UNIVERSITY OF WESTERN MACEDONIA
FACULTY OF ENGINEERING
Department of Informatics & Telecommunications Engineering

Computer Vision with OpenCV C++/Python

Tourtouras Evangelos

Supervisor: Dr. Minas Dasygenis

Date: 19/6/2018

Contents

- Module A: Introduction
    1. The theory of computer vision.
    2. The importance of it.
    3. The Basic functionality of OpenCV programming library with examples in C++ and Python.
- Module B: Advance subjects
    1. Feature extraction
    2. Background subtraction
    3. Object Detection

Module A

In this lecture, the students will be introduced to:

- The theory of computer vision.
- The importance of it.
- The Basic functionality of OpenCV programming library with examples in C++ and Python.

Computer Vision

- It is a field that is trying to make computers to understand at an advanced level what you gain from images or videos.
- It is the data that transformed from a still or video camera.
- From the side of engineering, it is trying to find a way to automate tasks that also can be done by our visual system.

What methods does computer vision include?

- Acquiring digital images.
- Processing digital images.
- Analyzing digital images.
- Understanding digital images.
- Extracting high- dimensional data from the real World.
- As for science, it is believed that is the theory behind artificial systems which can gain information by the images.

The Importance of Computer vision #1/12C

- Video Event Detection

We can detect events happening on a video by searching its data directly.

In videos, there are elements for each event or a set of elements that help us for the process.

The Importance of Computer vision #2/12

- Machine learning is using techniques that are statistical so it can give computer systems the ability to learn. We can use machine learning in various computing tasks such as email filtering.

The Importance of Computer vision #3/12

- Indexing

  We use Indexing to optimize access to data records which are organized in files.
- Scene reconstruction is how we can recover a 3D scene from pictures or photographs so that we can extract and track from the objects.

The Importance of Computer vision #4/12

- Motion estimation

  When we want to describe the transformation of one 2D image to another we use this processes of determining motion vectors.

  This motion vectors can be related to specific parts of the image but it may also relate to the whole image that's what we call "GLOBAL MOTION ESTIMATION".

The Importance of Computer vision #5/12

- Image restoration

  This is a method that we can restore a corrupted image. The corruption of an image may be caused by noise, motion blur or with camera miss-focus.

The Importance of Computer vision #6/12

- 3D Pose estimation

  This is a problem of how to define the transformation of an object in a 2D image to a 3D object. This problem is appearing from the limitations of feature-based pose estimation.

The Importance of Computer vision #7/12

- Video tracking

  With this process, we can locate a moving object or multiple objects over time with the use of cameras.

  We can use this process for :

  1. video communication and compression
  2. human-computer interaction
  3. medical imaging

The Importance of Computer vision #8/12

- Video tracking

  We can also use this process for:

  1. video editing
  2. security and surveillance
  3. traffic control
  4. augmented reality

The Importance of Computer vision #9/12

- Object recognition

  This technology allows us to find or identify objects from an image or a video sequence.

  Object Recognition it is best for retail and fashion to find products in real-time based off of an image or scan and much more.

The Importance of Computer vision #10/12

- Optical Character Recognition (OCR) with these we can recognize and identify text in documents.
- Medical Imaging: we can obtain 3D imaging and image-guided surgery.
- Sports: In a game when they draw additional lines on the field.

The Importance of Computer vision #11/12

- Vision Biometrics where we can Recognize people who have been missing through iris patterns.
- Smart Cars: Through computer vision, they can identify objects and humans.
- Special Effects: Motion capture and shape capture, any movie with CGI.
- 3-D Printing and Image Capture: Used in movies, architectural structures and much more.
- Social Media: Anything that has a story which allows you to wear something on your face.

The Importance of Computer vision #12/12

In conclusion, the are various domains for computer vision such as video games, computer vision in Space, medicine, robotics and in other industries too. May Computer Vision is one of the most easier terms to define but it is also difficult to teach computers. The proof of that matter is that it has past at least 80 years for the AI and deep learning to reach its level we see it now and still keep evolving.

Introduced to OpenCV (Open Source Computer Vision Library) #1/3

- OpenCV is a library with a lot of programming functions and it is focused on real-time computer vision.
- OpenCV was developed by an Intel research initiative and the purpose of it was to advance CPU-intensive applications.

