
kk_matr:=round(100*strtofloat(formatfloat('###.##', kk_
matr)));
label1.caption:='коэффициент подобия: '+floattostr(kk_
matr)+' %';
showmessage(label1.caption);
Возврат цикла
Нет перехода
END SHEMA

3.2. Описание программ точечно сравнивание

Программа PointToPoint позволяет точечно сравнивать два изображения. В этой программе пользователь можно получить полутоновое изображение и бинарное изображение. В каждом этапе можно сравнивать часть изображений. Результат сравнения отображается как коэффициент подобия (в проценте).



Описания функциональных кнопок:

Открыть изображение 1 – можно открыть 1-го изображения;

Открыть изображение 2 – можно открыть 2-го изображения;

Полутоновий – преобразования цветных изображений на полутоновое изображение;

Моно - по порогам преобразования цветных изображений на бинарное изображение;

Контур – выделение границ изображении методами Роберта и Собеля;

Отмена – восстанавливает изображении;

Сравнивать – точечно сранения фиксированные участки на изображении

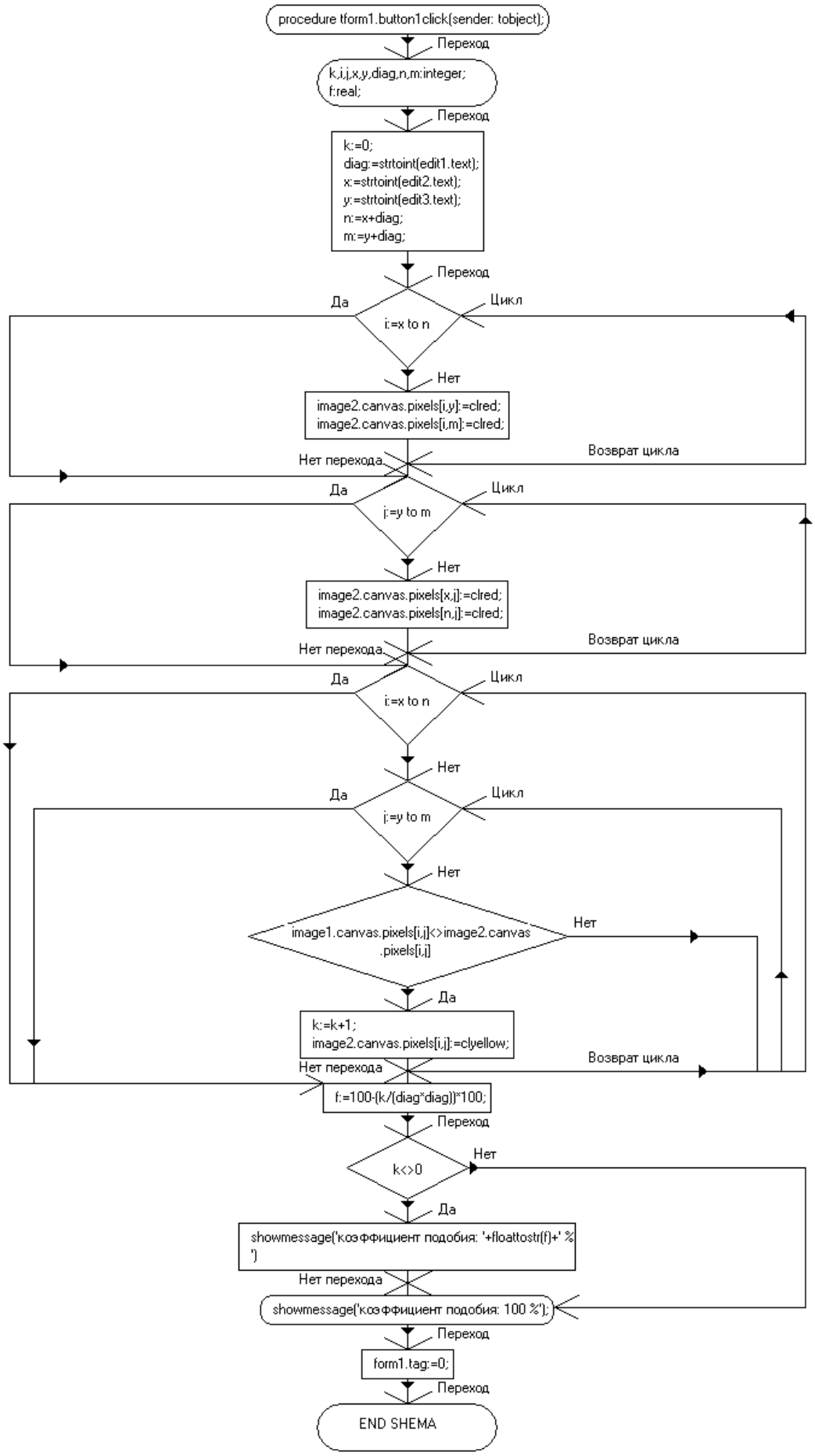
Алгоритм точечно сравнения написано в Delphi 7:

```
var
k,i,j,x,y,diag,n,m:Integer;
f:real;
begin
  k:=0;
  diag:=StrToInt(Edit1.Text);
  x:=StrToInt(Edit2.Text);
  y:=StrToInt(Edit3.Text);
  n:=x+diag;
  m:=y+diag;
For i:=x to n do
 begin
   Image2.Canvas.Pixels[i,y]:=clRed;
   Image2.Canvas.Pixels[i,m]:=clRed;
 end;
For j:=y to m do
 begin
   Image2.Canvas.Pixels[x,j]:=clRed;
   Image2.Canvas.Pixels[n,j]:=clRed;
 end;

for i:=x to n do
for j:=y to m do
 if  Image1.Canvas.Pixels[i,j]<>Image2.Canvas.Pixels[i,j] then
   Begin
     k:=k+1;
     Image2.Canvas.Pixels[i,j]:=ClYellow;
   end;
 f:=100-(k/(diag*diag))*100;
```

```
 if k<>0 then ShowMessage('Коэффициент подобия: '+FloatToStr(f)+' %')
   else ShowMessage('Коэффициент подобия: 100 %');
   Form1.Tag:=0;
end;
```

Блок-схема процедуры иллюстрируется ниже.


procedure tform1.button1click(sender: tobject);
Переход
k,i,j,x,y,diag,n,m:integer;
f:real;
Переход
k:=0;
diag:=strtoint(edit1.text);
x:=strtoint(edit2.text);
y:=strtoint(edit3.text);
n:=x+diag;
m:=y+diag;
Переход
Да
i:=x to n
Цикл
Нет
image2.canvas.pixels[i,y]:=clred;
image2.canvas.pixels[i,m]:=clred;
Нет перехода
Возврат цикла
Да
j:=y to m
Цикл
Нет
image2.canvas.pixels[x,j]:=clred;
image2.canvas.pixels[n,j]:=clred;
Нет перехода
Возврат цикла
Да
i:=x to n
Цикл
Нет
Да
j:=y to m
Цикл
Нет
image1.canvas.pixels[i,j]<>image2.canvas
.pixels[i,j]
Нет
Да
k:=k+1;
image2.canvas.pixels[i,j]:=clyellow;
Нет перехода
Возврат цикла
f:=100-(k/(diag*diag))*100;
Переход
k<>0
Нет
Да
showmessage('коэффициент подобия: '+floattostr(f)+' %
')
Нет перехода
showmessage('коэффициент подобия: 100 %');
Переход
form1.tag:=0;
Переход
END SHEMA

4. БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

4.1. Основы проектирования техносферы по условиям безопасности жизнедеятельности

Жизнедеятельность – это повседневная деятельность и способ существования человека. Приступая к изучению безопасности жизнедеятельности человека в техносфере, следует определить его место в общем объеме знаний о взаимодействии живых существ и окружающей среды.

