Нижегородский государственный университет им. Н.И. Лобачевского

Разработка мультимедийных приложений с использованием библиотек OpenCV и IPP

Лабораторная работа
Сравнение производительности некоторых алгоритмов в библиотеках OpenCV и IPP

При поддержке компании Intel

Сысоев А.В., Мееров И.Б.,
кафедра математического обеспечения ЭВМ

Содержание

- Цели и задачи работы
- Тестовая инфраструктура
- Разработка приложения для сравнения производительности операций над изображениями, реализованных на базе OpenCV и Intel IPP
- Сравнение производительности операций над изображениями, реализованных на базе OpenCV и Intel IPP
- Дополнительные задания
- Литература

Введение

- Производительность является одной из важнейших характеристик программного обеспечения.
- Для достижения хорошей производительности прежде всего необходимо определиться с тем, что это такое, и научиться ее оценивать, как теоретически, так и экспериментально.
- Оценка и анализ производительности – вещи субъективные, во многом зависящие от методики постановки и анализа результатов вычислительного эксперимента. Правильная постановка эксперимента для анализа производительности крайне важна.

Нужно понимать, что и как мы сравниваем!

Что не является целью работы?

- Мы **не хотим показать**, что библиотека IPP работает быстрее библиотеки OpenCV или наоборот (сравнение килограммов с сантиметрами ☺).

- Тем не менее, в ряде случаев библиотеки предоставляют реализации одних и тех же алгоритмов, применимых при решении задач компьютерного зрения.

  Для таких алгоритмов интересно сравнить время работы.

  Попутно научимся:

  – готовить тестовые данные;

  – использовать возможности библиотек.

Цели и задачи

Цель работы – сравнение производительности некоторых алгоритмов, реализованных как в библиотеке OpenCV, так и в библиотеке IPP.

- Построить каркас приложения для сравнения производительности.
- Запрограммировать вызов алгоритмов медианной фильтрации, эрозии, дилатации, построения гистограммы из библиотек OpenCV и IPP.
- Подобрать подходящие тестовые данные.
- Запустить приложение на выбранных данных, сравнить время работы алгоритмов.

Тестовая инфраструктура

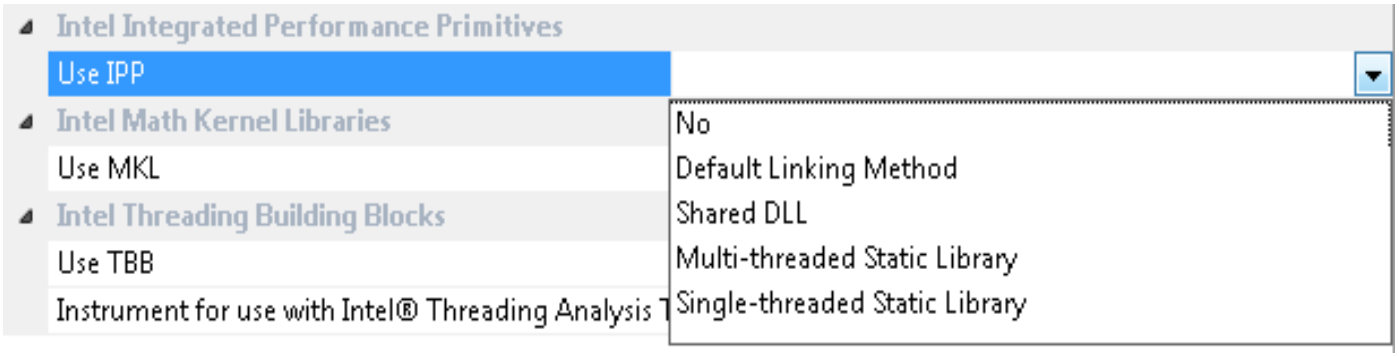
- В работе использовалось следующее аппаратное и программное обеспечение:
  - CPU: Intel Core2 Quad Q9550 2.83 GHz.
  - RAM: 5 GB.
  - OS: Windows 7 Pro SP1.
  - Microsoft Visual Studio 2010.
  - IPP из Intel Parallel Studio 2013 XE.
  - OpenCV 2.4.2.