Introduced to OpenCV (Open Source Computer Vision Library) #2/3

Intel continued to launch projects such as 3D display walls and real-time ray tracing. OpenCV was created to make computer vision infrastructure available for many things.

Some of the goals for the OpenCV was:

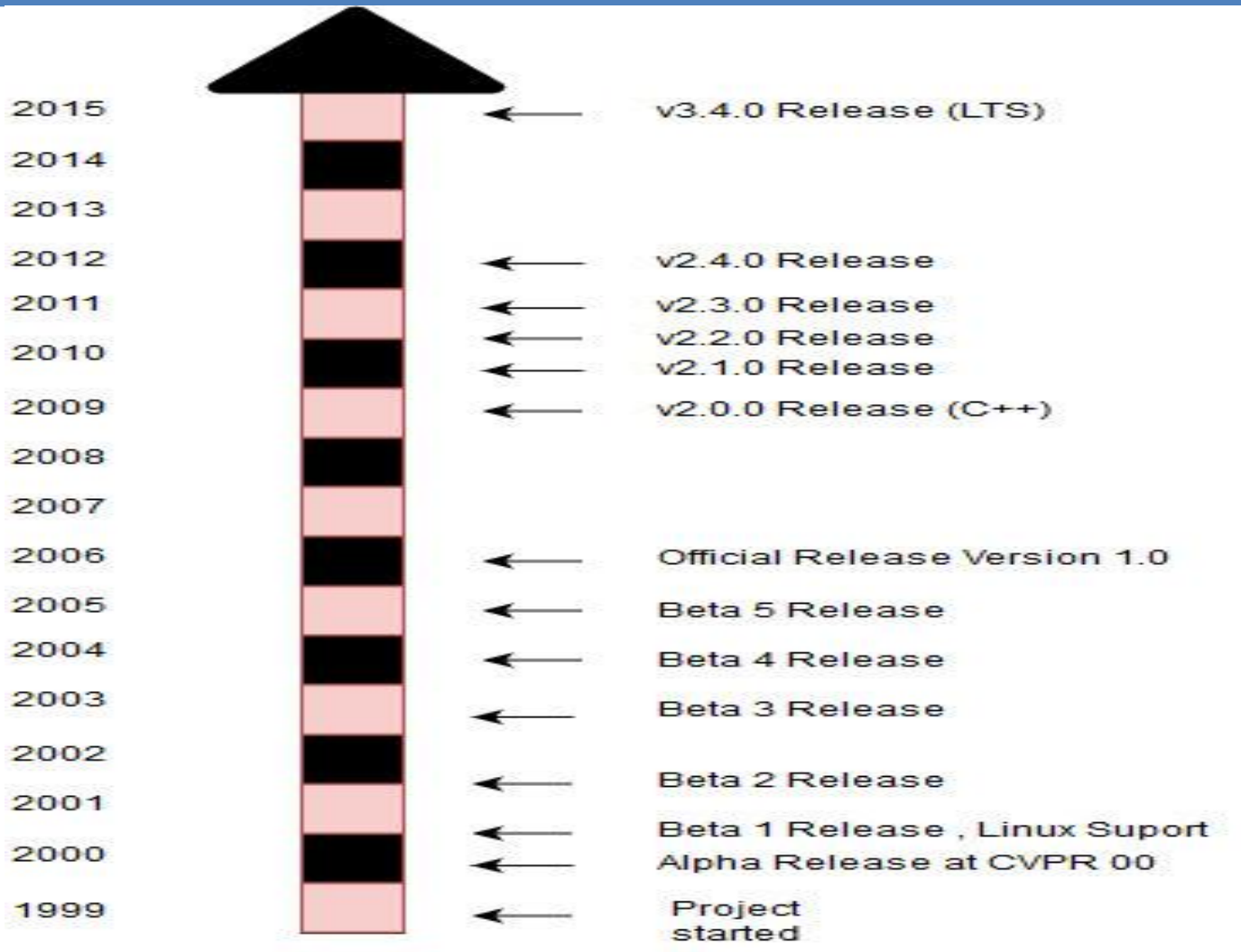
- To advance vision research so they provided open and they had optimized the code for basic vision vi infrastructure.
- To intersperse the knowledge of computer vision so they gave a common infrastructure so the developers could build on and the code could be readable and transferable.
- To advance vision-based commercial applications to achieve that, they wanted to make an portable performance –optimized code which could be available for free.

