Нижегородский государственный университет им. Н.И. Лобачевского

Разработка мультимедийных приложений с использованием библиотек OpenCV и IPP

Лабораторная работа Intel® Integrated Performance Primitives. Использование библиотеки в среде Microsoft® Visual Studio

При поддержке компании Intel

Кустикова В.Д.,
кафедра математического обеспечения ЭВМ

Содержание

- Цели и задачи работы
- Установка Intel® Integrated Performance Primitives в составе пакета Intel® Parallel Studio XE 2013
- Подготовка среды Microsoft® Visual Studio для разработки приложений с использованием Intel® Integrated Performance Primitives
- Разработка приложения для медианной фильтрации с использованием библиотеки Intel® Integrated Performance Primitives
- Разработка приложения для поиска прямых линий на изображении с использованием Intel® Integrated Performance Primitives

Библиотека Intel® IPP (1)

- Библиотека высокопроизводительных инструментов и программных функций обработки данных. Функции библиотеки можно разделить на четыре основные группы:
- ***Функции обработки звуковых сигналов:***
  - ДПФ, некоторые функции распознавания и кодирования речи;
  - функции сжатия данных и др.
- ***Функции обработки изображений и видео:***
  - функции конвертирования изображений из одного цветового пространства в другое;
  - морфологические операции;
  - реализации алгоритмов компьютерного зрения и др.

Библиотека Intel® IPP (2)

- ***Функции работы с небольшими с векторами и матрицами небольших размеров:***
  - операции над векторами (сложение, вычитание, умножение и т.п.);
  - операции с матрицами (транспонирование, вычисление определителя, умножение, определение следа матрицы и т.п.);
  - функции решения систем линейных алгебраических уравнений и др.
- ***Функции шифрования.***

Цели работы

- Рассмотреть технические этапы подготовки инфраструктуры.

- Продемонстрировать использование некоторых функций библиотеки Intel® Integrated Performance Primitives на примере простых задач медианной фильтрации и поиска прямых линий на изображении.

Задачи работы

- Установка библиотеки Intel® IPP в составе пакета Intel® Parallel Studio XE 2013.
- Настройка среды Microsoft Visual Studio с целью использования библиотеки при разработке C/C++ приложений.
- Разработка приложения, которое осуществляет медианную фильтрацию изображения средствами библиотеки Intel® IPP.
- Разработка приложения для поиска прямых линий на изображении средствами библиотеки Intel® IPP, содержащей функции вычисления преобразования Хафа.

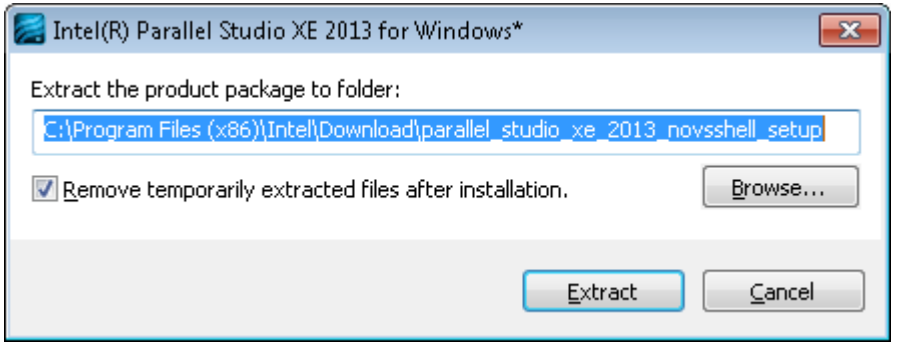
Тестовая инфраструктура

Операционная система	Microsoft Windows 7
Среда разработки	Microsoft Visual Studio 2010
Библиотеки Intel® IPP	Intel® Integrated Performance Primitives 7.1 (в составе Intel® Parallel Studio XE 2013)
Библиотеки OpenCV	Версия 2.4.3

УСТАНОВКА IPP В СОСТАВЕ ПАКЕТА IPS XE 2013

Установка IPP в составе пакета IPS XE 2013 (1)

- Загрузить установочный файл с официальной страницы компании Intel.
- Запустить установочный файл и следовать инструкциям мастера установки:
  - Выберите путь для извлечения временных файлов.



  - Распакуйте временные файлы, нажав на кнопку «**Extract**».

