СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность

Восстановление искаженных изображений является одной из наиболее интересных и важных проблем в задачах обработки изображений – как с теоретической, так и с практической точек зрения. Частными случаями являются размытие из-за неправильного фокуса и смаз – эти дефекты, с которым каждый из вас хорошо знаком, очень сложны в исправлении – именно они и выбраны темой статьи. С остальными искажениями (шум, неправильная экспозиция, дисторсия) человечество научилось эффективно бороться, соответствующие инструменты есть в каждом уважающем себя фоторедакторе.

Почему же для устранения смаза и расфокусировки практически ничего нету (unsharp mask не в счет) – может быть это в принципе невозможно? На самом деле возможно – соответствующий математический аппарат начал разрабатываться примерно 70 лет назад, но, как и для многих других алгоритмов обработки изображений, все это нашло широкое применение только в недавнее время.

Цели и перечень задач

1.Проектирование графического пользовательского интерфейса для загрузки из указанного места на диске исходного изображения.
2.Его разбиения на определённое пользователем количество прямоугольных фрагментов (путём наложения сетки разбиения с заданным количеством горизонтальных и вертикальных ячеек)
3.Указанием соотношения пересечения фрагментов, углов их поворотов с сохранением изображений фрагментов в отдельных файлах, последующем чтении файлов фрагментов изображения, их анализа и объединения в единое целое.

1. МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ ПРОЦЕССА ПОИСКА ОБЩИХ ФРАГМЕНТОВ В ИЗОБРАЖЕНИЯХ

1.1 Понятие изображения

ИЗОБРАЖЕНИЕ:

Одна из разновидностей информационной модели, в которой используется картинный (наглядный) способ представления информации оператору. Основной характеристикой модели этого класса является то, что образ, формирующийся у наблюдателя при его восприятии, близок к образу объекта, воспроизводимого в модели. Закономерности и свойства восприятия (предметность, целостность, структурность, избирательность, константность, апперцепция) проявляются при работе с изображениями почти во всей полноте. Наблюдатель имеет возможность оперировать образом, формирующимся при восприятии изображения почти так же, как он оперирует перцептивным образом реального объекта. Сложившиеся в процессе восприятия реальных объектов навыки наблюдения используются и при восприятии изображения, благодаря чему упрощается задача обучения и тренировки наблюдателя; наблюдателю нет необходимости осваивать специальный код, способы кодирования и декодирования поступающей информации, мысленного преобразования образа модели в образ объекта. Являясь многомерным отображением, изображение позволяет передавать наблюдателю значительную по объему информацию, при этом в процессе ее восприятия реализуются широкие возможности различения сигналов: количество различаемых градаций того или иного измерения сигнала на порядок превосходит количество точно идентифицируемых градаций. Вместе с тем точность оценки величин при восприятии изображения ограничена возможностями зрительного анализатора (острота зрения, пороги различения, глазомер и др.). Кроме того, там, где требуется передавать небольшую по объему информацию, изображение является неэкономным и ненадежным средством ее передачи; свойственная ему избыточность может быть помехой для зрительной селекции сигналов. Наиболее широко используемыми в технике видами изображений являются фотография, телевизионное, радиолокационное и киноизображение.

1.2 МАТЕМАТИЧЕСКИЕ МОДЕЛИ ИЗОБРАЖЕНИЙ

Модели непрерывных изображений

Компьютерная обработка изображений возможна после преобразования сигнала изображения из непрерывной формы в цифровую форму. Эффективность обработки зависит от адекватности модели, описывающей изображение, необходимой для разработки алгоритмов обработки. При этом необходимо учитывать влияние передающей и приемной систем и канала связи на сигнал изображения. Модель изображения представляет систему функций, описывающих существенные характеристики изображения: функцию яркости, отражающую изменение яркости в плоскости изображения, пространственные спектры и спектральные интенсивности изображений, функции автокорреляции. Канал изображения содержит оптическую систему, оптико – электрический преобразователь, устройство аналого – цифрового преобразования (АЦП) и цифровой обработки сигналов изображения. В общем случае непрерывное изображение может быть представлено функцией пяти аргументов: трех пространственных координат, времени и длины волны электромагнитного излучения. Упрощения модели пространственно – временных сигналов в некотором диапазоне волн $f(x, y,z,t,\lambda)$ приводят к моделям пространственно – временного сигнала $f(x, y,z,t)$, пространственного сигнала $f(x, y,z)$, временного сигнала $f(t)$. Здесь x, y,z – пространственные координаты, t – время, λ – длина волны электромагнитного излучения.

Пространственные спектры изображений

При обработке изображений широко используется анализ спектров изображений. Спектр изображения получают прямым двумерным преобразованием Фурье функции, описывающей изображение [12]:

$$F(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\exp(-i(\omega_x x + \omega_y y))dxdy ,$$

Функция $\exp(-i(\omega_x x+\omega_y y))$ при фиксированных значениях пространственных частот описывает плоскую волну в плоскости изображения (x,y) (в соответствии с рисунком 2.1).

Формула (2.1) связывает вещественную функцию, описывающую яркость изображения $f(x,y)$ с комплексной функцией частоты – спектром изображения $F(\omega_x,\omega_y)$:

$$F(\omega_x,\omega_y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\cos(\omega_x x+\omega_y y)dxdy+ \\ +i\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(-f(x,y))\sin(\omega_x x+\omega_y y)dxdy=Re(\omega_x,\omega_y)+i\,Im(\omega_x,\omega_y) \quad (2.2)$$

где $Re(\omega_x,\omega_y)$ - реальная часть спектра; $Im(\omega_x,\omega_y)$ - мнимая часть спектра.

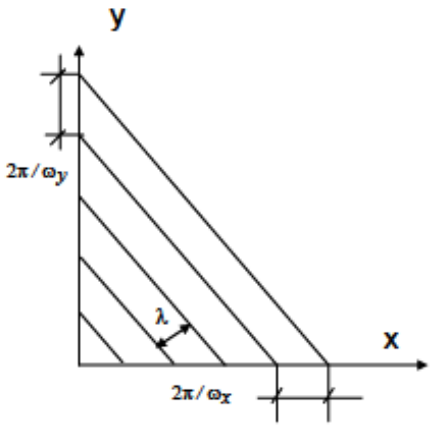


Рисунок 2.1 Определение пространственных частот изображения.

Амплитуда и фаза спектра определяются по формулам (2.3) и (2.4) соответственно:

$$\mathrm{F}(\omega_x,\omega_y)=\sqrt{Re(ш_x,ш_y)^2+Im(ш_x,ш_y)^2}, \quad (2.3)$$

$$\varphi(ш_x,ш_y)=\operatorname{arctg}(Im(ш_x,ш_y)/Re(ш_x,ш_y)).$$

Из (2.3)

$$F(\omega_x,\omega_y)=\mathrm{F}(ш_x,ш_y)\exp(i\varphi(\omega_x,\omega_y)). \quad (2.4)$$

Обратное преобразование Фурье позволяет восстановить изображение по его спектру:

$$f(x,y)=(1/4\pi^2)\int_{-\infty}^{\infty}\int F(\omega_x,\omega_y)\exp(i(\omega_x x+\omega_y y))d\omega_x d\omega_y. \quad (2.5)$$

Спектральные интенсивности изображений

Спектральная интенсивность изображения характеризует распределение энергии по пространственным частотам. Она определяется как квадрат модуля спектра изображения:

$$S(\omega_x,\omega_y) = Re(\omega_x,\omega_y)^2 + Im(\omega_x,\omega_y)^2 = F^2(\omega_x,\omega_y). \qquad (2.6)$$

Для ее названия используются термины спектральная плотность и энергетический спектр

Энергия изображения определяется как интеграл энергетического спектра по пространственным частотам. В соответствии с теоремой Парсеваля энергия изображения может быть вычислена в соответствии с (2.7):

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f^2(x,y)dxdy = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left|F(\omega_x,\omega_y)\right|^2 d\omega_x d\omega_y . \qquad (2.7)$$

2.4 Вероятностные модели изображений и функции автокорреляции

Вероятностные модели изображений широко используются для описания изображений. Изображение в этом случае рассматривается как случайная функция пространственных координат (x,y) и времени t. Случайный процесс называется стационарным в широком смысле, если он имеет постоянные значения математического ожидания и дисперсии, а его автокорреляционная функция зависит не от координат, а от их разностей (сдвига). Случайный процесс называется стационарным в узком смысле, если его n-мерная плотность распределения вероятностей инвариантна к сдвигу. В этом случае не зависят от времени и моменты более высокого порядка, в частности, асимметрия и эксцесс. Случайный процесс описывается плотностью вероятности распределения яркости в изображении по пространственным координатам для некоторого фиксированного момента времени t p(x, y).

В соответствии с определением математическое ожидание (среднее значение) стационарного процесса в широком смысле

$$Mf=\xi=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)p(x,y)dxdy=\text{const.} \qquad (2.8)$$

Дисперсия

$$Df=\sigma^2=E(f(x,y)-\xi)^2=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(f(x,y)-\xi)^2 p(x,y)dxdy=\text{const.} \qquad (2.9)$$

Функция автокорреляции вычисляется в соответствии с (2.10):

$$R(\tau_x,\tau_y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)f(x-\tau_x,y-\tau_y)dxdy, \qquad (2.10)$$

Для действительной функции f автокорреляционная функция является действительной и четной. Спектр двумерной автокорреляционной функции изображения (прямое преобразование Фурье автокорреляционной функции) равен энергетическому спектру изображения (спектральной плотности мощности) по определению:

$$S(\omega_x,\omega_y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} R(\tau_x,\tau_y)\exp(-i(\omega_x\tau_x+\omega_y\tau_y))d\tau_x d\tau_y. \qquad (2.11)$$

Стационарный случайный процесс называется эргодическим, если любая его вероятностная характеристика может быть получена из одной реализации путем усреднения по времени. При этом среднее по времени равно среднему по ансамблю реализаций. Свойство эргодичности используется при оценке вероятностных характеристик изображений.