Основы проектирования техносферы по условиям БЖД включают в себя:

1. Обеспечение комфорта в зонах жизнедеятельности;
2. Правильное расположение источников опасностей и зон пребывания человека;
3. Сокращение размеров опасных зон;
4. Применение экобиозащитной техники;
5. Применение средств индивидуальной защиты.

1. Комфорт - оптимальное сочетание параметров микроклимата, удобств, благоустроенности и уюта в зонах деятельности и отдыха человека.

Комфортность техносферы. Наилучшие показатели работоспособности и отдыха достигаются при комфортном состоянии среды обитания и при рациональных режимах труда и отдыха.

Безопасность жизнедеятельности обеспечивается: комфортом в зонах жизнедеятельности; правильным расположением источников опасности и зон пребывания человека; сокращением размеров опасных зон; средств индивидуальной защиты.

Наилучшие показатели работоспособности и отдыха достигаются при: комфортном состоянии среды обитания; рациональных режимах труда и отдыха.

Комфортные и допустимые параметры воздушной среды в рабочих зонах регламентируются государственными стандартами и обеспечиваются в основном применением систем кондиционирования, вентиляции и отопления. Нормативные (оптимальные, допустимые) значения параметров микроклимата в рабочих зонах производственных помещениях зависят от категории выполняемых работ, периода года и некоторых других показателей.

Важную роль в достижении эффективной деятельности играет искусственное освещение.

Эффективность деятельности человека в значительной степени зависит от организации рабочего места, его оборудованию в соответствии с требованиями эргономики.

Важное значение при достижении максимально эффективной деятельности играет режим труда и отдыха. Сохранение высокой работоспособности достигается правильным чередованием режимов труда и отдыха.

2. Опасные зоны и зоны пребывания человека.

Вредные и травмирующие воздействия, генерируемые техническими системами, образуют в жизненном пространстве техносферы опасные зоны.

Одновременно с опасными зонами в жизненном пространстве существуют зоны деятельности (пребывания) человека. (Рис 4.1.1.) В быту — зона жилища, городская среда. В условиях производства — рабочая зона, рабочее место.

Рабочая зона — пространство высотой 2 м над уровнем пола или площадки, на которой расположено рабочее место.

Рабочее место — зона постоянной или временной (более 50% или более 2 ч непрерывно) деятельности рабочего.

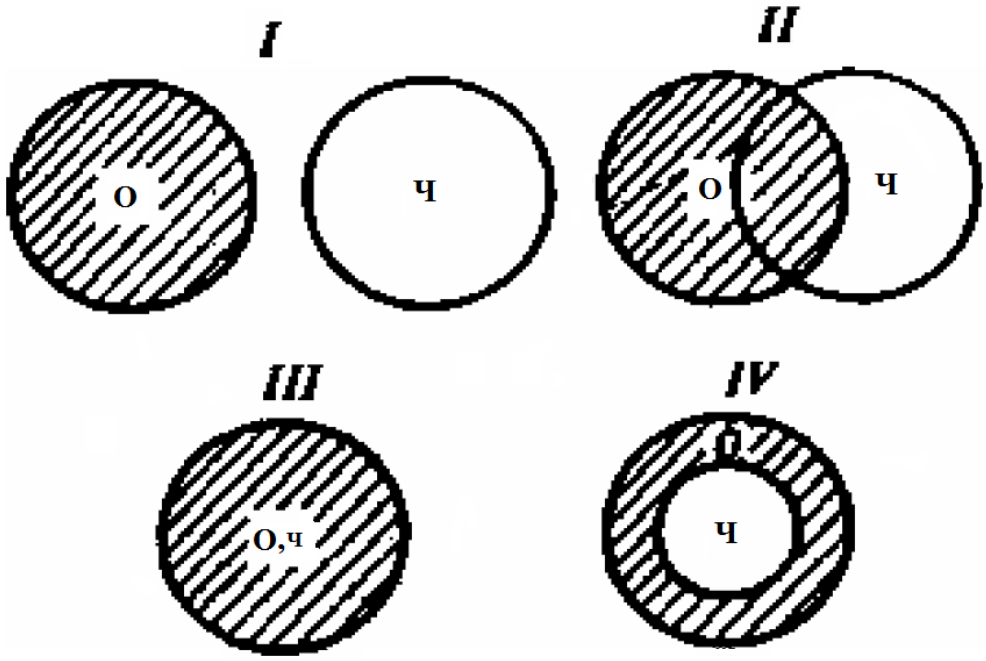


Рис 4.1.1. Варианты взаимного положения зоны опасности (О) и зоны пребывания человека (Ч):

I—безопасная ситуация; ***II***—ситуация кратковременной или локальной опасности;

III— опасная ситуация; ***IV***—условная безопасная ситуация

Защита расстоянием.

Полную безопасность гарантирует только 1 вариант взаимного расположения зон пребывания и действия негативных факторов — защита расстоянием, реализуемый при дистанционном управлении, наблюдении и т.п.

Во 2 варианте негативное воздействие существует лишь в совмещенной части областей: если человек в этой части находится кратковременно (осмотр, мелкий ремонт и т.п.), то и негативное воздействие возможно только в этот период времени,

В 3 варианте — негативное воздействие может быть реализовано в любой момент,

В 4 варианте — только при нарушении функциональной целостности средств защиты зоны пребывания человека (как правило, средств индивидуальной защиты — (СИЗ), кабин наблюдения и т.п.).

3. Сокращение размеров опасных зон.

При воздействии вредных факторов сокращение размеров зон должно достигаться, прежде всего, совершенствованием технических систем, приводящим к уменьшению выделяемых ими отходов.

Для ограничения вредного воздействия на человека и среду обитания к технической системе предъявляются требования по величине выделяемых в среду токсичных веществ в виде предельно допустимых выбросов или сбросов (ПДВ или ПДС), а также по величине энергетических загрязнений в виде предельно допустимых излучений в среду обитания.

4. Средства индивидуальной защиты.

На ряде предприятий существуют такие виды работ или условия труда, при которых человек может получить травму или иное воздействие, опасное для здоровья. Номенклатура СИЗ, включает обширный перечень средств, применяемых в производственных условиях (СИЗ повседневного использования), а также средств, используемых в чрезвычайных ситуациях (СИЗ кратковременного использования)

4.2. Влияние гиподинамии на организм человека

Гиподинамия (от греческогоhypo - внизу и dynamis - сила) - ослабление мышечной работы, обусловленное сидячим образом жизни и ограничением двигательной активности. Медицинские работники называют симптомокомплекс гиподинамии болезнью столетия и оборотной стороной прогресса. Справедливость подобного утверждения, к несчастью, очевидна. Прогресс дарит человеку большое число наиболее совершенных приспособлений, способных освободить нас от любой физической нагрузки. В нашем распоряжении машины, поезда и самолеты, чтоб без малейших усилий перемещаться на большие расстояния.