Introduced to OpenCV (Open Source Computer Vision Library) #3/3

Generally, OpenCV is receiving many user contributions. OpenCV was under active development at Willow Garage and now there is an OpenCV foundation that supports it.

Nowadays OpenCV is still evolving with the help both of OpenCV foundation and several public and private institutions.

OpenCV timeline from 1999 to 2015



The block diagram of OpenCV

- The built of OpenCV is in layers. In the top of the layer is the Operating System (OS) that we use to operate OpenCV.
- In the next layer, we have the opencv_contrib which is the contributed code. This code includes higher-level functionality.
- The next layer is the core of OpenCV.
- At the last layer which is the bottom layer, there are many hardware optimizations we call that layer "hardware acceleration layer".

Block Diagram Of OpenCV

Operating Systems (OS)				
LINUX	**WINDOWS**	**iOS**	**ANDROID**	**OSX**
Bindings: Java, Python			**Apps, Samples, Solutions**	
Opencv_contrib				
rgbd, text, face, etc..				
OpenCV				
Object detection, core, image processing, etc..				
OpenCV HAL				
OpenCL, OpenCV4Tegra, NEON, SSE, CUDA, IPP, etc..				

OpenCV Contribution Repository

The old library of OpenCV has been split into two parts these parts are:

1. OpenCV mature.
2. Current state of the art in larger vision functionality at opencv_contrib.

The first is maintained by the core OpenCV group and it includes mostly stable code, on the other hand, the second is less mature, is developed and maintained by the Programmer communities for open source.

A few modules of opencv_contrib repository #1/2

- Tracking (This has modern object tracking algorithms).
- rgdb (This is for processing depth maps and RGB obtained by depth sensors such as Kinect or it can be computed by stereo correspondence algorithms).
- Face (Face recognition)
- Dnn (Deep neural networks)

A few modules of opencv_contrib repository #2/2

- Text (It can recognize and detect text, it might use optionally OCR Tesseract open source as backend).
- xphoto, ximgproc (This is for both computational photography and advanced image processing).
- Bioinspired (which is biologically inspired vision).

OpenCV portability

- The goal for the design of OpenCV was to be portable. OpenCV code was created to compile by any compliant C++ compiler. That helped to make cross-platform support easier.
- There is support for systems of Intel and AMD which is more mature but the ARM support is getting an improvement over the time.

OpenCV portability guide for 1.0 released version

Compatibility	Architecture x86/x64	ARM	PPC,MIPs ..
Linux	Parallel, [1]I/O, IPP, SIMD	Parallel, [1]I/O, SIMD	Parallel, [1]I/O
Windows	Parallel, I/O, IPP, SIMD	Parallel(3.0), I/O, SIMD	N/A
OS X/ iOS	Parallel, [2]I/O, IPP(3.0), SIMD	Parallel, I/O, SIMD	N/A
Android	Parallel, I/O, IPP(3.0), SIMD	Parallel, [2]I/O, SIMD	MIPS basic support
QNX,BSD,etc.	SIMD	SIMD	-

1. **The parallelization in Linux can be done by enabling OpenMP or via a third – party library.**
2. **In Android the parallelization is done by Intel TBB.**

Download and install OpenCV #1/34

- You can download the complete source code and the latest updates from the main OpenCV page, which is: https://opencv.org/releases.html
- If you want more recent higher functionality you can download and build opencv_contrib via GitHub in the following link: https://github.com/opencv/opencv_contrib

Download and install OpenCV #2/34

- Nowadays OpenCV uses Git as its development version control system and CMake to build.
- For building, there are compiled libraries for environments.
- As far as you go in OpenCV and become a high-level user, you will probably want to recompile the libraries with options that are more suited to your application.

Download and install OpenCV #3/34

- Windows: You can find and download OpenCV in the following link: https://opencv.org/releases.html
- After you Download the executable file you are almost ready to use OpenCV.
- If you want to make easier for your compiler to find the OpenCV binaries you need to add an OPENCV_DIR environment variable.

Download and install OpenCV #4/34

- To do that you need to go to command prompt and type the following: "setx –m OPENCV_DIR C:\OpenCV\Build\x64\vc10".
- OpenCV3 has IPP linked in, that means that you are going to get performance advantage of modern x86 and x64 CPUs.