Критерии качества изображений

Качество изображения может определяться статистическими, спектральными, яркостными характеристиками изображения. В большинстве практических применений качество рассматривается как мера близости двух изображений: реального и идеального или преобразованного и исходного. При таком подходе можно оценивать как субъективную степень похожести изображений, так и получать объективные оценки параметров сигналов изображения: моменты первого и второго порядка разностного сигнала сравниваемых изображений, такие параметры преобразования как отношение С/Ш, коэффициенты сжатия информации и другие. Субъективные критерии – это критерии визуального восприятия, оцениваемые в процессе экспертизы некоторой группой наблюдателей (экспертов). Наибольшее распространение получил метод оценок, при котором наблюдатель оценивает качество изображения в баллах по определенной шкале, считая, что идеальное изображение имеет максимальный балл. Этот метод позволяет оценить такие характеристики изображения как правильность цветопередачи,

координатные искажения, чистоту переходов и др. Для интерпретации полученных экспертных оценок разработаны методы их представления, например построение кумулятивных кривых распределения оценок как функции от искажений. Поэтому некоторые результаты, рассматриваемые с точки зрения объективных оценок как одинаковые, визуально могут восприниматься различно. Однако объективные критерии используются при компьютерной обработке изображений в системах с автоматическим принятием решений. Функционирование автоматических компьютерных систем полностью подчинено математическим критериям, и качество их работы оценивается только объективными показателями. Понятно, что и качество изображений, используемых в этих системах, также должно оцениваться только по объективным критериям.

1.3 МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ ОПЕРАТОРОВ ПОВОРОТА И МАСШТАБИРОВАНИЯ ИЗОБРАЖЕНИЙ.

Оператор поворота заключается в математическом повороте матрицы. Любое изображение является матрицей. Мы создаём график на основе изображения, и с помощью математических операторов поворота матрицы поворачиваем данное нам изображение.

1.4 МЕТОДЫ СРАВНЕНИЯ ИЗОБРАЖЕНИЙ И ПОИСКА В НИХ ОБЩИХ ФРАГМЕНТОВ

Для начала мы сравниваем размеры фрагментов . В своей курсовой раоте я использовал C#, более безопасный язык по сравнению с C++. Для того что бы сравнить наш объяект класса Bitmap, так как ячейки Bitmap сравнивать нет смысла, то мы должны отсылаться на память данного объекта класса . Для этого сужествуте Unsafe код, который позволяет обратиться к памяти определенной ячейки и сравнить её.

2. ФОРМАТЫ ПРЕДСТАВЛЕНИЯ И ХРАНЕНИЯ ФАЙЛОВ ИЗОБРАЖЕНИЙ

2.1 Способы хранения изображений в памяти компьютера

•РАСТРОВОЕ ПРЕДСТАВЛЕНИЕ

Растровая графика – способ представления изображений в виде совокупности отдельных точек (пикселей) различных цветов и оттенков

Недостатки растровой графики:
1. для хранения и обработки изображений высокого качества необходим большой объем памяти;
2. сложно манипулировать отдельными объектами при редактировании изображений;
3. ухудшается качество при масштабировании.

Сферы применения растровой графики: • представление фотографий и фотореалистичных изображений

Параметры растровых изображений Разрешение (resolution) – это степень детализации изображения, число пикселей отводимых на единицу площади. Измеряется в точках на дюйм:
D P I – Dots Per Inch
1 дюйм X 1 дюйм = 300 точек X 300 точек = 300 dpi

•ВЕКТОРНОЕ ПРЕДСТАВЛЕНИЕ

Векторная графика – способ представления изображений в виде совокупности отдельных объектов (графических примитивов). Каждый примитив описывается математически относительно его узлов.

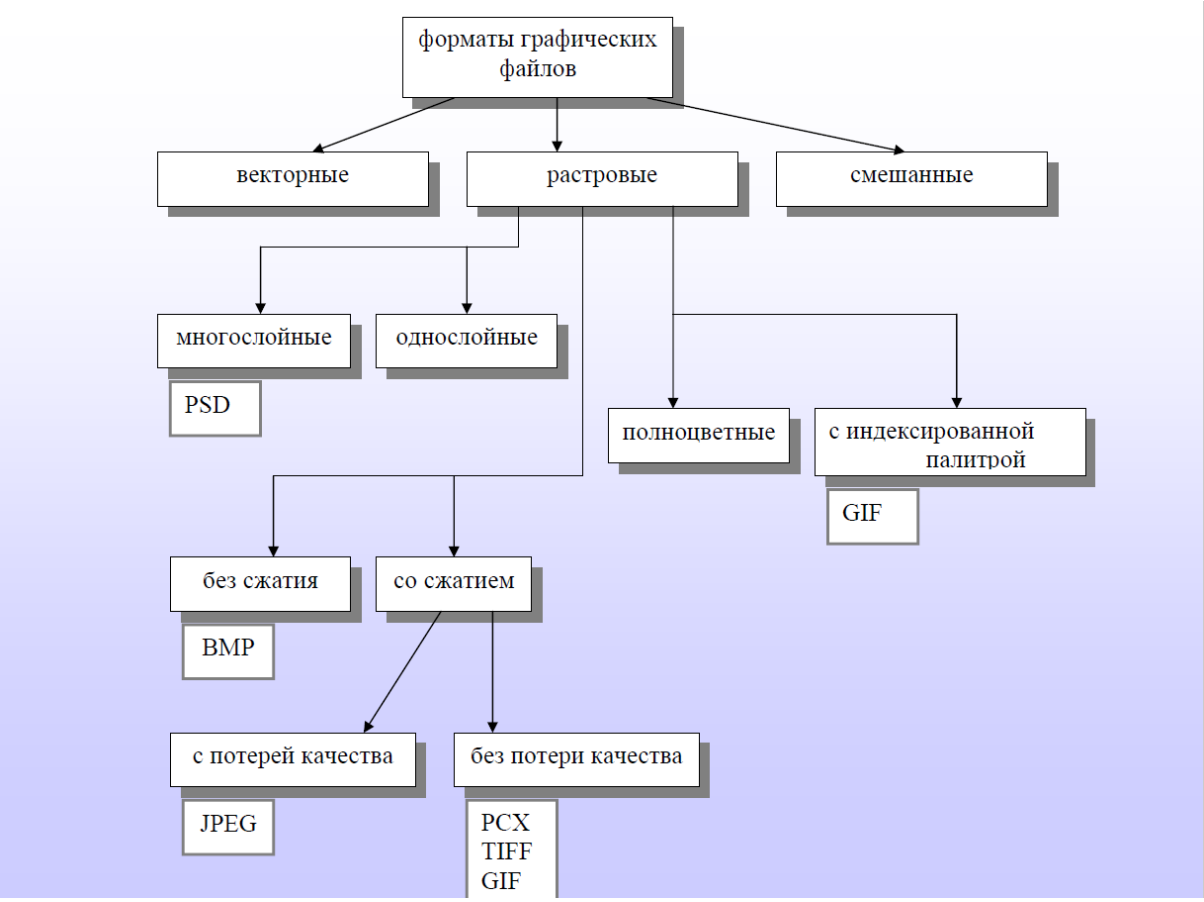
Достоинства векторной графики:
1. для хранения и обработки изображений высокого качества необходим небольшой объем памяти;
2. легко манипулировать отдельными объектами при редактировании изображений;
3. не ухудшается качество при масштабировании.

Сферы применения векторной графики:
• схемы
• планы
• чертежи
• диаграммы

• шрифты

•ПРЕДСТАВЛЕНИЕ С ПОМОЩЬЮ ФРАКТАЛОВ

Фрактальная графика
• Фрактал — объект, отдельные элементы которого наследуют свойства родительских структур. Поскольку более детальное описание элементов меньшего масштаба происходит по простому алгоритму, описать такой объект можно всего лишь несколькими математическими уравнениями. • Фракталы позволяют описывать целые классы изображений, для детального описания которых требуется относительно мало памяти. С другой стороны, к изображениям вне этих классов, фракталы применимы слабо.



2.2 Классификация цветовых систем в представлении изображений

Как видим из вышеизложенного, описание цвета может опираться на составление любого цвета на основе основных цветов или на такие понятия, как светлота, насыщенность, цветовой тон. Применительно к компьютерной графике описание цвета также должно учитывать специфику аппаратуры для ввода/вывода изображений. В связи с необходимостью описания различных физических процессов воспроизведения цвета были разработаны различные цветовые модели. Цветовые модели позволяют с помощью математического аппарата описать определенные цветовые области спектра. Цветовые модели описывают цветовые оттенки с помощью смешивания нескольких основных цветов.

Основные цвета разбиваются на оттенки по яркости (от темного к светлому), и каждой градации яркости присваивается цифровое значение (например, самой темной – 0, самой светлой – 255). Считается, что в среднем человек способен воспринимать около 256 оттенков одного цвета. Таким образом, любой цвет можно разложить на оттенки основных цветов и обозначить его набором цифр – цветовых координат.