Научно-технический прогресс, наряду с улучшением условий жизни и работы в современном обществе, создает предпосылки для малоподвижного образа жизни. Ограничение функции движения вызывает особое состояние - гипокинезический синдром или болезнь. Гиподинамия (или гипокинезия) как ржавчина разъедает профессиональную работоспособность, ухудшает здоровье, сокращает продолжительность жизни.

Самым серьезным отрицательным для здоровья фактором умственной деятельности является гиподинамия. Под термином **«гиподинамия»** подразумевают ограниченная по интенсивности и времени мышечная деятельность в режиме дня здорового человека, которая в свою очередь значительно влияет на предельные физиологические возможности организма. Двигательная активность принадлежит к числу основных факторов, определяющих уровень обменных процессов организма и состояние его костной, мышечной и сердечно-сосудистой системы. Она связана тесно с тремя аспектами здоровья: физическим, психическим и социальным.

Малоподвижный образ жизни студентов приводит к тому, что нарушается функциональное состояние всех систем организма – сердечно-сосудистой, дыхательной, опорно-двигательной, нервной, пищеварительной, так как деятельность всех его систем направлена на хорошее обеспечение работоспособности мышц. При отсутствии достаточной дозы ежедневных мышечных движений происходят нежелательные и существенные изменения функционального состояния мозга и сенсорных систем. Наряду с изменением в деятельности высших отделов головного мозга снижается уровень функционирования и подкорковых образований, отвечающих за работу органов чувств (слух, равновесие, вкус и другие) или ведающих жизненно важными функциями (дыханием, кровообращением, пищеварением). Вследствие этого наблюдается снижение общих защитных сил организма, увеличение риска возникновения различных заболеваний. Так же характерна повышенная утомляемость, крайняя неустойчивость настроения, ослабление самообладания, нетерпеливость, утрата способности к длительному

умственному и физическому напряжению. Кроме того, при уменьшении физической нагрузки в мышцах отмечается усиливающаяся атрофия со структурными и функциональными изменениями, ведущими к прогрессирующей мышечной слабости. Например, из-за ослабления мышц связочного и костного аппарата туловища, нижних конечностей, которые не могут выполнять полноценно свою функцию - удержание опорно-двигательного аппарата, развиваются нарушения осанки, деформация позвоночника, грудной клетки, таза и т.д., которые влекут целый ряд нарушений здоровья, что приводит к снижению работоспособности.

Скелетно-мышечная система, как никакая другая, особо подвержена длительным влияниям со стороны окружающей среды. Длительность воздействия поражающих факторов и незаметные внешние проявления первых симптомов делают совокупность болезней этой системы особо важной для рассмотрения. Сюда относятся артриты, остеохондроз, сколиоз.

Профессиональная гиподинамия значительно усугубляет влияние на организм человека других факторов трудового процесса, например, нервно-эмоционального напряжения и монотонности. Отрицательные эмоции приобретают выраженную интенсивность и характер стресса. На фоне снижения эмоциональной устойчивости к стрессогенным факторам отрицательные эмоции оказываются трудноперенос имыми для человека. Гиподинамия в сочетании с высоким уровнем нервно-эмоционального напряжения может стать причиной срыва адаптационных реакций человека.

Снижение физических нагрузок в условиях современной жизни, с одной стороны, и недостаточное развитие массовых форм физической культуры среди населения, с другой стороны, приводят к ухудшению различных функций и появлению негативных состояний организма человека.

Для обеспечения нормальной жизнедеятельности организма человека необходима достаточная активность скелетных мышц. Работа мышечного аппарата способствует развитию мозга и установлению межцентральных и межсенсорных взаимосвязей. Двигательная деятельность повышает

энергопродукцию и образование тепла, улучшает функционирование дыхательной, сердечно-сосудистой и других систем организма.

Научные данные свидетельствуют о том, что у большинства людей при соблюдении ими гигиенических правил и ведении здорового образа жизни есть возможность жить до 100 лет и более.

К сожалению, многие люди не соблюдают самых простейших, обоснованных наукой норм здорового образа жизни. Последние годы в силу высокой нагрузки на работе и дома и других причин у большинства отмечается дефицит в режиме дня, недостаточная двигательная активность, обусловливающая появление гипокинезии, которая может вызвать ряд серьёзных изменений в организме людей.

Гипокинезия – это пониженная двигательная активность. Она может быть связана с физиологической незрелостью организма, с особыми условиями работы в ограниченном пространстве, с некоторыми заболеваниями и др. причинами. В некоторых случаях (гипсовая повязка, постельный режим) может быть полное отсутствие движений или акинезия, которая переносится организмом еще тяжелее.

Существует и близкое понятие – гиподинамия. Это понижение мышечных усилий, когда движения осуществляются, но при крайне малых нагрузках на мышечный аппарат. В обоих случаях скелетные мышцы нагружены совершенно недостаточно. Возникает огромный дефицит биологической потребности в движениях, что резко снижает функциональное состояние и работоспособность организма.

Наиболее устойчивы к развитию гиподинамических признаков мышцы антигравитационного характера (шеи, спины). Мышцы живота атрофируются сравнительно быстро, что неблагоприятно сказывается на функции органов кровообращения, дыхания, пищеварения. Это атрофические изменения в мышцах, общая физическая детренированность, детренированностьсердечно-сосудистой системы, понижение ортостатической устойчивости, изменение водно-солевого баланса, системы крови, деминерализация костей и т.д. В

конечном счете снижается функциональная активность органов и систем, нарушается деятельность регуляторных механизмов, обеспечивающих их взаимосвязь, ухудшается устойчивость к различным неблагоприятным факторам; уменьшается интенсивность и объем афферентной информации, связанной с мышечными сокращениями, нарушается координация движений, снижается тонус мышц (тургор), падает выносливость и силовые показатели.

В условиях гиподинамии снижается сила сердечных сокращений в связи с уменьшением венозного возврата в предсердия, сокращаются минутный объем, масса сердца и его энергетический потенциал, ослабляется сердечная мышца, снижается количество циркулирующей крови в связи с застаиванием ее в депо и капиллярах. Тонус артериальных и венозных сосудов ослабляется, падает кровяное давление, ухудшаются снабжение тканей кислородом (гипоксия) и интенсивность обменных процессов (нарушения в балансе белков, жиров, углеводов, воды и солей).

Уменьшается жизненная емкость легких и легочная вентиляция, интенсивность газообмена. Все это ослаблением взаимосвязи двигательных и вегетативных функций, неадекватностью нервно-мышечных напряжений. Таким образом, при гиподинамии в организме создается ситуация, чреватая "аварийными" последствиями для его жизнедеятельности. Если добавить, что отсутствие необходимых систематических занятий физическими упражнениями связано с негативными изменениями в деятельности высших отделов головного мозга, его подкорковых структурах и образованиях, то становится понятно, почему снижаются общие защитные силы организма и возникает повышенная утомляемость, нарушается сон, снижается способность поддерживать высокую умственную или физическую работоспособность.