Download and install OpenCV #5/34

- An alternative way is to Build OpenCV via a source tarball. The steps are:
    a) Run the CMake GUI.
    b) Specify paths to the source tree of the OpenCV and the build directory.
    c) Press Configure two times, choose the appropriate generator of Visual Studio or MinGW makefiles if you are going to use MinGW and then press Generate.
    d) After you Generate it you need to open the generated solution in visual studio and built it. If we are going to use MinGW we need to check the next page that explains the installation of OpenCV in Linux.

Download and install OpenCV #6/34

- Linux: The link to download OpenCV is the same to the link for windows. In Linux to build libraries and Demos we will need the following:
    a) GTK+ 2.x or higher, including headers.
    b) We will need a gcc compiler.
    c) The essential development packages.
    d) cmake, libtbb, zlib, libjpeg, libpng, libtiff and libjasper.
    e) And last the development files, i.e. the versions with –dev at the end of their package names.

Download and install OpenCV #7/34

- In order to make Python buildings to work in Linux, we are going to need a Python 2.6 or later version with the headers being installed and NumPy.
- We are going to need libav* libraries such as libavcodec from *ff*mpeg.

Download and install OpenCV #8/34

- To download the libav/*ff*mpeg packages we need to visit the following webpage: http://www.ffmpeg.org/.
- If you want to use it with non-GPL software, build and use a shared *ff*mpeg library:

```
$> ./configure –enable-shared
$> make
$> sudo make install
```

Download and install OpenCV #9/34

- In the end we are going to have the following
    1. /usr/local/lib/libavcodec.so.*
    2. /usr/local/lib/libavformat.so.*
    3. /usr/local/lib/libavutil.so.*

- And the we can include files with various paths like /usr/local/include/libav*.

Download and install OpenCV #10/34

- Actually to build the library, unpack the .tag.gz file, after that go to the created source directory and type this :

```
mkdir release
cd release
cmake –D
  CMAKE_BUILD_TYPE=RELEASE –D
  CMAKE_INSTALL_PREFIX=/usr/local ..
make
sudo make install # optional
```

Download and install OpenCV #11/34

- Lets see step by step how to install open cv3 in ubuntu.
- Step one: We need to update and upgrade the packages of our Operation System.
    1. sudo apt-get update
    2. sudo apt-get upgrade

Download and install OpenCV #12/34

- Step two: We need to Install the Operation System Libraries.
    1. Remove any previous installations of x264</h3>
    2. sudo apt-get remove x264 libx264-dev
    3. We will Install dependencies now
    4. sudo apt-get install build-essential checkinstall cmake pkg-config yasm

Download and install OpenCV #13/34

- Step two: We need to Install Operation System Libraries.
  5. sudo apt-get install git gfortran
  6. sudo apt-get install libjpeg8-dev libjasper-dev libpng12-dev
  7. # If you are using Ubuntu 14.04
  8. sudo apt-get install libtiff4-dev

Download and install OpenCV #14/34

- Step two: We need to Install Operation System Libraries.

```
9.  # If you are using Ubuntu 16.04
10. sudo apt-get install libtiff5-dev
11. sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev
12. sudo apt-get install libxine2-dev libv4l-dev
```

Download and install OpenCV #15/34

- Step two: We need to Install Operation System Libraries.

    13. sudo apt-get install libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev
    14. sudo apt-get install qt5-default libgtk2.0-dev libtbb-dev
    15. sudo apt-get install libatlas-base-dev

Download and install OpenCV #16/34

- Step two: We need to Install Operation System Libraries.
    16. sudo apt-get install libfaac-dev libmp3lame-dev libtheora-dev
    17. sudo apt-get install libvorbis-dev libxvidcore-dev
    18. sudo apt-get install libopencore-amrnb-dev libopencore-amrwb-dev
    19. sudo apt-get install x264 v4l-utils

Download and install OpenCV #17/34

- Step two: We need to Install Operation System Libraries.