Таким образом, при выборе цветовой модели можно определять трехмерное цветовое координатное пространство, внутри которого каждый цвет представляется точкой. Такое пространство называется пространством цветовой модели.

Профессиональные графические программы обычно позволяют оперировать с несколькими цветовыми моделями, большинство из которых создано для специальных целей или особых типов красок: CMY, CMYK, CMYK256, RGB, HSB, HLS, L*a*b, YIQ, Grayscale (Оттенки серого) и Registration color. Некоторые из них используются редко, диапазоны других перекрываются.

Цветовая модель RGB.

В основе одной из наиболее распространенных цветовых моделей, называемой RGB моделью, лежит воспроизведение любого цвета путем сложения трех основных цветов: красного (Red), зеленого (Green) и синего (Blue). Каждый канал – R, G или B имеется свой отдельный параметр, указывающий на количество соответствующей компоненты в конечном цвете. Например: (255, 64, 23) – цвет, содержащий сильный красный компонент, немного зелёного и совсем немного синего. Естественно, что этот режим наиболее подходит для передачи богатства красок окружающей природы. Но он требует и больших расходов, так как глубина цвета тут наибольшая – 3 канала по 8 бит на каждый, что дает в общей сложности 24 бита.

Поскольку в RGB модели происходит сложение цветов, то она называется ***аддитивной*** (additive). Именно на такой модели построено воспроизведение цвета современными мониторами.

Цветовым пространством RGB модели является единичный куб.

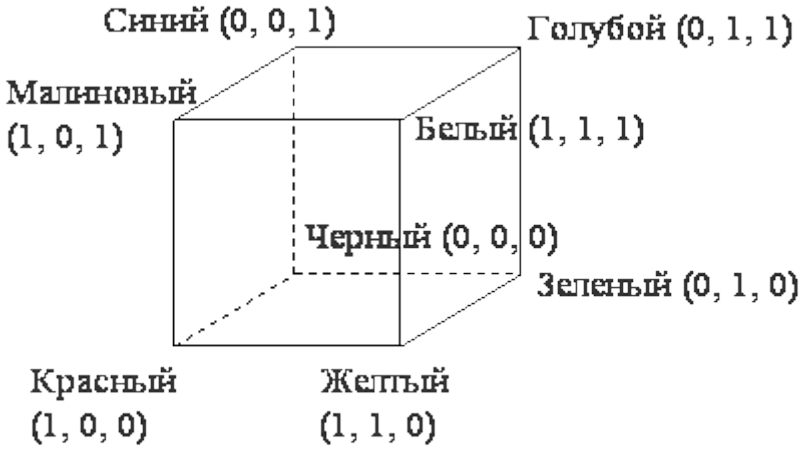


Рис. 1.7. Цветовое пространство RGB модели

Цветовые модели CMY и CMYK.

Модель CMY использует также три основных цвета: Cyan (голубой), Magenta (пурпурный, или малиновый) и Yellow (желтый). Эти цвета описывают отраженный от белой бумаги свет трех основных цветов RGB модели. Поэтому можно описать соотношения между RGB и CMY моделями следующим образом:

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} R \\ G \\ B \end{pmatrix} \qquad \begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} C \\ M \\ Y \end{pmatrix}.$$

Модель CMY является ***субтрактивной*** (основанной на вычитании) цветовой моделью. Как уже говорилось, в CMY-модели описываются цвета на белом носителе, т. Е. краситель, нанесенный на белую бумагу, вычитает часть спектра из падающего белого света. Например, на поверхность бумаги нанесли голубой (Cyan) краситель. Теперь красный свет, падающий на бумагу, полностью поглощается. Таким образом, голубой носитель вычитает красный свет из падающего белого.

Такая модель наиболее точно описывает цвета при выводе изображения на печать, т. Е. в полиграфии.

Поскольку для воспроизведения черного цвета требуется нанесение трех красителей, а расходные материалы дороги, использование CMY-модели является не эффективным. Дополнительный фактор, не добавляющий привлекательности CMY-модели, – это появление нежелательных визуальных эффектов, возникающих за счет того, что при выводе точки три базовые цвета могут ложиться с небольшими отклонениями. Поэтому к базовым трем цветам CMY-модели добавляют черный (blacK) и получают новую цветовую модель CMYK.

Для перехода из модели CMY в модель CMYK иногда используют следующее соотношение:

$K = \min(C, M, Y)$;
$C = C - K$;
$M = M - K$;
$Y = Y - K$.

Соотношения преобразования RGB в CMY и CMY в CMYK-модель верны лишь в том случае, когда спектральные кривые отражения для базовых цветов не пересекаются. Поэтому в общем случае можно сказать, что существуют цвета, описываемые в RGB-модели, но не описываемые в CMYK-модели.

Существует также модель CMYK256, которая используется для более точной передачи оттенков при качественной печати изображений.

Цветовые модели HSV и HLS.

Рассмотренные модели ориентированы на работу с цветопередающей аппаратурой и для некоторых людей неудобны. Поэтому модели HSV, HLS опираются на интуитивные понятия тона насыщенности и яркости.

В цветовом пространстве модели HSV (Hue, Saturation, Value), иногда называемой HSB (Hue, Saturation, Brightness), используется цилиндрическая система координат, а множество допустимых цветов представляет собой шестигранный конус, поставленный на вершину.

Основание конуса представляет яркие цвета и соответствует $V = 1$. Однако цвета основания $V = 1$ не имеют одинаковой воспринимаемой интенсивности. Тон (H) измеряется углом, отсчитываемым вокруг вертикальной оси OV. При этом красному цвету соответствует угол 0°, зелёному – угол 120° и т. Д. Цвета, взаимно дополняющие друг друга до белого, находятся напротив один другого, т. Е. их тона отличаются на 180°. Величина S изменяется от 0 на оси OV до 1 на гранях конуса.

Конус имеет единичную высоту ($V = 1$) и основание, расположенное в начале координат. В основании конуса величины H и S смысла не имеют. Белому цвету соответствует пара $S = 1$, $V = 1$. Ось OV ($S = 0$) соответствует ахроматическим цветам (серым тонам).

Процесс добавления белого цвета к заданному можно представить как уменьшение насыщенности S, а процесс добавления чёрного цвета – как уменьшение яркости V. Основанию шестигранного конуса соответствует проекция RGB куба вдоль его главной диагонали.

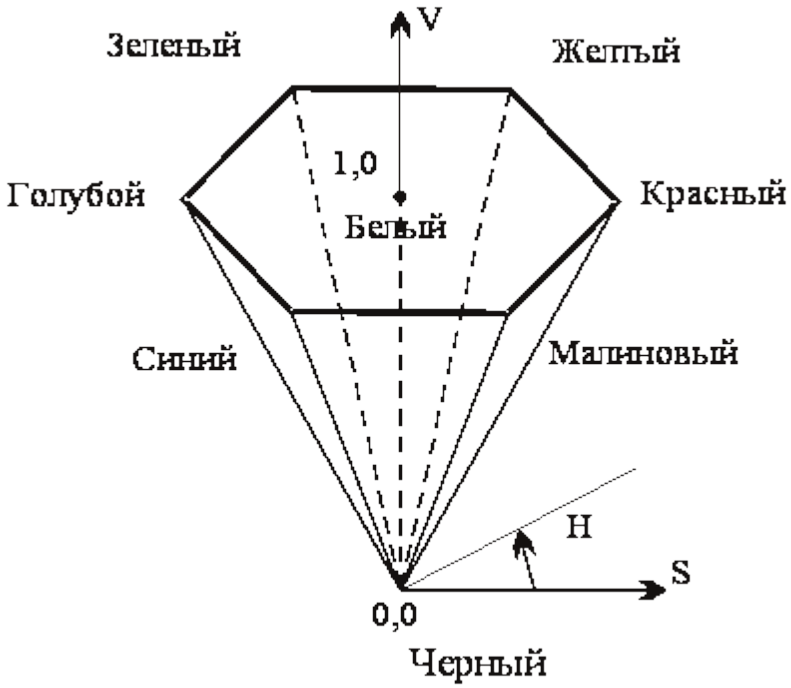


Рис. 1.8. Цветовое пространство HSV модели

Еще одним примером системы, построенной на интуитивных понятиях тона насыщенности и яркости, является система HLS (Hue, Lightness, Saturation). Здесь множество всех цветов представляет собой два шестигранных конуса, поставленных друг на друга (основание к основанию).

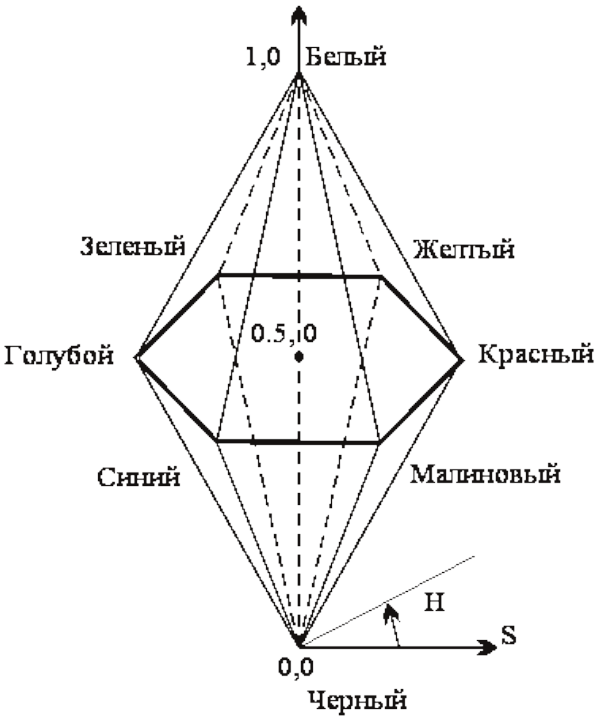


Рис. 1.9. Цветовое пространство HLS-модели

Полноцветные и индексированные изображения. Как мы увидели, цвета пикселей можно определять, явно задавая несколько параметров цвета. Например, в RGB-модели конечный цвет определяется тремя слагаемыми для трех основных цветов. Такой подход позволяет формировать так называемые полноцветные изображения.