Последствия гиподинамии. Еще в древности было замечено, чтофизическая активность способствует формированию сильного и выносливого человека, а неподвижность ведет к снижению работоспособности, заболеваниям и тучности. Все это происходит

вследствие нарушения обмена веществ. Уменьшение энергетического обмена, связанное с изменением интенсивности распада и окисления органических веществ, приводит к нарушению биосинтеза, а также к изменению кальциевого обмена в организме. Вследствие этого в костях происходят глубокие изменения. Прежде всего, они начинают терять кальций. Это приводит к тому, что кость делается рыхлой, менее прочной. Кальций попадает в кровь, оседает на стенках кровеносных сосудов, они склерозируются, т. е. пропитываются кальцием, теряют эластичность и делаются ломкими. Способность крови к свертыванию резко возрастает. Возникает угроза образования кровяных сгустков (тромбов) в сосудах. Содержание большого количества кальция в крови способствует образованию камней в почках.

Отсутствие мышечной нагрузки снижает интенсивность энергетического обмена, что отрицательно сказывается на скелетных и сердечной мышцах. Кроме того, малое количество нервных импульсов, идущих от работающих мышц, снижает тонус нервной системы, утрачиваются приобретенные ранее навыки, не образуются новые. Все это самым отрицательным образом отражается на здоровье. Следует учесть также следующее. Сидячий образ жизни приводит к тому, что хрящ постепенно становится менее эластичным, теряет гибкость. Это может повлечь снижение амплитуды дыхательных движений и потерю гибкости тела. Но особенно сильно от неподвижности или малой подвижности страдают суставы.

ЗАКЛЮЧЕНИЕ

В результате написания выпускной квалификационной работы был создан новый метод сравнения изображений, который моделирует человеческое восприятие, используя для этого специальные цветовые пространства, а также функцию чувствительности контраста. На основе предложенного метода написаны программы сравнения изображений, а также программа для полуавтоматического тестирования алгоритмов сжатия изображений с потерями качества, которая получала на вход исходные и восстановленные после сжатия изображения, сравнивала их.

В ходе работы обнаружились направления, которые, возможно, могут повысить точность сравнения изображений. Среди них наиболее интересными представляется использование других функций чувствительности контраста, а также учет ограниченности разрешающей способности глаза.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

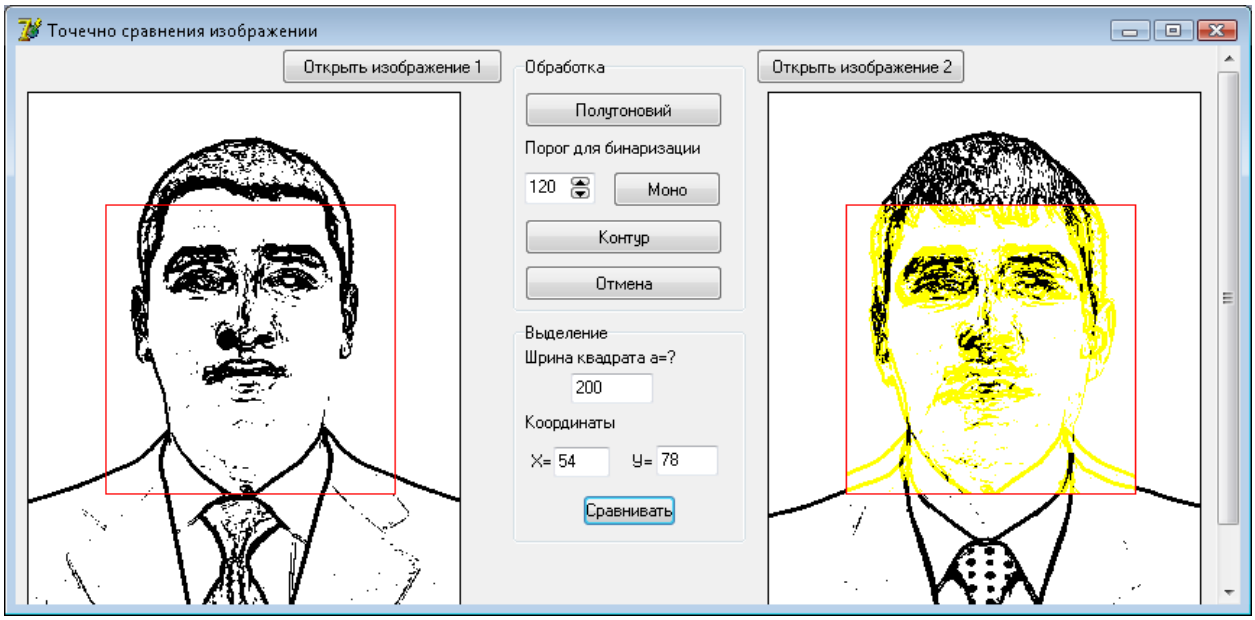
1. И.А.Каримовнинг “Мамлакатимизда демократик ислоҳотларни янада чуқурлаштириш ва фуқаролик жамиятини ривожлантириш концепцияси” 2010 й.
2. И.А.Каримовнинг “Юксак маънавият енгилмас куч” 2007 й.
3. И.А.Каримов. Мировой финансово-экономический кризис, пути и меры по его преодолению в условиях Узбекистана. Ташкент, март 2009.
4. М.Фленов. Библия Delphi. «БХВ-Петербург». Санкт-Петербург. 2011 г.
5. М.Фленов. Программирование в Delphi с глазами хакера. «БХВ-Петербург». Санкт-Петербург. 2006 г.
2. J.L. Mannos, D.J. Sakrison "The Effects of Visual Fidelity Criterion on the Encoding of Images" // IEEE Transactions on Information Theory IT-20(4) 2004, pp. 525-536.
3. A. B. Watson “The Cortex Transform: Rapid Computation of Simulated Neural Images” // CVGIP, 39 2007 pp. 311-327.
4. Shapiro, Stockmann "ComputerVision" // Aug. 99, Chapter 8
5. Charles Jacobs, Adam Finkelstein, David Salesin "Fast Multiresolution Image Quering" // SIGGRAPH-95 Proceedings, pp 277-286.
6. E. Kopylov, A. Khodulev, V. Volevich “The comparison of Illumination Maps Techniques in Computer Graphics Software” // 8-я международная конференция по компьютерной графике и визуализации ГрафиКон-98 (Труды конференции) ISBN 5-89209-294-1-ВМК МГУ 1998 стр. 146-153
7. Mahesh Ramasubramanian, Sumanta Pattanaik, Donald Greenberg "A Perceptually Based Physical Error Metric for Realistic Image Sythesis" // SIGGRAPH-99 Proceedings.
8. Ajeetkumar Gaddipatti, Raghu Machiraju, Roni Yagel "Steering Image Generation with Wavelet Based Perceptual Metric" // EuroGraphics-97, Volume 16, Number 3, pp.C-241 – C-251.