Установка IPP в составе пакета IPS XE 2013 (2)

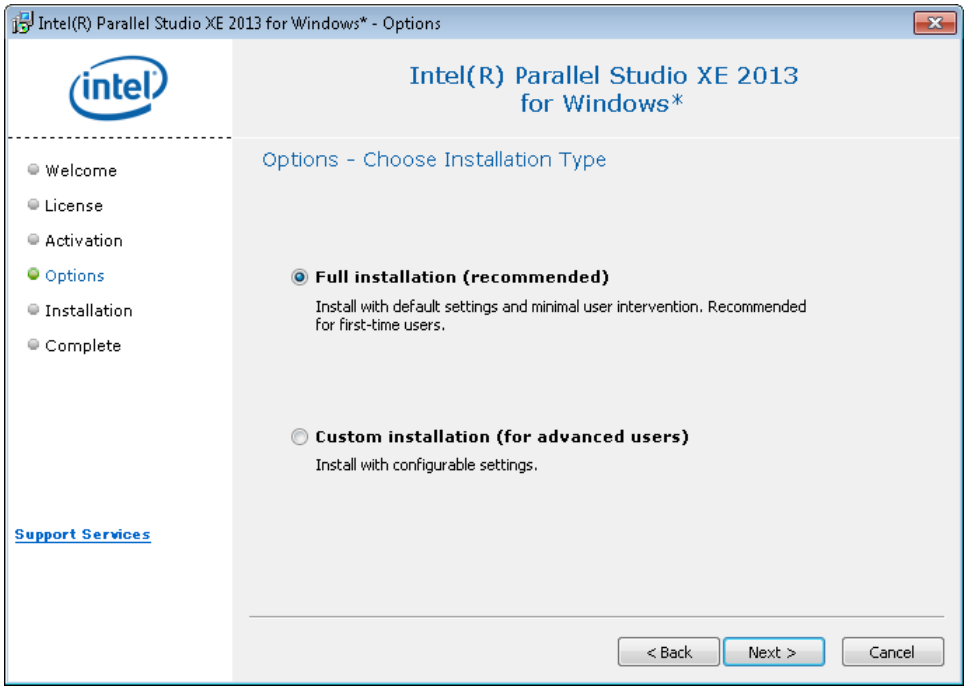
– После извлечения временных файлов будет выполнен переход в основное окно установки.

Установка IPP в составе пакета IPS XE 2013 (3)

– Выберите тип лицензии и укажите необходимые параметры активации пакета, нажмите кнопку «**Next**».

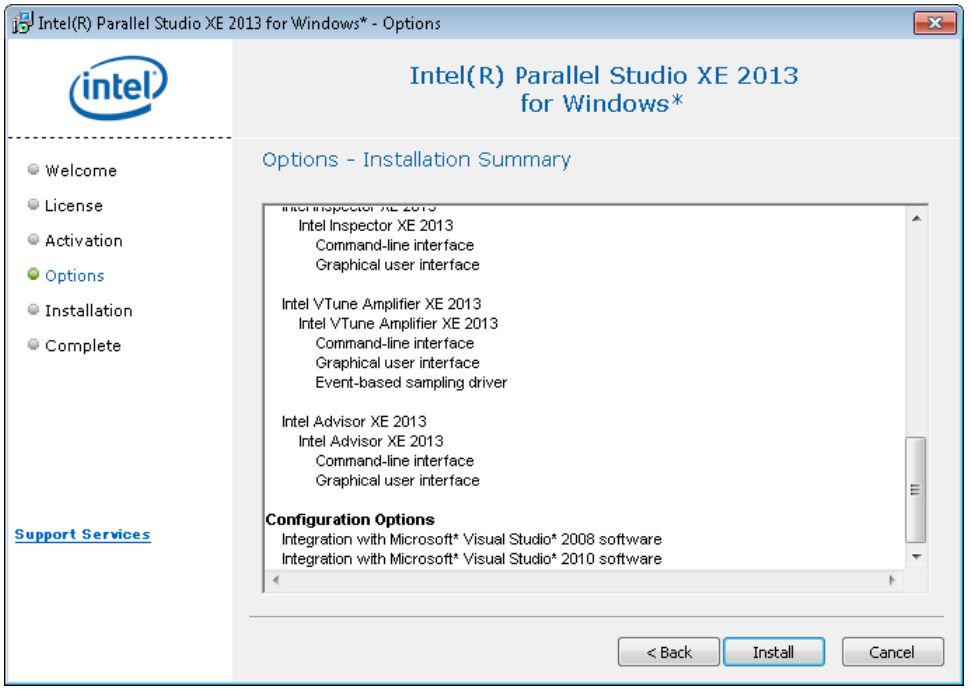
Установка IPP в составе пакета IPS XE 2013 (4)

– Выберите тип установки (полная или выборочная) и нажмите кнопку «**Next**», чтобы перейти к следующему этапу установки.



Установка IPP в составе пакета IPS XE 2013 (5)

– Проверьте наличие опции автоматической интеграции в Visual Studio и инициируйте процесс инсталляции посредством нажатия на кнопку «**Install**».