Второй подход заключается в том, что в первой части файла, хранящего изображение, хранится «палитра», в которой с помощью одной из цветовых моделей кодируются цвета, присутствующие на изображении. А вторая часть, которая непосредственно описывает пикселы изображения, фактически состоит из индексов в палитре. Изображения, формируемые таким способом, называются изображениями с индексированной палитрой.

Частным случаем индексированного изображения является черно-белое изображение. В подобном изображении могут быть только 2 цвета – чёрный и белый, кодируемые соответственно 0 и 1. Глубина изображения составляет в данном случае 1 бит. Эта глубина очень плохо подходит к представлению фотореалистичных образов и применяется лишь для специализированных изображений.

Достоинством палитры является возможность существенно сократить размер файла с изображением. Недостатком является возможность потери цветов при ограниченном размере палитры. Обычно размер палитры составляет до 256-ти цветов.

2.3 Описание форматов bmp, jpeg, png, gif хранения растровых изображений

Изображения JPG / JPEG / JFIF

Аббревиатура от: Joint Photographic Experts Group – Объединённая группа экспертов по фотографиям.

Расширения файлов: .jpg / .jpeg

Самый распространенный формат среди растровых файлов изображений. Цифровые фотокамеры сохраняют свои изображения именно в этом формате. Файлы JPEG применяют метод сжатия с потерями, который может значительно уменьшить размер файла без существенного ущерба для качества.

Минусы: этот формат не защищен от деградации поколений. Это означает, что при каждом редактировании и сохранении качество изображения с каждой новой версией файла будет ухудшаться.

Применяемость: непрозрачные изображения, устройства захвата изображений (гаджеты, фотокамеры, экш-камеры).

Формат изображения PNG

Аббревиатура от: Portable Network Graphics – портативная сетевая графика.

Расширение файлов: .png

Эта бесплатная альтернатива GIF с открытым исходным кодом, которая поддерживает 16 миллионов цветов, в отличие от GIF, максимум которого 256-цветовая палитра. Это лучший формат файлов для изображений с сохранением источника цветов. Формат подходит для передачи идеального баланса тона. Анимированный файл PNG доступен в формате APNG. Эти файлы, как правило, имеют прозрачный фон.

Минусы: Больше всего подходит для файлов больших размеров. Сам формат PNG не поддерживает анимированную графику.

Применяемость: редактирование изображений, веб-изображения, изображения с учётом слоев, таких как прозрачность или эффекты затухания.

Расширение файла TIF

Аббревиатура от: Tagged Image File Format – формат для хранения растровых графических изображений.

Расширение файлов: .tif / .tiff

Гибкий и легко расширяемый формат файла, способный сохранять файлы с большой глубиной цвета. Эти файлы имеют прозрачный фон. Они идеально подходят для логотипов компании.

Минусы: Не идеальны для веб-браузеров.

Применяемость: Начальный этап фотографических файлов в печати. Программные пакеты OCR.

Формат файла GIF

Аббревиатура от: Graphics Interchange Format – формат для обмена изображениями.

Расширение файлов: .gif

Хотя GIF имеет низкую степень сжатия по сравнению с большинством видеоформатов, этот формат наиболее популярен среди пользователей для анимации изображений.

Минусы: формат ограничен 8-битной палитрой (256 цветов) и не подходит для фотографических изображений или сглаживания.

Применяемость: Графика, которая требует нескольких цветов, например упрощенные диаграммы, логотипы и анимации, которые состоят на более чем 50% из одного цвета.

Формат файла изображения BMP

Аббревиатура от: Bitmap Picture – дословно формат для хранения растровых изображений

Расширение файлов: .bmp

Этот формат разработан компанией Microsoft и предназначен для хранения больших несжатых файлов внутри ОС Windows.

Минусы: этот формат не использует сжатие.

Применяемость: упрощенная структура формата делает файлы bmp идеальными для программ Windows.

3. МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ И АЛГОРИТМЫ КОРРЕЛЯЦИОННОГО АНАЛИЗА ДВУМЕРНЫХ МАССИВОВ ДАННЫХ

3.1 Понятие о корреляционном анализе

Часто бывает необходимо определить степень независимости одного процесса от другого или установить сходство одного набора данных с другим. Другими словами, искомой является корреляция процессов или данных, которую можно определить математически и измерить.

Термин «корреляция» впервые применил французский палеонтолог Ж.Кювье, который вывел «закон корреляции частей и органов животных» (этот закон позволяет восстанавливать по найденным частям тела облик всего животного). В статистику указанный термин ввёл в 1886 году английский биолог и статистик Френсис Гальтон (не просто связь – relation, а «как бы связь» – co-relation). Однако точную формулу для подсчёта коэффициента корреляции разработал его ученик – математик и биолог – Карл Пирсон.

Корреляция – статистический метод, позволяющий определить, существует ли зависимость между переменными и насколько она сильна.

Корреляционный анализ – метод, позволяющий обнаружить зависимость между несколькими случайными величинами.

В основе корреляционного алгоритма лежит процедура сравнения коэффициента корреляции с пороговым значением. Величина коэффициента корреляции ниже порогового значения свидетельствует о динамике (изменениях), происходящей в технологическом процессе, в противном случае динамика (изменения) отсутствует.

Корреляционный алгоритм обработки изображений содержит ряд действий, которые можно условно разделить на подготовительную часть и основную часть, непосредственно расчёт и пороговая обработка результата, итогом которой становится вывод – есть изменения в развитии объекта на время наблюдения или нет.

Важные недостатки корреляционных методов обнаружения проявляются в присутствии радиометрических (яркостных) и особенно геометрических искажений текущего изображения по сравнению с эталонным. В частности, наблюдается быстрое уменьшение

корреляционной связи при так называемых ракурсных искажениях, например, при поворотах изображений.

Корреляцией называется статистическая взаимосвязь двух или нескольких случайных величин (либо величин, которые можно с некоторой допустимой степенью точности считать таковыми). При этом изменения значений одной или нескольких из этих величин сопутствуют систематическому изменению значений другой или других величин. Например, мы можем измерять рост и вес разных людей, и каждое измерение представлять точкой в двухмерном пространстве. Несмотря на то, что величины носят случайный характер, некоторая зависимость между ними будет наблюдаться, это пример положительной корреляции.Взаимосвязь между величинами необходимо охарактеризовать численно, например, для того чтобы различать два случая, показанных в верхней части рисунка 5.

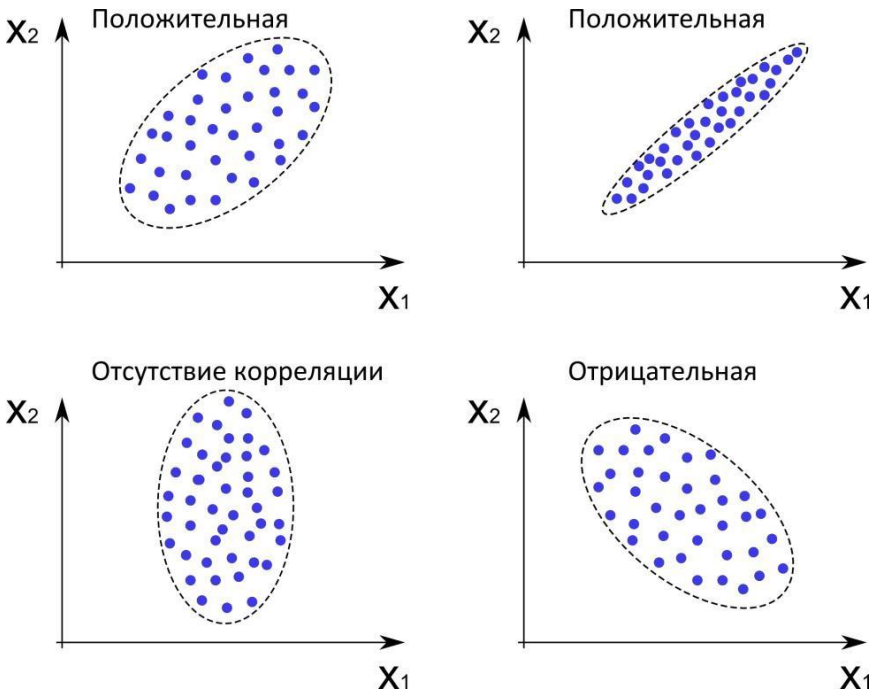


Рисунок 5 – Две величины, связанные разными статистическими закономерностями

Математической мерой корреляции двух случайных величин служит коэффициент корреляции. Для массива из n точек (x1i, y1i) он определяется следующим образом:

Рассчитываются средние значения для каждого из параметров

$$\overline{x_1} = \frac{\sum_{i=1}^{n} x_i}{n}; \qquad \overline{y_1} = \frac{\sum_{i=1}^{n} y_i}{n}. \tag{3.1}$$

И коэффициент корреляции составляет

$$r = \frac{\sum(x_{1i} - \overline{x_1})(y_{1i} - \overline{y_1})}{\sqrt{\sum(x_{1i} - \overline{x_1})^2}\sqrt{\sum(y_{1i} - \overline{y_1})^2}} \tag{3.2}$$

В обработке изображений чаще используют немного видоизменённое определение корреляции. Нетрудно увидеть, что если в выражении (3.2) рассматривать не сами случайные величины, а их отклонение от средних значений, то можно записать аналогичное выражение: (3.3)

$$r_{xy} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$

Взаимная корреляция двух изображений:

Методы корреляционного анализа применимы и к сигналам более высокой размерности, в том числе к двумерным, то есть к изображениям. Формулы в этом случае немного усложняются.