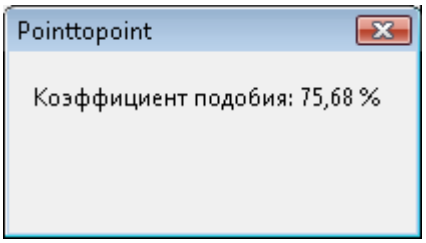
9. Laszlo Neumann, Kresimir Matkovic, Werner Purgathofer “Perception Based Color Image Difference” // EuroGraphics’98 Vol. 17(98) #3 pp.233-241.
10. G.C. Higgins “Image Quality Criteria” // Journal of Applied Photographic Engeneering. 1997 3(2), pp. 53-60
11. David Travis "Effective Color Displays. Theory and practice" // pp.89-100; Academic Press 1991 ISBN 0-12-697690-2.
12. Прэтт У.К. Цифровая обработка изображений: Кн. 1. – М.: Мир, 1992. – 367 с.
13. Прэтт У.К. Цифровая обработка изображений: Кн. 2. – М.: Мир, 1992. – 425 с.
14. Садыков С.С., Кан В.Н., Самандаров И.Р. Методы выделения структурных признаков изображений. Ташкент: Фан, 1990. 104с.
15. С. Симонович, Г. Евсеев. Программирование Delphi. М., 2000 г.
16. В. В. Форонов. Turbo Pascal 7.0. Практика програмирования. Учебное пособие.-М.: “Нолидж”, 1999.-432 с.
17. «Охрана труда в вычислительных центрах» под редакцией Ю.Г. Сибаров, Н.Н. Сколотнев, В.К. Васин, В.Н. Нагинаев, - М.: Наука, 1994 г.

ПРИЛОЖЕНИЕ

Полученные результаты

Выделена часть цветного изображения





Результат сравнения (Контур)