ПОДГОТОВКА СРЕДЫ VS 2010 ДЛЯ РАБОТЫ С БИБЛИОТЕКОЙ INTEL® IPP

Создание проекта (1)

- Запустите приложение Microsoft Visual Studio 2010.
- В меню **File** выполните команду **New→Project…**.
- В диалоговом окне **New Project** в типах проекта выберите **Win32**, в шаблонах **Win32 Console Application**, в поле **Name** введите название проекта (для каждого приложения будет использовано свое название), в поле **Solution Name** – название решения **LW_InstallIPP**, в поле **Location** укажите путь к папке с лабораторными работами. Нажмите OK.
- В диалоговом окне **Win32 Application Wizard** нажмите **Next** (или выберите **Application Settings** в дереве слева) и установите флаг **Empty Project**. Нажмите **Finish**.

Создание проекта (2)

- В окне **Solution Explorer** в папке **Source Files** выполните команду контекстного меню **Add→New Item…**. В дереве категорий слева выберите **Code**, в шаблонах справа – **C++ File (.cpp)**, в поле **Name** введите имя файла **main**. Нажмите **Add**. В результате выполненной последовательности действий в окне редактора кода Visual Studio будет открыт пустой файл **main.cpp**.
- Создайте заготовку функции **main()** с параметрами командной строки:

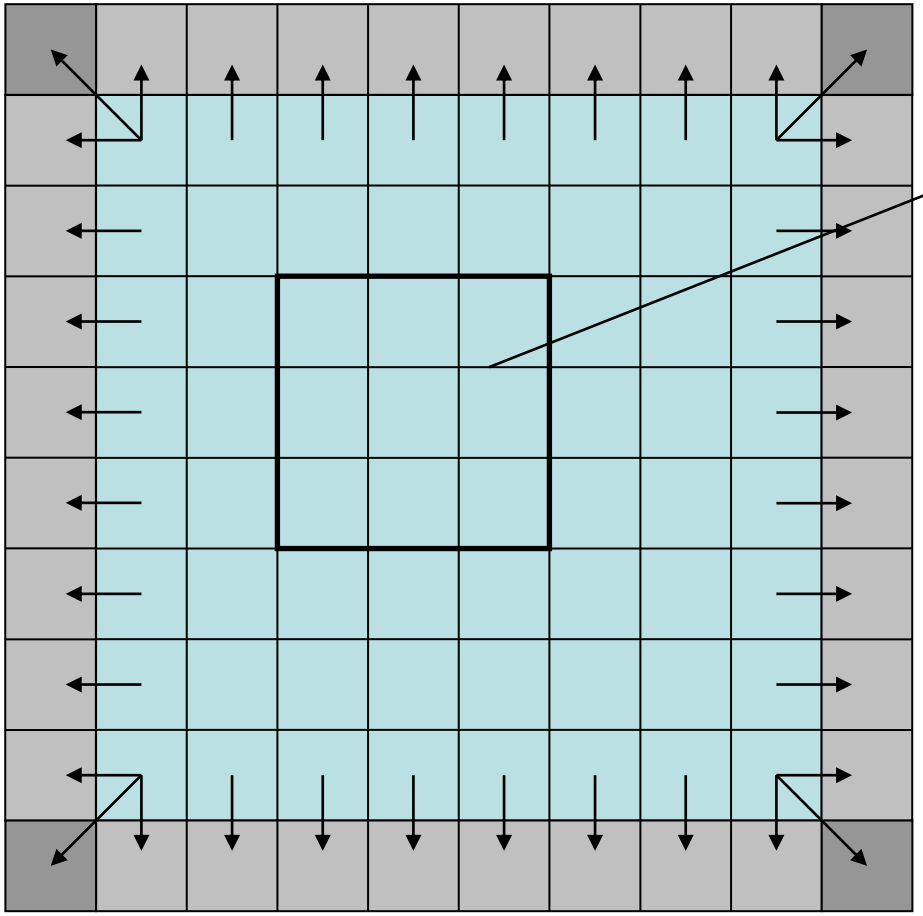
```
int main(int argc, char *argv[])
{
    // TODO: исходный код приложения
    return 0;
}
```

Настройка свойств проекта

- Выполните команду контекстного меню **Properties**, чтобы получить доступ к настройкам проекта.
- Откройте вкладку **Configuration Properties→Intel Performance Libraries→Use IPP**.
- В выпадающем списке напротив свойства **Use IPP** выберите способ линковки библиотеки Intel® IPP.

РАЗРАБОТКА ПРИЛОЖЕНИЯ МЕДИАННОЙ ФИЛЬТРАЦИИ НА БАЗЕ INTEL® IPP

Задача медианной фильтрации



14	7	47
12	56	14
80	14	30

7 12 14 14 **14** 30 47 56 80

Алгоритм медианной фильтрации

- Алгоритм реализации медианного фильтра предполагает проход по всем точкам изображения:
  - Выбор окрестности точки – шаблона.
  - Сортировка значений интенсивности цветов окрестных точек (по каждому каналу).
  - Поиск значения, расположенного в середине упорядоченного массива, – новое значение цвета текущего пиксела.
  - Сдвиг шаблона – переход к следующей точке.