Теперь у нас есть два изображения, первое изображение z(x,y), а второе – t(i,j) – назовём шаблоном. Обычно шаблон должен быть меньше размером, чем исследуемое изображение, ширину шаблона обозначим через w, длину – l. Значение функции корреляции изображения с шаблоном в точке (x,y) исходного изображения определяется по формуле: (3.4)

$$R_{tz}(x, y) = \sum_{i=0}^{w-1}\sum_{j=0}^{l-1} t(i, j) * z\left(x + i - \frac{w}{2}, y + j - \frac{l}{2}\right).$$

При этом нормализованное выражение, подобное (3.3) будет выглядеть следующим образом (3.5)

$$\rho_{tz}(x,y) = \frac{R_{tz}(x,y)}{\sqrt{R_{zz(x,y)}R_{tt}(\frac{w}{2},\frac{l}{2})}},$$

где Rzz и Rtt – автокорреляционные функции изображения и шаблона.

Некоторые виды коэффициентов корреляции могут быть положительными или отрицательными. В первом случае предполагается, что мы можем определить только наличие или отсутствие связи, а во втором – также и её направление. Если предполагается, что на значениях переменных задано отношение строгого порядка, то отрицательная корреляция – корреляция, при которой увеличение одной переменной связано с уменьшением другой. При этом коэффициент корреляции в таких условиях – это такая связь, при которой увеличение одной переменной связано с увеличением другой переменной. Возможна также ситуация отсутствия статистической взаимосвязи – например, для независимости случайных величин. Положительная корреляция – корреляция, при которой увеличение одной переменной связано с увеличением другой переменной, при этом коэффициент корреляции может быть положительным.

Традиционная техника сравнения текущего изображения с эталоном основывается на рассмотрении изображений как двумерных функций яркости (дискретных двумерных матриц интенсивности). При этом измеряется либо расстояние между изображениями, либо мера их близости.

Корреляционная функция – функция времени или пространственных координат, которая задаёт корреляцию в системах со случайными процессами.

Зависящая от времени корреляция двух случайных функций X(t) и Y(t) определяется как $C(t,t') = \langle X(t)Y(t')\rangle$, где угловые скобки обозначают процедуру усреднения.

Если корреляционная функция вычисляется для одного и того же процесса, она называется автокорреляционной $C_{auto}(t,t') = \langle X)t,r)Y(t',r')\rangle$.

Так же можно вычислить корреляционную функцию для процессов, происходящих в разных точках пространства в различные моменты времени $C(t,r,t',r') = \langle X(t,r)Y(t',r')\rangle$.

Корреляционные функции широко используются в статистической

физике и других дисциплинах, изучающих случайные процессы.

Если в качестве двух случайных величин выступают выборки случайного процесса в два различных момента времени, то такое математическое ожидание зависит от того, насколько быстро эти функции изменяются во времени. Полагаем, что случайные величины будут сильно коррелированы, когда моменты времени очень близки друг к другу, поскольку случайная величина, зависящая от времени, за короткое время не может существенно измениться. В то же время корреляция между двумя выборочными значениями, взятыми в далеко отстоящие друг от друга моменты времени, скорее всего, весьма мала, так как за такое время случайные величины могут претерпевать практически любые изменения. Поскольку корреляция, безусловно, зависит от того, насколько быстро меняется во времени случайная величина, можно предположить, что она определяется также и тем, каким образом энергия случайного процесса распределяется по частотному спектру. Данное положение подтверждается тем, что у процесса, быстро изменяющегося во времени, высокочастотные составляющие обладают достаточной энергией, чтобы обеспечить его изменение.

Для случайных процессов, для фиксированных двух моментов времени t1 и t2 каждая пара случайных величин t1 и t2 может быть охарактеризована разделяющим их временным интервалом, при этом корреляция становится функцией этого интервала. Поэтому для данного случая подходит использование термина корреляционной функции, у которой аргументом является временной интервал между выборочными случайными величинами. Если эти случайные величины являются выборочными значениями одного и того же процесса, то указанная функция является автокорреляционной (корреляционной) функцией данного процесса, если же они принадлежат различным случайным процессам – взаимной корреляционной функцией.

Описание двумерного корреляционного анализа

Для выявления взаимосвязей между двумя переменными применяется двумерный корреляционный анализ. Значительная корреляция между двумя случайными величинами является свидетельством существования некоторой статистической связи в данной выборке, но та связь не обязательно должна наблюдаться для другой

выборки и иметь причинно- следственный характер. Часто простота корреляционного исследования подталкивает исследователя делать ложные интуитивные выводы о наличии причинно-следственной связи между парами признаков, в то время как коэффициенты корреляции устанавливают лишь статистические взаимосвязи. В то же время, отсутствие корреляции между двумя величинами ещё не значит, что меду ними нет никакой связи. Например, зависимость может иметь сложный нелинейный характер, который корреляция не выявляет.

Одним из основных показателей взаимозависимости двух случайных величин является парный коэффициент корреляции, служащий мерой линейной статистической зависимости между двумя величинами. Этот показатель соответствует своему прямому назначению, когда статистическая связь между соответствующими признаками в генеральной совокупности линейна. То же самое относится к частным и множественным коэффициентам корреляции. Одним из требований, определяющих корреляционный метод, является требование линейности статистической связи, т. Е. линейности всевозможных уравнений регрессии.

Парный коэффициент корреляции, характеризующий тесноту связи между случайными величинами X и Y в генеральной совокупности, определяется по формуле: (3.6)

$$\rho(x,y) = \frac{M[(X - M_X) \cdot (Y - M_{Y)}}{\sigma_X \cdot \sigma_Y},$$

где Мх и Му математические ожидания величин х и у; σх и σу – их среднеквадратические отклонения.

4. ОБЪЕКТНО-ОРИЕНТИРОВАННЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

По мере развития вычислительной техники создавались новые подходы, помогающие справляться с растущим усложнением программ. Использование структурного программирования при написании умеренно сложных программ принесло результаты, но оказалось несамостоятельным, когда программа достигала определённой длины. Чтобы писать более сложные программы, были разработаны принципы объектно-ориентированного программирования.

Объектно-ориентированное программирование (ООП) – методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования. Другими словами, ООП – технология программирования, при которой объект представляется не только описанием конкретных его свойств, но и функциями его поведения.

Идеологически ООП – подход к программированию как к моделированию информационных объектов, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов.

ООП – подход к разработке программного обеспечения, основанный на объектах, а не на процедурах, что позволяет максимизировать принципы модульности и «сокрытия информации». Объектно-ориентированное программирование базируется на связывании или инкапсуляции структур данных и процедуры, которая работает с данными в структуре, с модулем.

Объектно-ориентированное программирование позволяет разложить проблему на составные части. В этом случае вся процедура упрощается, и появляется возможность оперировать с гораздо более объёмными программами. Каждая составляющая становится самостоятельным объектом, содержащим свои собственные коды и данные, относящиеся к нему.

Все данные об объекте программирования и его связях с другими объектами можно объединить в одну структурированную переменную. В первом приближении её можно назвать объектом.

С объектом связывается набор действий, называемый методами. С точки зрения языка программирования набор действий или методов – это функции, получающие в качестве обязательного параметра указатель на объект и выполняющие определённые действия с данными объекта программирования. Технология ООП запрещает работать с объектом иначе, чем через методы, таким образом, внутренняя структура объекта скрыта от внешнего пользователя.

Объект обладает набором заранее определённых встроенных методов обработки, либо созданных пользователем, либо заимствованным

в стандартных библиотеках, которые выполняются при наступлении заранее определённых событий (вход в поле ввода, нажатие определённых клавиш).

Объект – это структурированная переменная, содержащая всю информацию о некотором физическом предмете или реализуемом в программе понятии.

Класс – это описание множества объектов программирования (объектов) и выполняемых над ними действий.

Класс можно сравнить с чертежом, согласно которому создаются объекты. Часто классы разрабатывают так, чтобы их объекты соответствовали объектам предметной области решаемой задачи. Любая функция в программе представляет собой метод для объекта некоторого класса.

Класс должен формироваться в программе естественным образом, как только в ней возникает необходимость описания новых объектов программирования. Каждый новый шаг в разработке алгоритма должен представлять собой разработку нового класса на основе уже существующих.

Свойство – характеристика объекта, его параметр. Все объекты наделены определёнными свойствами, которые в совокупности выделяют объект из множества других объектов.

Объект обладает качественной определённостью, что позволяет выделить его из множества других объектов и обуславливает независимость создания и обработки от других объектов.

Свойства объектов различных классов могут пересекаться, то есть возможны объекты, обладающие одинаковыми свойствами.

Событие – изменение состояния объекта.

Внешние события генерируются пользователем (например, клавиатурный ввод или нажатие кнопки мыши, выбор пункта меню, запуск макроса); внутренние события генерируются системой.