Код программы

FR_NEW

```
unit UQPForm;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, StdCtrls,
  UQPixels, ComCtrls, ExtCtrls, Math, Jpeg, Buttons, Dialogs, ExtDlgs;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    GroupBox1: TGroupBox;
    RunTest: TButton;
    rgGetSet: TRadioGroup;
    cbBPP: TComboBox;
    chIndex: TCheckBox;
    Label1: TLabel;
    TestAll: TButton;
    rgEffects: TRadioGroup;
    OpenPictureDialog1: TOpenPictureDialog;
    LoadPic: TBitBtn;
    chKeep: TCheckBox;
    btReset: TButton;
    BitBtn1: TBitBtn;
    SavePictureDialog1: TSavePictureDialog;
    procedure FormCreate(Sender: TObject);
```

```
    procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,
      Y: Integer);
    procedure FormMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormMouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormDestroy(Sender: TObject);
    procedure RunTestClick(Sender: TObject);
    procedure cbBPPChange(Sender: TObject);
    procedure TestAllClick(Sender: TObject);
    procedure rgEffectsClick(Sender: TObject);
    procedure LoadPicClick(Sender: TObject);
    procedure btResetClick(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  public
    procedure TestOneMode(BPP: integer; ByIndex, SetPix, DrawBmp: boolean);
    procedure Blur;
    procedure FlyImage;
    procedure LoadPicture;
  end;

const
  SomeColors: array[0..7] of TColor =
  (clBlack, clWhite, clRed, clGreen, clBlue, clGray, clLime, clAqua);
  BPPs: array[0..6] of integer = (1, 4, 8, 15, 16, 24, 32);

var
  Form1: TForm1;
  Bmp, NewBmp: TBitmap;
  QP, QP2: TQuickPixels;
```

```
  Freq: int64;
  TestBPP, Cnt: integer;
  Effect: integer;
  w, h: integer;
  x0: integer = -1;
  y0: integer = -1;
  FullRct, Rct: TRect;
implementation

{$R *.DFM}
{$R PIC.RES}

procedure TForm1.FormCreate(Sender: TObject);
var
  RS: TResourceStream;
  JP: TJpegImage;
begin
  cbBPP.ItemIndex := 6;
  TestBpp := 32;
  rgGetSet.ItemIndex := 0;
  RS := TResourceStream.Create(HInstance, 'PICT', 'RT_RCDATA');
  JP := TJpegImage.Create;
  Bmp := TBitmap.Create;
  try
    JP.LoadFromStream(RS);
    Bmp.Assign(JP);
  finally
    JP.Free;
    RS.Free;
  end;
```

```
  NewBmp := TBitmap.Create;
  QP := TQuickPixels.Create;
  QP2 := TQuickPixels.Create;
  QueryPerformanceFrequency(Freq);
  Cnt := 70000000 div Freq;
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
var
  xi, yi, xx, yy, ifi: integer;
  r, g, b: byte;
  c: TColor;

  procedure CalcXY(xi, yi, x, y, x0, y0, w1, h1: Integer; var xx, yy: Integer);
  var
    xb, yb, cf: double;
  begin
       cf := y * xi - yi * x;
    if (xi < x) then
    begin
      xb := 0;
      yb := cf / (xi - x);
      if yb < 0 then
      begin
        xb := cf / (y - yi);
        yb := 0;
      end;
      if yb > h1 then
      begin
```

```
        xb := (h1 * (xi - x) - cf) / (yi - y);
        yb := h1;
      end;
    end
    else
    begin
      xb := w1;
      yb := (cf + w1 * (yi - y)) / (xi - x);
      if yb < 0 then
      begin
        xb := cf / (y - yi);
        yb := 0;
      end;
      if yb > h1 then
      begin
        xb := (h1 * (xi - x) - cf) / (yi - y);
        yb := h1;
      end;
    end;
    if xb <> x then
      xx := Round(xb + (x0 - xb) * (xi - xb) / (x - xb));
    if yb <> y then
      yy := Round(yb + (y0 - yb) * (yi - yb) / (y - yb));
  end;

begin
  case Effect of
    2: if PtInRect(FullRct, Point(x, y)) then
      begin
        for xi := 0 to w - 1 do
```

```
      for yi := 0 to h - 1 do
      begin
        ifi := Trunc(2 * (Hypot(x - xi, y - yi)));
        if ifi > 255 then
          ifi := 255;
        c := QP.getpixel(xi, yi);
        r := getRValue(c);
        g := getGValue(c);
        b := getBValue(c);
        r := (r * (255 - ifi) + 128 * ifi) shr 8;
        g := (g * (255 - ifi) + 128 * ifi) shr 8;
        b := (b * (255 - ifi) + 128 * ifi) shr 8;
        c := RGB(r, g, b);
        QP2.SetPixel(xi, yi, c);
      end;
    BitBlt(Canvas.Handle, 0, 0, w - 1, h - 1, NewBmp.Canvas.Handle, 0, 0,
      srccopy);
  end;
3: if (ssLeft in Shift) and PtInRect(FullRct, Point(x, y)) then
  begin
    xx := x0 + w - x;
    yy := y0 + h - y;
    for xi := 0 to w - 1 do
      for yi := 0 to h - 1 do
        QP2.SetPixel(xi, yi, QP.GetPixel((xi + xx) mod w, (yi + yy) mod h));
    BitBlt(Canvas.Handle, 0, 0, w - 1, h - 1, NewBmp.Canvas.Handle, 0, 0,
      srccopy);
  end;
4: if (ssLeft in Shift) and PtInRect(rct, point(x, y)) then
  begin
```

```
      for xi := 0 to w - 1 do
        for yi := 0 to h - 1 do
        begin
          CalcXY(xi, yi, x - Ord(xi = x), y, x0, y0, w, h, xx, yy);
          if (xx >= 0) and (yy >= 0) and (xx < w) and (yy < h) then
          begin
            c := QP.GetPixel(xx, yy);
            QP2.SetPixel(xi, yi, c);
          end;
        end;
      BitBlt(Canvas.Handle, 0, 0, w - 1, h - 1, NewBmp.Canvas.Handle, 0, 0,
        srccopy);
    end;
  end;
end;

procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  i, j: integer;
  col: TColor;
begin
  if PtInRect(rct, Point(x, y)) then
  begin
    case Effect of
      2:
        begin
          Col := RGB(128, 128, 128);
          for i := 0 to w - 1 do
            for j := 0 to h - 1 do
```

```
            QP2.SetPixel(i, j, col);
          BitBlt(Canvas.Handle, 0, 0, w, h, NewBmp.Canvas.Handle, 0, 0,
           srccopy);
        end;
      3: if ssLeft in Shift then
        begin
          x0 := x;
          y0 := y;
          Screen.Cursor := crSizeAll;
        end;
      4: if ssLeft in Shift then
        begin
          x0 := x;
          y0 := y;
          Screen.Cursor := crHandPoint;
        end;
    end;
  end;
end;

procedure TForm1.FormMouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if (Effect=4) and chKeep.Checked then begin
    Bmp.Assign(NewBmp);
    QP.Attach(Bmp);
    QP2.Attach(NewBmp);
  end;
  x0 := -1;
  y0 := -1;
```

```
  Screen.Cursor := crDefault;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  QP.Free;
  QP2.Free;
  Bmp.Free;
  NewBmp.Free;
end;

procedure TForm1.TestOneMode(BPP: integer; ByIndex, SetPix, DrawBmp:
boolean);
var
  SmallBmp: TBitmap;
  Tim1, Tim2: int64;
  Seconds, OverHead: double;
  SetGet, sByIndex: string;
  i, j: DWord;
  Col: TColor;
begin
  SmallBmp := TBitmap.Create;
  SmallBmp.Width := 8;
  SmallBmp.Height := 8;
  case BPP of
    1: SmallBmp.PixelFormat := pf1bit;
    4: SmallBmp.PixelFormat := pf4bit;
    8: SmallBmp.PixelFormat := pf8bit;
    15: SmallBmp.PixelFormat := pf15bit;
    16: SmallBmp.PixelFormat := pf16bit;
```

```
  24: SmallBmp.PixelFormat := pf24bit;
  32: SmallBmp.PixelFormat := pf32bit;
end;
if DrawBmp then
begin
  Refresh;
  SmallBmp.Canvas.Brush.Color := clWhite;
  SmallBmp.Canvas.FillRect(Rect(0, 0, SmallBmp.Width, SmallBmp.Height));
end;
QueryPerformanceCounter(Tim1);
for i := 1 to Cnt * 1000000 do
begin
  j := i and 7;
end;
QueryPerformanceCounter(Tim2);
OverHead := (Tim2 - Tim1 + j -j) / Freq;
if SetPix then
  SetGet := 'Set'
else
  SetGet := 'Get';
if BPP > 8 then
  ByIndex := False;
if ByIndex then
  sByIndex := 'Indx'
else
  sByIndex := '';
QP.Attach(SmallBmp);
if QP.BPP <= 8 then
  QP.ByPaletteIndex := ByIndex
else
```

```
  QP.ByPaletteIndex := False;
if SetPix then
begin
  QueryPerformanceCounter(Tim1);
  for i := 1 to Cnt * 1000000 do
  begin
    j := i and 7;
    QP.SetPixel(j, j, clRed);
  end;
  QueryPerformanceCounter(Tim2);
  if DrawBmp then
    Canvas.StretchDraw(Rect(0, 0, 80, 80), SmallBmp);
  Seconds := (Tim2 - Tim1) / Freq;
end
else
begin
  for i := 0 to 7 do
    for j := 0 to 7 do
      SmallBmp.Canvas.Pixels[i, j] := SomeColors[j];
  QueryPerformanceCounter(Tim1);
  for i := 1 to Cnt * 1000000 do
  begin
    j := i and 7;
    Col := QP.GetPixel(j, j);
  end;
  QueryPerformanceCounter(Tim2);
  if DrawBmp then
    Canvas.StretchDraw(Rect(0, 0, 80, 80), SmallBmp);
  Seconds := (Tim2 - Tim1) / Freq;
end;
```

```
  Memo1.Lines.Add(Format('%2d bpp %s%4s: %5f MP/s',
    [BPP, SetGet, sByIndex, Cnt / (Seconds - OverHead)]));
  SmallBmp.Free;
end;

procedure TForm1.RunTestClick(Sender: TObject);
begin
  Effect := 0;
  TestOneMode(TestBpp, chIndex.Checked, rgGetSet.ItemIndex = 1, True);
end;

procedure TForm1.cbBPPChange(Sender: TObject);
begin
  TestBpp := BPPs[cbBPP.ItemIndex]
end;

procedure TForm1.TestAllClick(Sender: TObject);
var
  i: Integer;
  GetSet, byIndx: Boolean;
begin
  Refresh;
  Effect := 0;
  Memo1.Clear;
  for GetSet := False to True do
  begin
    for i := 0 to 6 do
      for byIndx := False to (i < 3) do
        TestOneMode(BPPs[i], byIndx, GetSet, False);
    Memo1.Lines.Add('');
```

```
  end;
end;

procedure TForm1.rgEffectsClick(Sender: TObject);
begin
  Refresh;
  Effect := rgEffects.ItemIndex;
  case Effect of
    0: Blur;
    1: FlyImage;
  else
    begin
      NewBmp.Assign(Bmp);
      Canvas.Draw(0, 0, NewBmp);
      QP.Attach(Bmp);
      QP2.Attach(NewBmp);
      w := QP.Width;
      h := QP.Height;
      Rct := Rect(5, 5, w - 6, h - 6);
      FullRct := Rect(0, 0, w - 1, h - 1);
    end;
  end;
end;

procedure TForm1.Blur;
var
  fl: array[-1..1, -1..1] of integer;
  bm: TBitmap;
  i, j, k, l: integer;
  r, g, b: integer;
```

```
  c: tcolor;
begin
  for k := -1 to 1 do
    for l := -1 to 1 do
      fl[k, l] := 1;
  fl[0, 0] := 4;
  bm := TBitmap.Create;
  bm.width := 200;
  bm.height := 200;
  bm.PixelFormat := pf24bit;
  QP.Attach(bm);
  for i := 0 to 9 do
  begin
    bm.Canvas.MoveTo(0, 10 + i * 20);
    bm.Canvas.LineTo(200, 10 + i * 20);
    bm.Canvas.MoveTo(10 + i * 20, 0);
    bm.Canvas.LineTo(10 + i * 20, 200);
  end;
  canvas.Draw(0, 0, bm);
  for i := 1 to QP.Width - 2 do
    for j := 1 to QP.Height - 2 do
    begin
      r := 0;
      b := 0;
      g := 0;
      for k := -1 to 1 do
        for l := -1 to 1 do
        begin
          c := QP[i + k, j + l];
          inc(r, fl[k, l] * GetRValue(c));
```

```
          inc(g, fl[k, l] * GetGValue(c));
          inc(b, fl[k, l] * GetBValue(c));
        end;
      QP[i, j] := RGB(r div 12, g div 12, b div 12);
    end;
  canvas.Draw(0, 210, bm);
  bm.free;
end;

procedure TForm1.FlyImage;
var
  k, x_new, y_new, x0, y0: integer;
  fac: double;
  cosphi, sinphi, x, y, dx, dy: integer;

const
  BinaryFactor = 10;

  function rnd(const x, y: Integer): TPoint;
  begin
    Result.X := x shr BinaryFactor;
    Result.Y := y shr BinaryFactor;
  end;

begin
  NewBmp.Assign(Bmp);
  QP.Attach(Bmp);
  QP2.Attach(NewBmp);
  w := QP.Width;
  h := QP.Height;
```

```
  x0 := (w div 2);
  y0 := (h div 2);
  for k := 1 to 1080 do
  begin
    fac := Sqr(1080 / k);
    dx := round(w * 512 * cos(k * Pi / 400) * (1 - fac));
    dy := round(w * 512 * sin(k * Pi / 400) * (1 - fac));
    cosphi := round(fac * cos(k * Pi / 180) * (2 shl (BinaryFactor - 1)));
    sinphi := round(fac * sin(k * Pi / 180) * (2 shl (BinaryFactor - 1)));
    for x_new := 0 to w - 1 do
    begin
      y := ((-x_new + x0) * sinphi - y0 * cosphi) + (y0 shl BinaryFactor) + dx;
      x := ((-x0 + x_new) * cosphi - y0 * sinphi) + (x0 shl BinaryFactor) + dy;
      for y_new := 0 to h - 1 do
      begin
        with rnd(x, y) do
          if (x >= 0) and (x < w) and (y >= 0) and (y < h) then
            QP2.SetPixel(x_new, y_new, QP.getpixel(x, y))
          else
            QP2.SetPixel(x_new, y_new, clSilver);
        inc(y, cosphi);
        inc(x, sinphi);
      end;
    end;
    BitBlt(Canvas.Handle, 0, 0, w, h, NewBmp.canvas.handle, 0, 0, srccopy);
  end;
end;

procedure TForm1.LoadPicClick(Sender: TObject);
begin
```

```
  if OpenPictureDialog1.Execute then
    LoadPicture;
  Refresh;
end;

procedure TForm1.btResetClick(Sender: TObject);
begin
  if OpenPictureDialog1.FileName<>'' then
    LoadPicture;
  Refresh;
end;

procedure TForm1.LoadPicture;
var
  JP: TJpegImage;
  Bm: TBitmap;
  ScaleX, ScaleY: Double;
  NewX, NewY: Integer;
begin
  JP := TJpegImage.Create;
  Bm := TBitmap.Create;
  try
    if UpperCase(ExtractFileExt(OpenPictureDialog1.FileName))='.BMP' then
      BM.LoadFromFile(OpenPictureDialog1.FileName)
    else begin
      JP.LoadFromFile(OpenPictureDialog1.FileName);
      Bm.Assign(JP);
    end;
    ScaleX := Bm.Width / 400;
    ScaleY := Bm.Height / 400;
```

```
  if ScaleX > ScaleY then
    if ScaleX > 1 then
    begin
      NewX := 400;
      NewY := Round(Bm.Height / ScaleX);
      Bmp.Width := NewX;
      Bmp.Height := NewY;
      SetStretchBltMode(Bmp.Canvas.Handle, HALFTONE);
      StretchBlt(Bmp.Canvas.Handle, 0, 0, NewX, NewY,
        Bm.Canvas.Handle, 0, 0, Bm.Width, Bm.Height, SrcCopy);
    end
    else
      Bmp.Assign(Bm)
  else if ScaleY > 1 then
  begin
    NewY := 400;
    NewX := Round(Bm.Width / ScaleY);
    Bmp.Width := NewX;
    Bmp.Height := NewY;
    SetStretchBltMode(Bmp.Canvas.Handle, HALFTONE);
    StretchBlt(Bmp.Canvas.Handle, 0, 0, NewX, NewY,
      Bm.Canvas.Handle, 0, 0, Bm.Width, Bm.Height, SrcCopy);
  end
  else
    Bmp.Assign(Bm)
finally
  JP.Free;
  Bm.Free;
  Effect:=0;
end;
```

```
end;

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  if Assigned(NewBmp) then
    if SavePictureDialog1.Execute then
      NewBmp.SaveToFile(SavePictureDialog1.FileName);
end;

end.
```