Требования к программной реализации

- Поддержка медианной фильтрации с использованием библиотеки OpenCV.
- Поддержка медианной фильтрации средствами библиотеки Intel® IPP.
- Проверка корректности результата медианной фильтрации, полученного с помощью реализации на базе Intel® IPP, посредством сравнения с результатом, выданным соответствующей функцией библиотеки OpenCV.

- Условимся, что на входе приложения имеется цветное изображение в формате RGB, размер шаблона медианного фильтра равен 5.

Общая структура приложения

- Приложение состоит из основной функции и модуля **median_filtering**.
- Модуль содержит 2 функции-обертки, которые реализуют медианную фильтрацию средствами библиотек Intel® IPP и OpenCV соответственно.

Структура основной функции

- Загрузка исходного изображения с использованием функции **imread** библиотеки OpenCV.
- Вызов функции медианной фильтрации **median_opencv**, реализованной в модуле **median_filtering** на базе OpenCV.
- Вызов функции **median_ipp**, также реализованной в модуле **median_filtering** на базе библиотеки Intel® IPP.
- Попиксельное сравнение результатов медианной фильтрации с помощью функции **compare** (в модуле **median_filtering**). Функция возвращает количество пикселей с различающимися цветами.
- Отображение отфильтрованных изображений.
- Освобождение ресурсов.

Реализация с использованием функций библиотеки OpenCV

```
int median_opencv(const Mat &srcImg, Mat &dstImg,
                  const int kSize)
{
    medianBlur(srcImg, dstImg, kSize);
    return 0;
}
```

Реализация с использованием функций Intel® IPP (1)

```
int median_ipp(const Mat &srcImg, Mat &dstImg,
               const int msk)
{
 // размер границы для дублирования
 const int borderSize = msk / 2;
 // размер шаблона фильтра
 IppiSize mskSize = { msk    , msk      };
 // ведущая позиция в шаблоне
 IppiPoint anchor = { msk / 2, msk / 2 };
 Mat ippSrcImg, ippDstImg;
 Size ippImgSize;
 IppiSize ippSrcSize, ippDstSize;
 Ipp8u *pSrcData, *pDstData;

 // продолжение на следующем слайде…
```

Реализация с использованием функций Intel® IPP (2)

```
// размер изображения по горизонтали и по вертикали
// с дополнительной границей
ippImgSize.width  = srcImg.size().width  +
                    2 * borderSize;
ippImgSize.height = srcImg.size().height +
                    2 * borderSize;
// создание изображений с дополнительной границей
ippSrcImg.create(ippImgSize, srcImg.type());
ippDstImg.create(ippImgSize, srcImg.type());
// преобразование указателей на данные
pSrcData = (Ipp8u *)ippSrcImg.data;
pDstData = (Ipp8u *)ippDstImg.data;


// продолжение на следующем слайде…
```

Реализация с использованием функций Intel® IPP (3)

```
ippSrcSize.width  = srcImg.size().width;
ippSrcSize.height = srcImg.size().height;
ippDstSize.width  = srcImg.size().width  +
                    2 * borderSize;
ippDstSize.height = srcImg.size().height +
                    2 * borderSize;
// дублирование граничных пикселей
ippiCopyReplicateBorder_8u_C3R(srcImg.data,
       srcImg.step1(), ippSrcSize, pSrcData,
       srcImg.step1() + 3 * 2 * borderSize,
       ippDstSize, borderSize, borderSize);


// продолжение на следующем слайде…
```

Реализация с использованием функций Intel® IPP (4)

```
// медианная фильтрация
ippiFilterMedian_8u_C3R(
    pSrcData + ippImgSize.width * 3 * borderSize +
               3 * borderSize,
    srcImg.step1() + 3 * 2 * borderSize,
    pDstData + ippImgSize.width * 3 * borderSize +
               3 * borderSize,
    srcImg.step1() + 3 * 2 * borderSize,
    ippSrcSize, mskSize, anchor);

// продолжение на следующем слайде…
```

Реализация с использованием функций Intel® IPP (5)

```
    dstImg.create(srcImg.size(), srcImg.type());
    pSrcData = pDstData;
    pDstData = (Ipp8u *)dstImg.data;

    ippiCopy_8u_C3R(
         pSrcData + ippImgSize.width * 3 * borderSize +
                    3 * borderSize,
         srcImg.step1() + 3 * 2 * borderSize, pDstData,
         dstImg.step1(), ippSrcSize);

    // освободить память
    ippSrcImg.release();
    ippDstImg.release();
    return 0;
}
```

Проверка корректности

- Для проверки корректности реализации, разработанной на базе Intel® IPP, необходимо реализовать функцию сравнения цветов пикселей отфильтрованных изображений посредством Intel® IPP и OpenCV.

- Необходимо выполнить проход по всем пикселям и сравнить значения по каждому из трех каналов.

- В результате формируется количество пикселей, в которых не совпали значения интенсивности хотя бы по одному каналу.