Поле – элемент данных класса: переменная элементарного типа, структура или другой класс, являющийся частью класса.

Модификатор доступа – дополнительная характеристика членов класса, определяющая, имеется ли к ним доступ из внешней программы, или же они используются исключительно в границах класса и скрыты от окружающего мира. Модификаторы доступа разделяют все элементы класса на детали реализации и открытый или частично открытый

интерфейс.

Конструктор – специальный метод, выполняемый сразу же после создания экземпляра класса. Конструктор инициализирует поля объекта – приводит объект в начальное состояние. Конструкторы могут быть как с параметрами, так и без. Конструктор без параметров называют конструктором по умолчанию, который может быть только один. Имя метода конструктора, чаще всего, совпадает с именем самого класса.

Деструктор – специальный метод, вызываемый средой исполнения программы в момент, когда объект удаляется из оперативной памяти. Деструктор используется в тех случаях, когда в состав класса входят ресурсы, требующие явного освобождения (файлы, соединения с базами данных, сетевые соединения и т.п.)

Интерфейс – набор методов и свойств объекта, находящихся в открытом доступе и призванных решать определённый круг задач, к примеру, интерфейс формирования графического представления объекта на экране или интерфейс сохранения состояния объекта в файле или базе данных.

Статический член – любой элемент класса, который может быть использован без создания соответствующего объекта. К примеру, если метод класса не использует ни одного поля, а работает исключительно с переданными ему параметрами, то ничто не мешает его использовать в контексте всего класса, не создавая отдельных его экземпляров. Константы в контексте класса обычно всегда являются статическими его членами.

Наиболее очевидные недостатки ООП:

– ООП порождает огромные иерархии классов, что приводит к тому, что функциональность расползается или размывается по базовым и производным членам класса, и отследить логику работы того или иного метода становится сложно.

– В некоторых языках все данные являются объектами, в том числе и элементарные типы, а это не может не приводить к дополнительным расходам памяти и процессорного времени.

– Также, на скорости выполнения программ может неблагоприятно сказаться реализация полиморфизма, которая основана на механизмах позднего связывания вызова метода с конкретной его реализацией в одном из производных классов.

Основные принципы программирования

Основы программирования на языках ООП заключаются в использовании объектов и классов. При переходе от процедурного стиля написания исходного кода к объектно-ориентированному нередко возникают сложности, однако большинство разработчиков находят в ООП множество плюсов [7].

Абстракция – это придание объекту характеристик, которые отличают его от всех других объектов, чётко определяя его концептуальные границы. Основная идея состоит в том, чтобы отделить способ использования составных объектов данных от деталей их реализации в виде более простых объектов, подобно тому, как функциональная абстракция разделяет способ использования функции и деталей её реализации в терминах более примитивных функций, таким образом, данные обрабатываются функцией высокого уровня с помощью вызова функций низкого уровня. Осуществляется разделение интерфейса и внутренней реализации. Такой подход является основой ООП.

У объектно-программного программирования есть свои постулаты. Принципы ООП – это его основные идеи. Объектно-ориентированное программирование опирается на три базовых принципа: инкапсуляцию, наследование и полиморфизм.

Объектно-ориентированное программирование постоянно развивается, порождая новые парадигмы, такие как аспектно-ориентированное, субъектно-ориентированное и даже агентно-ориентиванное программирование.

a) Инкапсуляция (пакетирование)

Инкапсуляция – свойство языка программирования, позволяющее пользователю не задумываться о сложности реализации используемого программного компонента, а взаимодействовать с ним посредством предоставляемого интерфейса (публичных методов и членов), а также объединить и защитить жизненно важные для компонента данные. При этом пользователю предоставляется только спецификация (интерфейс) объекта. Инкапсуляция – объединение в пакет разнотипных данных и функций. Пользователь не может использовать закрытые данные и методы. Реализуется с помощью ключевых слов: private, protected, internal.

b) Наследование (заимствование)

Наследование – один из четырёх важнейших механизмов объектно-ориентированного программирования, позволяющий описать новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом. Другими словами, класс-наследник реализует спецификацию уже существующего класса (базовый класс). Это позволяет обращаться с объектами класса-наследника точно так же, как с объектами базового класса. Наследование бывает простым и множественным.

с) Полиморфизм

Полиморфизм – возможность объектов с одинаковой спецификацией иметь различную реализацию. Язык программирования поддерживает полиморфизм, если классы с одинаковой спецификацией могут иметь различную реализацию – например, реализация класса может быть изменена в процессе наследования. Кратко смысл полиморфизма можно выразить фразой: «Один интерфейс, множество реализаций». Полиморфизм позволяет писать более абстрактные программы и повысить коэффициент повторного использования кода. Общие свойства объектов объединяются в систему, которую могут называть по-разному – интерфейс, класс.

5. ПРОЕКТИРОВАНИЕ ГРАФИЧЕСКОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА СРЕДСТВАМИ COM-ОБЪЕКТОВ

Графический пользовательский интерфейс – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. П.), представленные пользователю на дисплее, исполнены в виде графических изображений.

Component Object Model (COM) – объектная модель компонентов – это технологический стандарт от компании Microsoft, предназначенный для создания программного обеспечения на основе взаимодействующих компонентов, каждый из которых может использоваться во многих программах одновременно. Стандарт воплощает в себе идеи полиморфизма и инкапсуляции ООП. Стандарт COM мог бы быть универсальным и платформо-независимым, но закрепился в основном на операционных системах семейства Microsoft Windows. В современных

версиях Windows COM используется очень широко.

5.1 Основы технологии COM

Технология COM применяется при описании API (Application Programming Interface) – это интерфейс программирования, интерфейс создания приложений) и двоичного стандарта для связи объектов различных языков и сред программирования. COM предоставляет модель взаимодействия между компонентами и приложениями [9].

Технология COM работает с COM-объектами. COM-объекты похожи на обычные объекты визуальной библиотеки компонентов Delphi. В отличие от объектов VCL Delphi, COM-объекты содержат свойства, методы и интерфейсы. Обычный COM-объект включает в себя один или несколько интерфейсов. Каждый из этих интерфейсов имеет собственный указатель.

Технология COM имеет два явных плюса:

– создание COM-объектов не зависит от языка программирования. Таким образом, COM-объекты могут быть написаны на различных языках;

– COM-объекты могут быть использованы в любой среде программирования под Windows. В число этих сред входят Delphi, Visual C++, C++Builder, Visual Basic и многие другие.

Все COM-объекты содержатся в файлах с расширением DLL или OCX. Один такой файл может содержать как один, так и несколько COM-объектов.

Ключевой аспект технологии COM – возможность предоставления связи и взаимодействия между компонентами и приложениями, а также реализация клиент-серверных взаимодействий при помощи интерфейсов.

Технология COM реализуется с помощью COM-библиотек (в число которых входят такие файлы операционной системы, как OLE32.DLL и OLE-Aut32.DLL). COM-библиотеки содержат набор стандартных интерфейсов, которые обеспечивают функциональность COM-объекта, а также небольшой набор функций API, отвечающих за создание и управление COM-объектов.

Основное понятие, которым оперирует COM, – COM-компонент. Программы, построенные на стандарте COM, не являются автономными программами, а представляют собой набор взаимодействующих между собой COM-компонентов. Каждый компонент имеет уникальный идентификатор (GUID) и может одновременно использоваться многими

программами. Компонент взаимодействует с другими программами через COM-интерфейсы. Каждый COM-компонент должен поддерживать стандартный интерфейс «Iunknown», предоставляющий базовые средства для работы с компонентом. Он включает в себя три метода: QueryInterface, AddRef, Release.

Windows API предоставляет базовые функции, позволяющие использовать COM-компоненты. Библиотеки MFC и, особенно, ATL/WTL предоставляют более гибкие и удобные средства для работы с COM. Библиотека ATL от Microsoft до сих пор остаётся самым популярным средством создания COM-компонентов. Но зачастую COM-разработка остаётся ещё довольно сложным делом, программистам приходится вручную выполнять многие рутинные задачи, связанные с COM (особенно это заметно в случае разработки на C++). Впоследствии (в технологиях COM+ и особенно .NET) Microsoft попыталась упростить задачу разработки COM-компонентов.

5.2 Терминология COM

COM-объект представляет собой двоичный код, который выполняет какую-либо функцию и имеет один или более интерфейс. COM-объект содержит методы, которые позволяют приложению пользоваться COM-объектом. Эти методы доступны благодаря COM-интерфейсам. Клиенту достаточно знать несколько базовых интерфейсов COM-объекта, чтобы получить полную информацию о составе свойств и методов объекта. COM-объект может содержать один или несколько интерфейсов. Для программиста COM-объект работает так же, как и класс в Object Pascal.

Пользователь COM-объекта – приложение или часть приложения, которое использует COM-объект и его интерфейсы в своих собственных целях. Как правило, COM-объект находится в другом приложении.

COM-сервер – приложение или библиотека, предоставляющая услуги приложению-клиенту или библиотеке. Он содержит один или более COM-объектов, выступающих в качестве наборов свойств, методов и интерфейсов.

Клиенты не знают, как действует COM-объект, предоставляющий свои услуги при помощи интерфейсов. В дополнение, приложению-клиенту не нужно знать, где находится COM-объект. Технология COM обеспечивает прозрачный доступ независимо от местонахождения COM-объекта.

Очень важно при разработке COM-приложений создание приложений, называемых COM-клиентами, которые могут запрашивать интерфейсы объектов, чтобы определить те услуги, которые может предоставить COM-объект. Типичным COM-клиентом является диспетчер автоматизации (Automation Controller). Диспетчер автоматизации – это часть приложения, которая знает какой тип информации необходим ему от разных объектов сервера, и она запрашивает данную информацию по мере надобности.