20. # Optional dependencies
21. sudo apt-get install libprotobuf-dev protobuf-compiler
22. sudo apt-get install libgoogle-glog-dev libgflags-dev
23. sudo apt-get install libgphoto2-dev libeigen3-dev libhdf5-dev doxygen

Download and install OpenCV #18/34

- Step three: We need to Install Python Libraries.
    1. sudo apt-get install python-dev python-pip python3-dev python3-pip
    2. sudo -H pip2 install -U pip numpy
    3. sudo -H pip3 install -U pip numpy

Download and install OpenCV #19/34

- Step three: We need to Install Python Libraries using the virtual Environment.
    1. # Install virtual environment
    2. sudo pip2 install virtualenv virtualenvwrapper
    3. sudo pip3 install virtualenv virtualenvwrapper
    4. echo "# Virtual Environment Wrapper" >> ~/.bashrc
    5. echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc
    6. source ~/.bashrc

Download and install OpenCV #20/34

- Step three: We need to Install Python Libraries using the virtual Environment.

```
1. # For Python 2 #
2. # create virtual environment
3. mkvirtualenv facecourse-py2 -p python2
4. workon facecourse-py2
5. # now install python libraries within this
   virtual environment
6. pip install numpy scipy matplotlib scikit-
   image scikit-learn ipython
7. # quit virtual environment
8. deactivate
```

Download and install OpenCV #21/34

- Step three: We need to Install Python Libraries using the virtual Environment.

```
1. # For Python 3 #
2. # create virtual environment
3. mkvirtualenv facecourse-py3 -p python3
4. workon facecourse-py3
5. # now install python libraries within this
   virtual environment
6. pip install numpy scipy matplotlib scikit-
   image scikit-learn ipython
7. # quit virtual environment
8. deactivate
```

Download and install OpenCV #22/34

- Step four: We need to Download OpenCV and OpenCV contrib from their GitHub repositories.
    1. # Download opencv from Github
    2. git clone https://github.com/opencv/opencv.git
    3. cd opencv
    4. git checkout 3.3.1
    5. cd ..

Download and install OpenCV #23/34

- Step four: We need to Download OpenCV and OpenCV contrib from their GitHub repositories.
    1. # Download opencv_contrib from Github
    2. git clone https://github.com/opencv/opencv_contrib.git
    3. cd opencv_contrib
    4. git checkout 3.3.1
    5. cd ..

Download and install OpenCV #24/34

- Step four: We need to Download OpenCV and OpenCV contrib from their GitHub repositories.
    1. git clone https://github.com/opencv/opencv_contrib.git
    2. cd opencv_contrib
    3. git checkout 3.3.1
    4. cd ..

Download and install OpenCV #25/34

- Step five: Create and build directory, Run CMake.
    1. cd opencv
    2. mkdir build
    3. cd build
    4. cmake -D CMAKE_BUILD_TYPE=RELEASE \
    5. -D CMAKE_INSTALL_PREFIX=/usr/local \
    6. -D INSTALL_C_EXAMPLES=ON \

Download and install OpenCV #26/34

- Step five: Create and build directory, Run CMake.

```
7.  -D INSTALL_PYTHON_EXAMPLES=ON \
8.  -D WITH_TBB=ON \
9.  -D WITH_V4L=ON \
10. -D WITH_QT=ON \
11. -D WITH_OPENGL=ON \
12.-D
    OPENCV_EXTRA_MODULES_PATH=../../open
    cv_contrib/modules \
 13.-D BUILD_EXAMPLES=ON ..
```

Download and install OpenCV #27/34

- Step six: compile and install
  1. # find out number of CPU cores in your machine
  2. nproc
  3. # substitute 4 by output of nproc
  4. make -j4
  5. sudo make install
  6. sudo sh -c 'echo "/usr/local/lib" >> /etc/ld.so.conf.d/opencv.conf'
  7. sudo ldconfig

Download and install OpenCV #28/34

- Step seven: virtual environment, create sym link.
  1. find /usr/local/lib/ -type f -name "cv2*.so"
  2. # For Python 2 #
  3. # binary installed in dist-packages
  4. /usr/local/lib/python2.6/dist-packages/cv2.so
  5. /usr/local/lib/python2.7/dist-packages/cv2.so
  6. # binary installed in site-packages
  7. /usr/local/lib/python2.6/site-packages/cv2.so
  8. /usr/local/lib/python2.7/site-packages/cv2.so