Запуск приложения


```
OpenCV and IPP give the same result.
```

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ПОИСКА ПРЯМЫХ ЛИНИЙ НА БАЗЕ INTEL® IPP

Задача поиска прямых линий

- Определить параметры задания геометрических примитивов (в настоящей работе прямых линий).

Преобразование Хафа (1)

- *Преобразование Хафа* – метод обнаружения прямых и более сложных кривых на бинарных изображениях.
- Метод позволяет определить параметры семейства кривых и обеспечивает поиск на изображении множества кривых указанного семейства.

- Прямую на плоскости можно задать параметрически

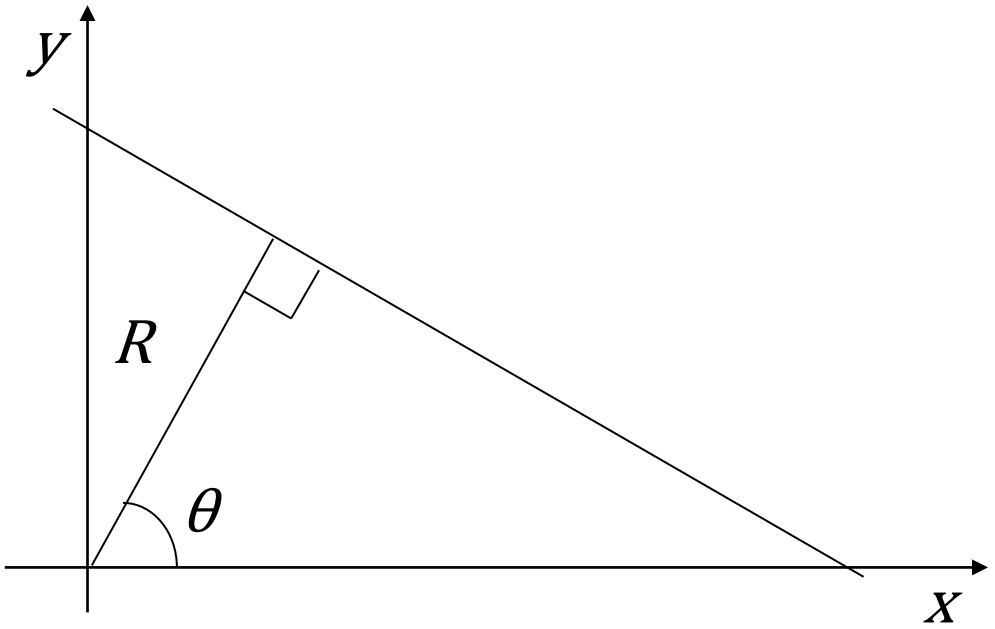
$$x \cos\theta + y \sin\theta = R,$$

где R – величина перпендикуляра, опущенного из начала координат системы Oxy на прямую, θ – угол наклона перпендикуляра относительно оси Ox, $0 \leq \theta < 2\pi$.

Преобразование Хафа (2)

- Тогда неявная функция, определяющая семейство прямых, может быть записана в следующем виде:

$$F(R, \theta, x, y) = x \cos\theta + y \sin\theta - R = 0$$



Преобразование Хафа (3)

- Через каждую точку (x, y) проходит несколько прямых, соответствующих разным значениям параметров R и θ.
- Любой фиксированной точке (x_0, y_0) ставится в соответствие набор точек в пространстве параметров (R, θ).

- В свою очередь каждой точке (R_0, θ_0) можно поставить в соответствие количество точек (x, y), лежащих на прямой $x \cos \theta_0 + y \sin \theta_0 - R_0 = 0$.

Преобразование Хафа (4)

- Дискретность представления данных требует введения сетки на плоскости параметров $\theta_i = i\Delta\theta,\ R_j = j\Delta R$.
- Каждой точке (R_0, θ_0) будем ставить в соответствие количество точек, принадлежащих семейству прямых:

$$x\cos\theta + y\sin\theta = R, \qquad \theta_i \le \theta \le \theta_{i+1}, R_i \le R \le R_{i+1}$$

- Выбрав ячейку сетки, которой соответствует максимальное количество точек, получим наиболее вероятную прямую. За параметры прямой можно принять координаты центра ячейки сетки.
- ***Замечание:*** если необходимо выделить несколько прямых, то достаточно отсортировать множество ячеек сетки в порядки убывания числа точек и выбрать определенное количество первых ячеек.

Требования к программной реализации

- Поддержка возможности поиска прямых линий с использованием преобразования Хафа, реализованного в библиотеке OpenCV.
- Поддержка поиска прямых линий с использованием преобразования Хафа и других необходимых функций библиотеки Intel® IPP.
- Отображение полученных прямых линий на исходном изображении.