Разработка приложения для сравнения производительности…

- Сравнение производительности будем проводить на реализациях некоторых алгоритмов обработки изображений из библиотек OpenCV и IPP.
- В OpenCV и IPP по-разному устроены базовые структуры данных.
- Используем OpenCV для загрузки изображений (в IPP нет средств загрузки изображений). Преобразуем данные для работы с IPP.
- Алгоритмы для сравнения:
  - медианная фильтрация, эрозия, дилатация, гистограмма.

Разработка приложения для сравнения производительности…

- Создаем проект в Visual Studio.
- Properties/Configuration Properties/Intel Performance Libraries



- Additional Include Directories: c:\OpenCV\build\include
- Additional Library Directories: c:\OpenCV\build\x86\vc10\lib
- Additional dependencies (debug/release):
  opencv_core242[d].lib, opencv_imgproc242[d].lib, opencv_highgui242[d].lib, opencv_legacy242[d].lib, opencv_video242[d].lib, opencv_ml242[d].lib, opencv_objdetect242[d].lib

Разработка приложения для сравнения производительности…

```
#include <stdio.h>
#include <time.h>
#include <opencv2/opencv.hpp>
#include <ippi.h>
#include <ippcc.h>
using namespace cv;
char helper[] =
\t<num_of_exp> - number of experiments\n";
"01_OpenCVvsIPP.exe <img_name> <mode> <num_of_exp>\n\
\t<img_name> - image filename\n\
\t<mode>:\n\
\t\t1 - median filtering\n\
\t\t2 - erode\n\
\t\t3 - dilate\n\
\t\t4 - calc histogram\n\
```

Разработка приложения для сравнения производительности…

```
int main(int argc, char *argv[]) {
  Mat srcImgOCV, srcImgIPP, dstImgOCV, dstImgIPP;
  int mode, i;
  double ocv_time, ipp_time;
  if (argc < 4) {
    printf("%s", helper);
    return 1;
  }
  // загрузить изображение
  srcImgOCV = imread(argv[1]);
  if (srcImgOCV.data == 0) {
    printf("ERROR!!! imread(...)");
    return 1;
  }
  srcImgOCV.copyTo(srcImgIPP);
```

Разработка приложения для сравнения производительности…

```
mode = atoi(argv[2]);
switch (mode) {
case 1:
  ocv_time = median_opencv(srcImgOCV, dstImgOCV);
  ipp_time = median_ipp(srcImgIPP, dstImgIPP);
  break;
case 2:
  ocv_time = erode_opencv(srcImgOCV, dstImgOCV);
  ipp_time = erode_ipp(srcImgIPP, dstImgIPP);
  break;
case 3:
  ocv_time = dilate_opencv(srcImgOCV, dstImgOCV);
  ipp_time = dilate_ipp(srcImgIPP, dstImgIPP);
  break;
case 4:
  ocv_time = hist_opencv(srcImgOCV, dstImgOCV);
  ipp_time = hist_ipp(srcImgIPP, dstImgIPP);
  break;
}
```

Разработка приложения для сравнения производительности

```
  printf("ocv time is %.3f\nipp time is %.3f",
          ocv_time, ipp_time);
  // освободить память
  srcImgOCV.release();
  srcImgIPP.release();
  dstImgOCV.release();
  dstImgIPP.release();
  return 0;
}
```

Функция медианной фильтрации. OpenCV

```
double median_opencv(const Mat &srcImg,
                     Mat &dstImg,
                     const int kSize = 3);

double median_opencv(const Mat &srcImg, Mat &dstImg,
  const int kSize)
{
  clock_t start, finish;
  start = clock();
  medianBlur(srcImg, dstImg, kSize);
  finish = clock();
  return double(finish - start) / CLOCKS_PER_SEC;
}
```