PointToPoint

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtDlgs, StdCtrls, Buttons, ExtCtrls, Spin, XPMan;

type
  TForm1 = class(TForm)
    Image1: TImage;
    Image2: TImage;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    OpenPictureDialog1: TOpenPictureDialog;
    BitBtn3: TBitBtn;
```

```
GroupBox1: TGroupBox;
Edit1: TEdit;
RadioButton1: TRadioButton;
RadioButton2: TRadioButton;
Label1: TLabel;
Label2: TLabel;
Edit2: TEdit;
Label3: TLabel;
Label4: TLabel;
Edit3: TEdit;
Button1: TButton;
GroupBox2: TGroupBox;
Button2: TButton;
Label5: TLabel;
SpinEdit1: TSpinEdit;
BitBtn4: TBitBtn;
BitBtn5: TBitBtn;
BitBtn6: TBitBtn;
Image3: TImage;
Button3: TButton;
Button4: TButton;
Image4: TImage;
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn2Click(Sender: TObject);
procedure BitBtn3Click(Sender: TObject);
procedure RadioButton1Click(Sender: TObject);
procedure RadioButton2Click(Sender: TObject);
procedure Edit1Change(Sender: TObject);

procedure Image1MouseUp(Sender: TObject; Button: TMouseButton;
```

```
    Shift: TShiftState; X, Y: Integer);
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure BitBtn4Click(Sender: TObject);
  procedure BitBtn5Click(Sender: TObject);
  procedure BitBtn6Click(Sender: TObject);
  procedure Button3Click(Sender: TObject);
  procedure Button4Click(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

uses Unit2;