Схема решения задачи поиска прямых линий с использованием преобразования Хафа

- Загрузка изображения.
- Преобразование изображения в оттенки серого.
- Поиск ребер на полученном изображении. Ребра можно найти, например, с использованием детектора Канни. Различные способы выделения ребер были рассмотрены в лабораторной работе «Базовые операции обработки изображений».
- Применение преобразования Хафа к бинарному изображению, содержащему ребра.
- Преобразование полученных параметров прямых из полярной системы координат в декартову систему, связанную с исходным изображением.

Общая структура приложения

- Приложение состоит из основной функции и модуля **hough_transform**.
- Модуль содержит две функции, которые реализуют поиск прямых линий средствами библиотек Intel® IPP и OpenCV соответственно, а также функцию отображения результирующего набора линий на изображении.

Структура основной функции

- Загрузка исходного изображения с использованием функции **imread** библиотеки OpenCV.
- Вызов функции поиска прямых линий **hough_opencv**, реализованной в модуле **hough_transform**, которая выполняет поиск средствами OpenCV.
- Вызов функции **hough_ipp**, также реализованной в модуле **hough_transform**. Функция выполняет определение прямых линий с использованием преобразования Хафа на базе Intel® IPP.
- Отображение исходных изображений с отрисованным набором прямых линий.
- Освобождение ресурсов, использованных для работы с изображениями.

Реализация поиска прямых с помощью функций библиотеки OpenCV (1)

```
int hough_opencv(const Mat &srcImg,
      vector<Point> &points1, vector<Point> &points2)
{
 Mat edges, grayImg;
 vector<Vec2f> lines;
 double threshold1 = 50, threshold2 = 200,
        rhoStep = 1, thetaStep = CV_PI / 180;
 int houghThreshold = 100, kLines, i;
 // преобразовать в оттенки серого
 cvtColor(srcImg, grayImg, CV_RGB2GRAY);
 // выполнить поиск ребер
 Canny(grayImg, edges, threshold1, threshold2);
 // отобразить результирующие ребра
 namedWindow("Canny (OpenCV)");
 imshow("Canny (OpenCV)", edges);
 // продолжение на следующем слайде…
```

Реализация поиска прямых с помощью функций библиотеки OpenCV (2)

```
// применить преобразование Хафа
HoughLines(edges, lines, rhoStep, thetaStep,
           houghThreshold);
kLines = lines.size();
for (i = 0; i < kLines; i++) {
 float rho = lines[i][0], theta = lines[i][1];
 Point pt1, pt2;
 double cosTheta = cos(theta), sinTheta = sin(theta);
 double x0 = cosTheta * rho, y0 = sinTheta * rho;
 pt1.x = cvRound(x0 + 1000 * (-sinTheta));
 pt1.y = cvRound(y0 + 1000 * cosTheta);
 pt2.x = cvRound(x0 - 1000 * (-sinTheta));
 pt2.y = cvRound(y0 - 1000 * cosTheta);
 points1.push_back(pt1); points2.push_back(pt2);
}
// продолжение на следующем слайде…
```

Реализация поиска прямых с помощью функций библиотеки OpenCV (3)

```
    // освободить ресурсы
    grayImg.release();
    edges.release();
    return 0;
}
```

Реализация поиска прямых с помощью функций библиотеки Intel® IPP (1)

```
int hough_ipp(const Mat &srcImg,
              vector<Point> &points1,
              vector<Point> &points2)
{
 Ipp32f low = 50.0f, high = 100.0f;
 int minNumPoints = 35, maxLineCount = 40,
     lineCount, bufSize;
 Ipp8u* pGraySrc, *pBinSrc, *buffer;
 IppiSize pGraySize;
 IppStatus error;


 // продолжение на следующем слайде…
```

Реализация поиска прямых с помощью функций библиотеки Intel® IPP (2)

```
// создать полутоновое изображение
Mat grayImg(srcImg.size(), CV_8UC1);
// заполнить размер IppiSize для изображения
// в оттенках серого
pGraySize.width = srcImg.size().width;
pGraySize.height = srcImg.size().height;
// преобразовать изображение в оттенки серого
error = ippiRGBToGray_8u_C3C1R(srcImg.data,
            srcImg.step1(), grayImg.data,
            grayImg.step1(), pGraySize);

// продолжение на следующем слайде…
```

Реализация поиска прямых с помощью функций библиотеки Intel® IPP (3)

```
pGraySrc = (Ipp8u *)grayImg.data;
// создать бинарное изображение
Mat binImg(srcImg.size(), CV_8UC1);
int vertSize, horzSize, vertSobelStep, horzSobelStep;
IppiSize vertSobelSize, horzSobelSize;
vertSobelSize.width = srcImg.size().width;
vertSobelSize.height = srcImg.size().height;
horzSobelSize.width = srcImg.size().width;
horzSobelSize.height = srcImg.size().height;
Ipp16s *horzSobel, *vertSobel;
ippiFilterSobelVertGetBufferSize_8u16s_C1R(
     vertSobelSize, ippMskSize3x3, &vertSize);
ippiFilterSobelHorizGetBufferSize_8u16s_C1R(
     horzSobelSize, ippMskSize3x3, &horzSize);
if (vertSize < horzSize) vertSize = horzSize;
// продолжение на следующем слайде…
```