Функция вычисления эрозии. OpenCV

```
double erode_opencv(const Mat &srcImg, Mat &dstImg)
{
  clock_t start, finish;
  start = clock();
  Mat element = Mat();
  erode(srcImg, dstImg, element);
  finish = clock();
  return double(finish - start) / CLOCKS_PER_SEC;
}
```

Функция вычисления дилатации. OpenCV

```
double dilate_opencv(const Mat &srcImg, Mat &dstImg)
{
  clock_t start, finish;
  start = clock();
  Mat element = Mat();
  dilate(srcImg, dstImg, element);
  finish = clock();
  return double(finish - start) / CLOCKS_PER_SEC;
}
```

Функция вычисления гистограммы. OpenCV

- Будем строить гистограмму по каждому каналу цветного изображения, предварительно расщепляя его по каналам с помощью функции **split()**.

- **Задание**: реализуйте функцию вычисления гистограммы средствами OpenCV.
  При необходимости воспользуйтесь текстовым описанием лабораторной работы.

Функция медианной фильтрации. IPP

- Функция медианной фильтрации:
  - Если мы хотим применить фильтр с шаблоном размера **msk**, необходимо расширить матрицу изображения на **msk / 2** пикселов с каждой стороны. Используем функцию **ippiCopyReplicateBorder_8u_C3R()**.
  - Фильтрация: **ippiFilterMedian_<mod>**.

    В случае цветного изображения **mod** = **3uC3R**.
  - Выделить изображение исходного размера. Это можно сделать, скопировав внутреннюю часть расширенной матрицы пикселей с помощью функции **ippiCopy_8u_C3R()**.
- **Задание**: выполните программную реализацию.

Функции вычисления эрозии/дилатации. IPP

- Реализацию функций для вычисления эрозии и дилатации средствами IPP мы выполним по-другому.
- Представленные выше варианты вызовов функций **erode()** и **dilate()** из OpenCV выполняют обработку **только внутренних пикселей** исходного изображения.
- Таким образом, и в IPP-версии нам нет необходимости расширять матрицу изображения. Вместо этого мы можем соответствующим образом определить область интереса (**ROI**).
- **Задание**: выполните программную реализацию. Используйте функции **ippiDilate3x3_8u_C3R()**, **ippiErode3x3_8u_C3R()**.

Функции вычисления гистограммы. IPP

- **Задание**:
  - Изучите описание лабораторной работы.
  - Выполните программную реализацию.

Используйте функцию **ippiHistogramRange_<mod>**.

Как и ранее **mod** = **8u_C3R**

Подготовка к анализу корректности

Задание:

1. Для медианного фильтра, эрозии и дилатации результирующие изображения, полученные средствами OpenCV и IPP, можно сравнить попиксельно. Для этого напишите функцию, возвращающую число точек в двух изображениях с несовпадающими значениями.
2. Реализуйте сравнение гистограмм.

Подготовка к сравнению времени работы…

- Как выбрать тестовые данные (построить бенчмарк)?
- Какими свойствами должен обладать бенчмарк?
- На какие из этих свойств нужно обратить внимание для рассматриваемых алгоритмов?

Подготовка к сравнению времени работы

- **Запуск приложения**
  - С учетом использованных нами функций OpenCV для запуска программы потребуются следующие динамические библиотеки: opencv_core242.dll, opencv_highgui242.dll, opencv_imgproc242.dll, tbb.dll.
  - Первые три располагаются в папке build\x86\vc10\bin\. Последняя – в build\common\tbb\ia32\vc10\.
  - Использование функций IPP предполагает, что следующий путь должен быть известен операционной системе “C:\Program Files (x86)\Intel\ Composer XE 2013\redist\ia32\ipp\”.
  - Запуск: **OpenCVvsIPP.exe Desert.jpg 1**

Результаты запуска

- Intel Core2 Quad Q9550 2.83 GHz, 5 GB RAM,
  Windows 7 Pro SP1, Intel Parallel Studio 2013 XE, OpenCV 2.4.2.
- Размеры изображения: 8192×6144.