{$R *.dfm}

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
if OpenPictureDialog1.Execute then
Image1.Picture.LoadFromFile(OpenPictureDialog1.FileName);
Image1.Picture.SaveToFile('image01.bmp');
//Image4.Picture:=Image1.Picture;
end;
```

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
if OpenPictureDialog1.Execute then
Image2.Picture.LoadFromFile(OpenPictureDialog1.FileName);
Image2.Picture.SaveToFile('image02.bmp');
//Image5.Picture:=Image2.Picture;
end;

procedure TForm1.BitBtn3Click(Sender: TObject);
var
k,i,n,m,j:Integer;
f:real;
begin
k:=0;
n:=Image1.Picture.Width;
m:=Image1.Picture.Height;
for i:=1 to n do
for j:=1 to m do
 if  Image1.Canvas.Pixels[i,j]<>Image2.Canvas.Pixels[i,j] then k:=k+1;
f:=k/(n*m)*100;
 if k<>0 then ShowMessage(FloatToStr(f)+'% узгариш бор' )
   else ShowMessage('Узгариш йук');
Form1.Tag:=0;
   end;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
{   GroupBox1.Visible:=False;
   BitBtn3.Visible:=True;}
```

```
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
{BitBtn3.Visible:=False;
GroupBox1.Visible:=True;    }
end;

procedure TForm1.Edit1Change(Sender: TObject);
var
i,j,diag,n,m:Integer;
begin
if form1.tag=0 then Image1.Picture.LoadFromFile('Image01.bmp');
if form1.tag=10 then Image1.Picture:=Image3.Picture;
end;

procedure TForm1.Image1MouseUp(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
i,j,diag,n,m:Integer;
begin
diag:=StrToInt(Edit1.Text);
n:=x+diag;
m:=y+diag;
Edit2.Text:=IntToStr(x);
Edit3.Text:=IntToStr(y);
For i:=x to n do
begin
  Image1.Canvas.Pixels[i,y]:=clRed;
  Image1.Canvas.Pixels[i,m]:=clRed;
```

```
end;
For j:=y to m do
begin
  Image1.Canvas.Pixels[x,j]:=clRed;
  Image1.Canvas.Pixels[n,j]:=clRed;
end;

end;

procedure TForm1.Button1Click(Sender: TObject);
var
k,i,j,x,y,diag,n,m:Integer;
f:real;
begin
k:=0;
diag:=StrToInt(Edit1.Text);
x:=StrToInt(Edit2.Text);
y:=StrToInt(Edit3.Text);
n:=x+diag;
m:=y+diag;
For i:=x to n do
begin
  Image2.Canvas.Pixels[i,y]:=clRed;
  Image2.Canvas.Pixels[i,m]:=clRed;
end;
For j:=y to m do
begin
  Image2.Canvas.Pixels[x,j]:=clRed;
  Image2.Canvas.Pixels[n,j]:=clRed;
end;
```

```
for i:=x to n do
for j:=y to m do
 if  Image1.Canvas.Pixels[i,j]<>Image2.Canvas.Pixels[i,j] then
    Begin
      k:=k+1;
      Image2.Canvas.Pixels[i,j]:=ClYellow;
    end;
  f:=100-(k/(diag*diag))*100;
 if k<>0 then ShowMessage('Коэффициент подобия: '+FloatToStr(f)+' %')
   else ShowMessage('Коэффициент подобия: 100 %');
   Form1.Tag:=0;
end;

procedure TForm1.Button2Click(Sender: TObject);
var
n1,n2,m1,m2,i,j:integer;
urt,r,g,b:Byte;
begin
form1.tag:=16;
 m1:=Image1.Picture.Width;
 n1:=Image1.Picture.Height;
 for i:=0 to m1-1 do
 for j:=0 to n1-1 do
 begin
   R:=GetRValue(Image1.Canvas.Pixels[i,j]);
   G:=GetGValue(Image1.Canvas.Pixels[i,j]);
   B:=GetBValue(Image1.Canvas.Pixels[i,j]);
   Urt:=(R+G+B) div 3;
   Image1.Canvas.Pixels[i,j]:=RGB(Urt,Urt,Urt);
```

```
 end;
 m2:=Image2.Picture.Width;
 n2:=Image2.Picture.Height;
 for i:=0 to m1-1 do
 for j:=0 to n1-1 do
 begin
   R:=GetRValue(Image2.Canvas.Pixels[i,j]);
   G:=GetGValue(Image2.Canvas.Pixels[i,j]);
   B:=GetBValue(Image2.Canvas.Pixels[i,j]);
   Urt:=(R+G+B) div 3;
   Image2.Canvas.Pixels[i,j]:=RGB(Urt,Urt,Urt);
 end;

end;

procedure TForm1.BitBtn4Click(Sender: TObject);
var m1,n1,m2,n2,i,j: integer;
    R,G,B,Urt,Porog: byte;
begin
 Porog:=SpinEdit1.Value;
 m1:=Image1.Picture.Width;
 n1:=Image1.Picture.Height;
 for i:=0 to m1-1 do
 for j:=0 to n1-1 do
 begin
   R:=GetRValue(Image1.Canvas.Pixels[i,j]);
   G:=GetGValue(Image1.Canvas.Pixels[i,j]);
   B:=GetBValue(Image1.Canvas.Pixels[i,j]);
   Urt:=(R+G+B) div 3;
   if Urt<Porog then Image1.Canvas.Pixels[i,j]:=clBlack else
```

```
                Image1.Canvas.Pixels[i,j]:=clWhite;
  end;
 m2:=Image1.Picture.Width;
 n2:=Image1.Picture.Height;
 for i:=0 to m2-1 do
 for j:=0 to n2-1 do
 begin
  R:=GetRValue(Image2.Canvas.Pixels[i,j]);
  G:=GetGValue(Image2.Canvas.Pixels[i,j]);
  B:=GetBValue(Image2.Canvas.Pixels[i,j]);
  Urt:=(R+G+B) div 3;
  if Urt<Porog then Image2.Canvas.Pixels[i,j]:=clBlack else
                Image2.Canvas.Pixels[i,j]:=clWhite;
  end;

end;

procedure TForm1.BitBtn5Click(Sender: TObject);
begin
Image1.Picture.LoadFromFile('Image01.bmp');
Image2.Picture.LoadFromFile('Image02.bmp');
Form1.Tag:=0;
end;

procedure TForm1.BitBtn6Click(Sender: TObject);
begin
Form1.Tag:=10;
form2.ShowModal;
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
var
k,i,j:integer;
r1,g1,b1,r2,g2,b2,u1,u2:byte;
f:real;
begin
k:=0;
for i:=0 to Image1.Picture.Width do
for j:=0 to Image1.Picture.Height do
 Begin
   r1:=GetRValue(Image1.Canvas.Pixels[i,j]);
   g1:=GetGValue(Image1.Canvas.Pixels[i,j]);
   b1:=GetBValue(Image1.Canvas.Pixels[i,j]);
   u1:=(r1+g1+b1) div 3;
   r2:=GetRValue(Image2.Canvas.Pixels[i,j]);
   g2:=GetGValue(Image2.Canvas.Pixels[i,j]);
   b2:=GetBValue(Image2.Canvas.Pixels[i,j]);
   u2:=(r2+g2+b2) div 3;
  if u1<>u2 then k:=k+1;
  end;
 f:=k/(Image2.Picture.Width*Image2.Picture.Height)*100;
 ShowMessage(FloatToStr(f)+'% узгариш бор');
end;

procedure TForm1.Button4Click(Sender: TObject);
var
k,i,j,x,y,diag,n,m:Integer;
f:real;
r1,g1,b1,r2,g2,b2,u1,s1,u2,s2:byte;
begin
```

```
 k:=0;
diag:=StrToInt(Edit1.Text);
x:=StrToInt(Edit2.Text);
y:=StrToInt(Edit3.Text);
n:=x+diag;
m:=y+diag;
For i:=x to n do
begin
  Image2.Canvas.Pixels[i,y]:=clRed;
  Image2.Canvas.Pixels[i,m]:=clRed;
end;
For j:=y to m do
begin
  Image2.Canvas.Pixels[x,j]:=clRed;
  Image2.Canvas.Pixels[n,j]:=clRed;
end;

for i:=x to n do
for j:=y to m do
 begin
   r1:=GetRValue(Image1.Canvas.Pixels[i,j]);
   g1:=GetGValue(Image1.Canvas.Pixels[i,j]);
   b1:=GetBValue(Image1.Canvas.Pixels[i,j]);
   u1:=(r1+g1+b1) div 3;
    //s1:=s1+u1;
   r2:=GetRValue(Image2.Canvas.Pixels[i,j]);
   g2:=GetGValue(Image2.Canvas.Pixels[i,j]);
   b2:=GetBValue(Image2.Canvas.Pixels[i,j]);
   u2:=(r2+g2+b2) div 3;
//   s2:=s2+u2;
```

```
   if  u1<>u2 then k:=k+1;
 end;

f:=(k/(diag*diag))*100;
 if k>10 then ShowMessage(FloatToStr(f)+'%ов есть разница' )
   else ShowMessage('Неть изменение');
// ShowMessage(IntToStr(s1)+' '+IntToStr(s2));
   Form1.Tag:=0;
end;

end.
```