Реализация поиска прямых с помощью функций библиотеки Intel® IPP (4)

```
ippiCannyGetSize(pGraySize, &vertSize);
horzSobel = ippiMalloc_16s_C1(srcImg.size().width,
                              srcImg.size().height,
                              &horzSobelStep);
vertSobel = ippiMalloc_16s_C1(srcImg.size().width,
                              srcImg.size().height,
                              &vertSobelStep);
buffer = ippsMalloc_8u(vertSize);
ippiFilterSobelVertBorder_8u16s_C1R(pGraySrc,
          grayImg.step1(), vertSobel, vertSobelStep,
          vertSobelSize, ippMskSize3x3,
          ippBorderRepl, 0, buffer);
ippiFilterSobelHorizBorder_8u16s_C1R(pGraySrc,
          grayImg.step1(), horzSobel, horzSobelStep,
          horzSobelSize, ippMskSize3x3,
```

Реализация поиска прямых с помощью функций библиотеки Intel® IPP (5)

```
// определить ребра с помощью детектора Канни
error = ippiCanny_16s8u_C1R(horzSobel,
            horzSobelStep,
            vertSobel, vertSobelStep, binImg.data,
            binImg.step1(), pGraySize, low, high,
            buffer);
// освободить память из-под вспомогательных буферов
ippsFree(buffer);
ippiFree(horzSobel);
ippiFree(vertSobel);
namedWindow("Canny (IPP)");
imshow("Canny (IPP)", binImg);

// продолжение на следующем слайде…
```

Реализация поиска прямых с помощью функций библиотеки Intel® IPP (6)

```
pBinSrc = (Ipp8u *)binImg.data;
IppPointPolar delta;
delta.rho = 1;
delta.theta = 1.0f * ((float)CV_PI) / 180.0f;
IppPointPolar *lines;
Ipp8u *pBuffer;
// вычислить размер вспомогательного буфера
ippiHoughLineGetSize_8u_C1R(pGraySize, delta,
                  maxLineCount, &bufSize);
pBuffer = ippsMalloc_8u(bufSize);
lines = (IppPointPolar *)malloc(
                sizeof(IppPointPolar) * maxLineCount);
ippiHoughLine_8u32f_C1R(pBinSrc, binImg.step1(),
          pGraySize, delta, minNumPoints, lines,
          maxLineCount, &lineCount, pBuffer);
ippsFree(pBuffer);
```

Реализация поиска прямых с помощью функций библиотеки Intel® IPP (7)

```
for (int lineIdx = 0; lineIdx < lineCount; lineIdx++)
{
 float rho = lines[lineIdx].rho,
       theta = lines[lineIdx].theta;
 Point pt1, pt2;
 double cosTheta = cos(theta), sinTheta = sin(theta);
 double x0 = rho * cosTheta, y0 = rho * sinTheta;
 pt1.x = cvRound(x0 + 1000*(-sinTheta));
 pt1.y = cvRound(y0 + 1000*(cosTheta));
 pt2.x = cvRound(x0 - 1000*(-sinTheta));
 pt2.y = cvRound(y0 - 1000*(cosTheta));
 points1.push_back(pt1);
 points2.push_back(pt2);
}
binImg.release(); free(lines);
return 0;
}
```

Реализация поиска прямых с помощью функций библиотеки Intel® IPP (8)

```
// определить ребра с помощью детектора Канни
error = ippiCanny_16s8u_C1R(horzSobel, horzSobelStep,
                vertSobel, vertSobelStep, binImg.data,
                binImg.step1(), pGraySize, low, high,
                buffer);

// освободить память из-под вспомогательных буферов
ippsFree(buffer);
ippiFree(horzSobel);
ippiFree(vertSobel);
// отобразить ребра на бинарном изображении
namedWindow("Canny (IPP)");
imshow("Canny (IPP)", binImg);

// продолжение на следующем слайде…
```

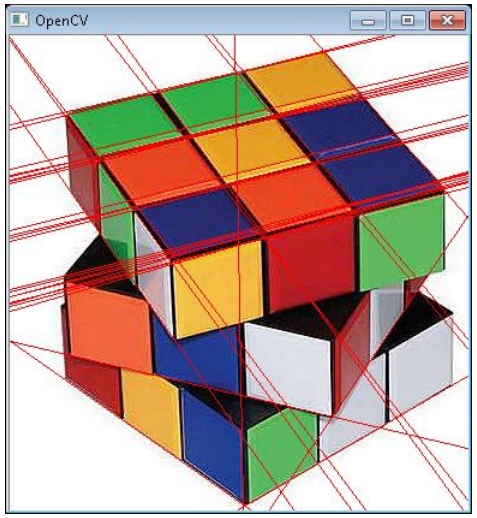
Реализация поиска прямых с помощью функций библиотеки Intel® IPP (9)