Клиенты COM связаны с объектами COM-интерфейсами. После того как **интерфейс** COM **опубликова**н (стандартным способом зарегистрирован системой), изменять его нельзя, что гарантирует одинаковую работу объекта COM в любых условиях.

COM-интерфейс применяется для объединения методов COM-объекта. Интерфейс позволяет клиенту правильно обратиться к COM-объекту, а объекту – правильно ответить клиенту. COM-интерфейс – набор абстрактных функций и свойств, через который программы взаимодействует с COM-компонентом. Клиент может не знать, какие интерфейсы имеются у COM-объекта. Для того чтобы получить их список, клиент использует базовый интерфейс Iunknown, который есть у каждого COM-объекта. Каждый COM-объект всегда поддерживает основной COM-интерфейс IUnknown, который применяется для передачи клиенту сведений о поддерживаемых интерфейсах.

Указатель интерфейса – это 32-битный указатель на экземпляр объекта, который является, в свою очередь, указателем на реализацию каждого метода интерфейса. Реализация методов доступна через массив указателей на эти методы, который называется vtable. Использование массива vtable похоже на механизм поддержки виртуальных функций в Object Pascal.

5.3 Технологии, основанные на стандарте COM

DCOM

Выпущенная в 1996 году технология DCOM основана на технологии DCE/RPC (разновидности RPC). DCOM позволяет COM-компонентам взаимодействовать друг с другом по сети. Главным конкурентом DCOM является другая известная распределённая технология – CORBA.

DCOM и CORBA вызывают метод объекта, расположенного на

другой машине и передают ссылки на объект с одной машины на другую.

Сетевой уровень DCOM называется ORPC (Object RPC) и является объектно-ориентированным расширением DCE RPC. Технология DCOM обеспечивает базовые установки безопасности, позволяя задавать, кто и из каких машин может создавать экземпляры объекта и вызывать его методы.

COM+

Microsoft Transaction Server был включен в Option Pack для Windows NT4 ещё в 1997 году. В составе Windows 2000 была выпущена технология COM+, которая являлась новой версией Microsoft Transaction Server.

Технология расширяла возможности разработчиков COM-компонентов, предоставляя им некоторые готовые услуги, например:

– автоматический пул потоков, создаваемый стандартным процессом-загрузчиком mtx.exe;

– доступ к контексту, в котором выполняется компонент (например, компоненты, используемые в ASP, могут с этой возможностью получить доступ к внутренним объектам той страницы, на которой они выполняются);

– интеграция с транзакциями монитора MS DTC (контекст COM+ может автоматически содержать в себе транзакцию MS DTC).

MTS/COM+ использовался внутри ряда версий веб-сервера MS IIS для загрузки и исполнения веб-приложений, как бинарных по технологии ISAPI, так и скриптовых по технологии ASP (сама asp.dll есть ISAPI-приложение).

COM+ объединяет компоненты в так называемые приложения COM+, что упрощает администрирование и обслуживание компонентов. Безопасность и производительность – основные направления усовершенствований COM+. Некоторые идеи, заложенные в основу COM+, были также реализованы в Microsoft .NET.

.NET и будущее COM

В 2002 году была официально выпущена платформа Microsoft .NET, которая на сегодняшний день объявлена Microsoft рекомендуемой основой для создания приложений и компонентов под Windows. По этой причине в .NET включены и средства, позволяющие обращаться к компонентам COM из приложений .NET, и наоборот. По словам представителей

Майкрософт, COM (точнее, COM+) и .NET – отлично взаимодополняющие технологии.

DCOM через интернет и решение проблемы XP SP2

В 2009 году DComLab опубликовал коммерческий продукт ComBridge. При использовании ComBridge для работы по DCOM через интернет не требуется CIS, в локальной сети не требуются настройки dcomcnfg. ComBridge встраивается в транспортный уровень DCOM, полностью выделяя весь трафик созданного объекта и всех полученных из него объектов в отдельный поток.

OPC

OPC (OLE for Process Control) – семейство программных технологий, предоставляющих единый интерфейс для управления объектами автоматизации и технологическими процессами. Многие из OPC-протоколов базируются на Windows-технологиях: OLE, ActiveX, COM/DCOM. Такие OPC-протоколы, как OPC XML DA и OPC UA, являются платформо-независимыми.

OLE

OLE (Object Linking and Embedding – связывание и встраивание объекта) – технология связывания и внедрения объектов в другие документы и объекты, разработанные корпорацией Майкрософт.

OLE позволяет передавать часть работы от одной программы редактирования к другой и возвращать результаты назад. Например, установленная на персональном компьютере издательская система может послать некий текст на обработку в текстовый редактор, либо некоторое изображение в редактор изображений с помощью OLE-технологии.

6. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРОЦЕССОВ ЦИФРОВОЙ ОБРАБОТКИ ИЗОБРАЖЕНИЙ

Разбиения изображения на пересекающиеся фрагменты:

```
private void button4_Click(object sender, EventArgs e)
        {
            if (pictureBox1.Image != null) //если в pictureBox есть изображение
            {
                //создание диалогового окна "Сохранить как..", для сохранения
изображения
```

```
                SaveFileDialog savedialog = new SaveFileDialog();
                savedialog.Title = "Сохранить картинку как...";
                //отображать ли предупреждение, если пользователь указывает имя уже
существующего файла
                savedialog.OverwritePrompt = true;
                //отображать ли предупреждение, если пользователь указывает
несуществующий путь
                savedialog.CheckPathExists = true;
                //список форматов файла, отображаемый в поле "Тип файла"
                savedialog.Filter = "Image Files(*.PNG)|*.PNG|All files (*.*)|*.*";
                //отображается ли кнопка "Справка" в диалоговом окне
                savedialog.ShowHelp = true;
                if (savedialog.ShowDialog() == DialogResult.OK) //если в диалоговом
окне нажата кнопка "OK"
                {
                    try

                    {
                        int k = 0;

                        int Ix, Iy;
                        Ix = int.Parse(textBox1.Text) + 1;
                        Iy = int.Parse(textBox2.Text) + 1;
                        Forms = new Bitmap[Ix * Iy];

                        int stepX = bm.Size.Width / Ix;
                        int stepY = bm.Size.Height / Iy;
                        Bitmap tempImg = new Bitmap(stepX, stepY);
                        Bitmap catBM = (Bitmap)pictureBox1.Image;
                        int pos = savedialog.FileName.Length - 5;
                        int posX = -stepX, posY = -stepY;

                        for (int i = 0; i < Iy; i++)
                        {
                            posY += stepY;
                            posX = -stepX;
                            for (int j = 0; j < Ix; j++)
                            {
                                posX += stepX;

        tempImg = catBM.Clone(new Rectangle(posX, posY, stepX, stepY), bm.PixelFormat);
                                Forms[k] = tempImg;
        savedialog.FileName = savedialog.FileName.Insert(pos, k.ToString());
        tempImg.Save(savedialog.FileName, System.Drawing.Imaging.ImageFormat.Png);
                                k++;
                            }
                        }
                    }
                    catch
                    {
                        MessageBox.Show("Невозможно сохранить изображение", "Ошибка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }

        }
        //fragments
```

Их поворотов:

```
private void trackBar1_Scroll(object sender, EventArgs e)
        {
            angle = trackBar1.Value;
            label6.Text = angle.ToString();
        }
```

И

```
public void button3_Click(object sender, EventArgs e)
        {
            Image img = new Bitmap(pictureBox1.Image.Width, pictureBox1.Image.Height);
            using (Graphics g = Graphics.FromImage(img))
            {
                g.TranslateTransform(img.Width / 2, img.Height / 2);
                g.RotateTransform(angle); // поворот на 45 градусов
                g.TranslateTransform(-(img.Width / 2), -(img.Height / 2));
                g.DrawImage(bm, 0, 0);
            }
            pictureBox1.Image = img;
        }
```

Объединения в единое целое:

```
private void button7_Click(object sender, EventArgs e)
        {
            Image tempBit = new Bitmap(bm.Size.Width, bm.Size.Height);
            Graphics g = Graphics.FromImage(tempBit);
            int kol = fragments.Length;
            int posX = 0, posY = 0;
            int Ix = int.Parse(textBox1.Text) + 1;
            int Iy = int.Parse(textBox2.Text) + 1;
            int stepX = bm.Size.Width / Ix;
            int stepY = bm.Size.Height / Iy;
            int count = 0;
            for (int i = 0; i < kol; i++)
            {
                if(count==Ix)
                {
                    count = 0;
                    posX = 0;
                    posY += stepY;

                }
                for (int j = 0; j < kol; j++)
                {
                    if (BitmapsEqual(Forms[i], fragments[j]))
                    {
                        g.DrawImage(fragments[j], posX, posY);
                        break;
                    }
                }
                posX += stepX;
                count++;


            }
            pictureBox3.Image = tempBit;
```

7. АНАЛИЗ РЕЗУЛЬТАТОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

В результате курсовой работы была решена задача восстановления изображения по его фрагментам путем деления изображения на эти фрагменты, а затем сравнивая данные нам фрагменты способом Unsafe кода, который позволяет обратиться к памяти определенной ячейки и сравнить её. Разработан графический пользовательский интерфейс, а также способы восстановления изображений. Были рассмотрены элементы управления, такие как Picture_box, кнопка и текстовое поле.