	OpenCV (время, с)	IPP (время, с)
Медианный фильтр	0.361	0.319
Эрозия	0.234	0.115
Дилатация	0.224	0.118
Гистограмма	0.351	0.130

- **Задание**:

  проведите эксперименты на серии изображений для того, чтобы время работы бенчмарка увеличилось до нескольких секунд.

Дополнительные задания

- Проведите серийные эксперименты над последовательностью изображений.
- Проведите сравнение других алгоритмов, присутствующих как в OpenCV, так и в IPP и решающих одни и те же задачи.
- Исследуйте возможность использования оптимизированных реализаций функций библиотеки IPP для повышения производительности алгоритмов библиотеки OpenCV. Изучите вопрос о совместном использовании библиотек.

Литература…

- Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Построение и анализ. – М.: МЦНМО, 2001.
- Coppersmith D., Winograd S. Matrix Multiplication via Arithmetic Progressions // J. Symbolic Computation, 9(3). – P. 251–280, 1990.
- Касперски К. Техника оптимизации программ. Эффективное использо-вание памяти. – BHV, 2003.
- Gerber R., Bik A.J.C., Smith K.B., Tian X. The Software Optimization Cookbook. High-Performance Recipes for the Intel® Architecture. – Intel Press, 2006.
- Bik A.J.C. The software vectorization handbook. Applying Multimedia Ex-tensions for Maximum Performance. – IntelPress, 2004.

Литература

- Таненбаум Э., Вудхалл А. Операционные системы. Разработка и реализация – 3-е изд. – СПб.: Питер, 2007.
- Hennessy J., Patterson D. Computer Architecture: A Quantitative Approach. –Morgan Kaufmann, 2006.
- Форсайт Д., Понс Ж. Компьютерное зрение. Современный подход. – М.: Изд. д. Вильямс, 2004. – 465с.
- Szeliski R. Computer Vision: Algorithms and Applications. – Springer, 2010.

Ресурсы сети Интернет…

- Сиднев А.А., Сысоев А.В., Мееров И.Б. Лабораторная работа №3 – Параллельная сортировка вещественных чисел за линейное время // Материалы образовательного комплекса «Параллельные численные методы». – Нижний Новгород, 2010. –
  URL: http://www.hpcc.unn.ru/file.php?id=458
- Официальная страница библиотеки Intel® Math Kernel Library [http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/].
- Официальная страница библиотеки OpenCV [http://www.opencv.org].

Ресурсы сети Интернет

- Intel® Integrated Performance Primitives (Intel® IPP) 7.1 [http://software.intel.com/en-us/intel-ipp]
- Intel® Integrated Performance Primitives for Linux* OS User's Guide [http://software.intel.com/sites/products/documentation/doclib/iss/2013/ipp/ipp_userguide_lnx/index.htm].
- Intel® Integrated Performance Primitives Reference Manual - Volume 1: Signal Processing [http://software.intel.com/sites/products/documentation/doclib/iss/2013/ipp/ipp_manual/ipps.pdf].
- Intel® Integrated Performance Primitives Reference Manual - Volume 2: Image and Video Processing [http://software.intel.com/sites/products/documentation/doclib/iss/2013/ipp/ipp_manual/ippi.pdf].

Авторский коллектив

- Сысоев Александр Владимирович,
  к.т.н., ассистент каф.
  Математического обеспечения ЭВМ факультета ВМК ННГУ
  sysoyev@vmk.unn.ru
- Мееров Иосиф Борисович,
  к.т.н., доцент, зам. зав. каф.
  Математического обеспечения ЭВМ факультета ВМК ННГУ
  meerov@vmk.unn.ru