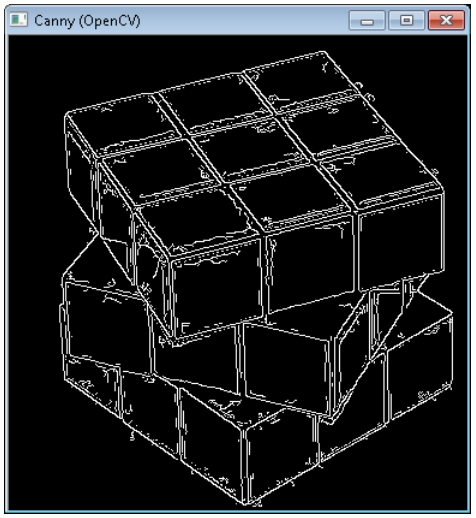
```
pBinSrc = (Ipp8u *)binImg.data;
// установить параметры для преобразования Хафа
IppPointPolar delta, *lines;
delta.rho = 1;
delta.theta = 1.0f * ((float)CV_PI) / 180.0f;
Ipp8u *pBuffer;
// вычислить размер вспомогательного буфера
error = ippiHoughLineGetSize_8u_C1R(pGraySize, delta,
                maxLineCount, &bufSize);
// выделить память под вспомогательный буфер
// и массив линий
pBuffer = ippsMalloc_8u(bufSize);
lines = (IppPointPolar *)malloc(sizeof(IppPointPolar)
                                * maxLineCount);

// продолжение на следующем слайде…
```

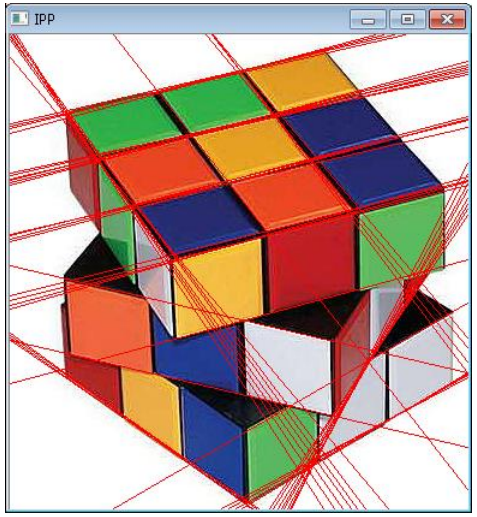
Реализация поиска прямых с помощью функций библиотеки Intel® IPP (10)

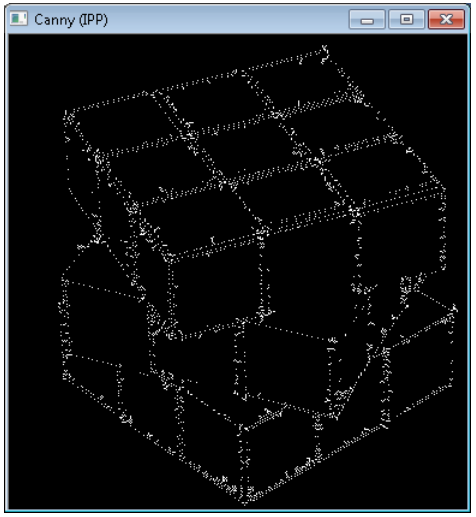
```
    // выполнить преобразование Хафа
    error = ippiHoughLine_8u32f_C1R(pBinSrc,
                    binImg.step1(), pGraySize, delta,
                    minNumPoints, lines, maxLineCount,
                    &lineCount, pBuffer);
    ippsFree(pBuffer);

    // TODO: преобразовать  координаты линий из полярной
    // системы координат в декартову систему по аналогии
    // с реализацией на базе OpenCV

    // освободить ресурсы
    alignedImg.release();
    binImg.release();
    free(lines);
    return 0;
}
```

Запуск приложения и анализ результатов









Задания для самостоятельной работы (1)

- Разработайте параллельную реализацию функции медианной фильтрации, которая использует возможности библиотеки Intel® Integrated Performance Primitives. Указание: установите соответствующие настройки проекта и предусмотрите возможность задания количества потоков. Выполните анализ масштабируемости разработанной параллельной реализации.

Задания для самостоятельной работы (2)

- Подумайте, какие операции можно применить к исходному изображению или изображению, содержащему контуры объектов, чтобы получить более качественный результат поиска прямых линий с использованием преобразования Хафа. Внесите необходимые изменения в исходный код приложения и проанализируйте результат.
- Попробуйте применить другие способы выделения контуров на изображении вместо детектора Канни в разработанном приложении поиска прямых линий с использованием преобразования Хафа.

Авторский коллектив

- Кустикова Валентина Дмитриевна,
  ассистент кафедры
  Математического обеспечения ЭВМ факультета ВМК ННГУ
  valentina.kustikova@gmail.com