Были реализованы такие возможности как: наложение сетки на изображение, выбор размерности сетки, загрузка и выгрузка из папки и в папку изображений и их фрагментов соответственно. Так же была реализована работа с различными расширениями графических файлов.

ЗАКЛЮЧЕНИЕ

Сегодня восстановление изображения широко применяется в различных областях инженерной конструкторской деятельности, медицине, веб-разработке, научной, деловой, конструкторской и иллюстративных сферах, поэтому очень важно понять принцип работы с изображениями, а также научиться применять технологию разбиения изображения на фрагменты и восстановления этого изображения в повседневной жизни.

Мною были изучены различные математические модели и операторы повороты и масштабирования для реализации в программе.

СПИСОК ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

[1]. В.Т. Фисенко, Т.Ю. Фисенко, Компьютерная обработка и распознавание изображений: учеб. пособие. - СПб: СПбГУ ИТМО, 2008. – 192 с

[2]. Косников Ю.Н. Геометрические преобразования в компьютерной графике. Конспект лекций. – Пенза: Пензенский государственный университет, 2011. – 50 с.

[3]. Основы информатики: Учеб. пособие /И.Я. Львович, Ю.П. Преображенский, В.В. Ермолова. ВИВТ, Воронеж, 2012, 236 с.

ПРИЛОЖЕНИЕ А (обязательное)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Drawing.Imaging;

namespace Shipai_Kursach
{
    public partial class Form1 : Form
    {
        Bitmap bm;
        Bitmap bmBack;
        Bitmap[] fragments;
        Bitmap[] Forms;
        Graphics g;
        float angle = 0f;
        float scale = 1f;
        int kolvo = 0;

        public Form1()
        {
            InitializeComponent();
            Size = new System.Drawing.Size(1000, 500);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog fl = new OpenFileDialog();
            fl.Filter = "Image
Files(*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG|All files (*.*)|*.*";
            if (fl.ShowDialog() == DialogResult.OK)
            {
                try
                {
                    bm = new Bitmap(fl.FileName);
                    bmBack = new Bitmap(fl.FileName);
                    pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
                    pictureBox2.SizeMode = PictureBoxSizeMode.StretchImage;
                    pictureBox3.SizeMode = PictureBoxSizeMode.StretchImage;
                    pictureBox1.Image = new Bitmap(bm);

                    g = pictureBox1.CreateGraphics();



                }
                catch
                {
                    MessageBox.Show("Bad file", "Ошибка", MessageBoxButtons.OK);
                }
            }
        }
```

```
        private void Draw_Click(object sender, EventArgs e)
        {



            int Gx, Gy;

            if (textBox1.Text == "" && textBox2.Text == "")
            {
                MessageBox.Show("X:?" +
                    "Y:?", "Ошибка", MessageBoxButtons.OK);
                return;
            }
            Gx = int.Parse(textBox1.Text);
            Gy = int.Parse(textBox2.Text);







            Pen p = new Pen(Color.Red, 1);
            int distance = 0, step = 0;
            distance = pictureBox1.ClientSize.Width / (Gx + 1);

            for (int i = 0; i < Gx; i++)
            {
                step += distance;
                g.DrawLine(p, new Point(step, 0), new Point(step,
pictureBox1.ClientSize.Height));

            }
            step = 0;
            distance = pictureBox1.ClientSize.Height / (Gy + 1);

            for (int i = 0; i < Gy; i++)
            {
                step += distance;
                g.DrawLine(p, new Point(0, step), new
Point(pictureBox1.ClientSize.Width, step));

            }

        }

        private void button2_Click(object sender, EventArgs e)
        {
            pictureBox1.Image = bmBack;
            trackBar1.Value = 0;
            angle = 0f;
            label6.Text = "0";
        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {

            angle = trackBar1.Value;
            label6.Text = angle.ToString();
```

```
        }

        public void button3_Click(object sender, EventArgs e)
        {
            Image img = new Bitmap(pictureBox1.Image.Width, pictureBox1.Image.Height);
            using (Graphics g = Graphics.FromImage(img))
            {
                g.TranslateTransform(img.Width / 2, img.Height / 2);
                g.RotateTransform(angle); // поворот на 45 градусов
                g.TranslateTransform(-(img.Width / 2), -(img.Height / 2));
                g.DrawImage(bm, 0, 0);
            }
            pictureBox1.Image = img;


        }

        private void button4_Click(object sender, EventArgs e)
        {
            if (pictureBox1.Image != null) //если в pictureBox есть изображение
            {
                //создание диалогового окна "Сохранить как..", для сохранения
изображения
                SaveFileDialog savedialog = new SaveFileDialog();
                savedialog.Title = "Сохранить картинку как...";
                //отображать ли предупреждение, если пользователь указывает имя уже
существующего файла
                savedialog.OverwritePrompt = true;
                //отображать ли предупреждение, если пользователь указывает
несуществующий путь
                savedialog.CheckPathExists = true;
                //список форматов файла, отображаемый в поле "Тип файла"
                savedialog.Filter = "Image Files(*.PNG)|*.PNG|All files (*.*)|*.*";
                //отображается ли кнопка "Справка" в диалоговом окне
                savedialog.ShowHelp = true;
                if (savedialog.ShowDialog() == DialogResult.OK) //если в диалоговом
окне нажата кнопка "OK"
                {
                    try

                    {
                        int k = 0;


                        int Ix, Iy;
                        Ix = int.Parse(textBox1.Text) + 1;
                        Iy = int.Parse(textBox2.Text) + 1;
                        Forms = new Bitmap[Ix * Iy];

                        int stepX = bm.Size.Width / Ix;
                        int stepY = bm.Size.Height / Iy;
                        Bitmap tempImg = new Bitmap(stepX, stepY);
                        Bitmap catBM = (Bitmap)pictureBox1.Image;
                        int pos = savedialog.FileName.Length - 5;
                        int posX = -stepX, posY = -stepY;

                        for (int i = 0; i < Iy; i++)
                        {
                            posY += stepY;
                            posX = -stepX;
                            for (int j = 0; j < Ix; j++)
                            {
```

```
                                posX += stepX;

                                tempImg = catBM.Clone(new Rectangle(posX, posY, stepX,
stepY), bm.PixelFormat);
                                Forms[k] = tempImg;
                                savedialog.FileName = savedialog.FileName.Insert(pos,
k.ToString());
                                tempImg.Save(savedialog.FileName,
System.Drawing.Imaging.ImageFormat.Png);
                                k++;
                            }
                        }
                    }
                    catch
                    {
                        MessageBox.Show("Невозможно сохранить изображение", "Ошибка",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                    }
                }
            }



        }
        //fragments
        private void button5_Click(object sender, EventArgs e)
        {
            OpenFileDialog dlg = new OpenFileDialog
            {
                Multiselect = true,
                Title = "Выберите файлы",
                InitialDirectory = @"C:\"
            };
            dlg.ShowDialog();
            fragments = new Bitmap[dlg.SafeFileNames.Length];

            // пользователь вышел из диалога ничего не выбрав
            if (dlg.FileName == String.Empty)
                return;
            int count = 0;
            foreach (string file in dlg.FileNames)
            {
                fragments[count] = new Bitmap(file);
                count++;
            }
            pictureBox2.Image = fragments[0];

        }

        private void button6_Click(object sender, EventArgs e)
        {
            kolvo++;
            if (kolvo == fragments.Length)
                kolvo = 0;
            pictureBox2.Image = fragments[kolvo];

        }
        unsafe public static Boolean BitmapsEqual(  Bitmap bmp1, Bitmap bmp2)
        {
            if (bmp1.Size != bmp2.Size)
                return false;
```

```
            BitmapData bmpd1 = bmp1.LockBits(new Rectangle(0, 0, bmp1.Width,
bmp1.Height), ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
            BitmapData bmpd2 = bmp2.LockBits(new Rectangle(0, 0, bmp2.Width,
bmp2.Height), ImageLockMode.ReadOnly, PixelFormat.Format32bppArgb);
            Boolean res = true;

                Int32* p1 = (Int32*)bmpd1.Scan0;
                Int32* p2 = (Int32*)bmpd2.Scan0;
                for (Int32 i = 0; i < bmpd1.Height; i++)
                {
                    for (Int32 j = 0; j < bmpd2.Width; j++)
                    {
                        if (*p1 != *p2)
                        {
                            res = false;
                            break;
                        }
                        p1++;
                        p2++;
                    }
                }

            bmp1.UnlockBits(bmpd1);
            bmp2.UnlockBits(bmpd2);
            return res;
        }
        private void button7_Click(object sender, EventArgs e)
        {
            Image tempBit = new Bitmap(bm.Size.Width, bm.Size.Height);
            Graphics g = Graphics.FromImage(tempBit);
            int kol = fragments.Length;
            int posX = 0, posY = 0;
            int Ix = int.Parse(textBox1.Text) + 1;
            int Iy = int.Parse(textBox2.Text) + 1;
            int stepX = bm.Size.Width / Ix;
            int stepY = bm.Size.Height / Iy;
            int count = 0;
            for (int i = 0; i < kol; i++)
            {
                if(count==Ix)
                {
                    count = 0;
                    posX = 0;
                    posY += stepY;

                }
                for (int j = 0; j < kol; j++)
                {
                    if (BitmapsEqual(Forms[i], fragments[j]))
                    {
                        g.DrawImage(fragments[j], posX, posY);
                        break;
                    }
                }
                posX += stepX;
                count++;


            }
            pictureBox3.Image = tempBit;
        }
```

```
    }
}
```