Download and install OpenCV #29/34

- Step seven: virtual environment, create sym link.
  1. # For Python 3 #
  2. # binary installed in dist-packages
  3. /usr/local/lib/python3.5/dist-packages/cv2.cpython-35m-x86_64-linux-gnu.so
  4. /usr/local/lib/python3.6/dist-packages/cv2.cpython-36m-x86_64-linux-gnu.so
  5. # binary installed in site-packages
  6. /usr/local/lib/python3.5/site-packages/cv2.cpython-35m-x86_64-linux-gnu.so
  7. /usr/local/lib/python3.6/site-packages/cv2.cpython-36m-x86_64-linux-gnu.so

Download and install OpenCV #30/34

- Step seven: virtual environment, create sym link.
  1. # For Python 2 #
  2. cd ~/.virtualenvs/facecourse-py2/lib/python2.7/site-packages
  3. ln -s /usr/local/lib/python2.7/dist-packages/cv2.so cv2.so
  4. # For Python 3 #
  5. cd ~/.virtualenvs/facecourse-py3/lib/python3.6/site-packages
  6. ln -s /usr/local/lib/python3.6/dist-packages/cv2.cpython-36m-x86_64-linux-gnu.so cv2.so

Download and install OpenCV #31/34

- Step eight: C++ code testing
  1. # compile and run
  2. # There are backticks (`) around pkg-config command not single quotes
  3. g++ -std=c++11 removeRedEyes.cpp `pkg-config --libs --cflags opencv` -o removeRedEyes
  4. ./removeRedEyes

Download and install OpenCV #32/34

- Step nine: Python code testing
  1. # Activate virtual environment for Python 2 #
  2. workon facecourse-py2
  3. # Activate virtual environment for Python 3 #
  4. workon facecourse-py3

Download and install OpenCV #33/34

- Step nine: Python code testing
  1. ipython
  2. import cv2
  3. print cv2.__version_
  4. # If the OpenCV is perfectly installed the print cv2 should give you 3.3.1 output.
  5. # To exit ipython press Ctrl+D
  6. # to exit python virtual environment write“ deactivate”.

Download and install OpenCV #34/34

- Mac OS X: The installation procedure is similar to Linux but OS X has its own development environment, Xcode that has everything except Cmake. Also, you don't need TBB, GTK+, *ff*mpeg, libjpeg, etc..
    1. Instead of GTK+ it has by default Cocoa.
    2. Instead of *ff*mpeg it has by default QTKit.
    3. Instead of OpenMP and TBB it uses Grand Dispatch Central (GDC).
    4. If you want you can pass the –G Xcode option to make Cmake Xcode project for OpenCV to build and debug the code within Xcode.

Header files OpenCV C++ #1/6

- To call the header files of each OpenCV module we use the main file header which is the following: "…/include/opencv2/opencv.hpp;".
- For old data structures in C and arithmetic routines we use #include <opencv2/core/core_c.h>.
- For new data structures in C++ and arithmetic routines we use #include <opencv2/core/core.hpp>.

Header files OpenCV C++ #2/6

- For background segmentation routines and video tracking we use #include <opencv2/video/video.hpp>.
- For image processing functions in old C we use #include <opencv2/imgproc/imgproc_c.h>.
- For image processing functions in new C++ we use #include <opencv2/imgproc/imgproc.hpp>.

Header files OpenCV C++ #3/6

- For two-dimensional feature tracking support we use #include <opencv2/features2d/featutes2d.hpp>.
- For using specific algorithms to handle and restore photographs we use #include <opencv2/video/photo.hpp>.
- For approximate the nearest neighbor matching functions we use #include <opencv2/flann/miniflann.hpp>.

Header files OpenCV C++ #4/6

- For user-contributed code such as fuzzy mean-shift tracking, self- similar features, flesh detection, and spin images we use #include <opencv2/contrib/contrib.hpp>.
- For planar patch detector, Cascade face detector, HoG and latent SVM we use #include <opencv2/objdetect/objdetect.hpp>.
- For machine learning such as pattern recognition and clustering we use #include <opencv2/ml/ml.hpp>.

Header files OpenCV C++ #5/6

- For image display ,mouse interaction, I/O and sliders in old C we use #include <opencv2/highgui/highgui_c.h>.
- For image display, mouse, I/O, sliders and buttons in new C++ we use #include <opencv2/highgui/highgui.hpp>.
- For stereo and calibration we use #include <opencv2/calib3d/calib3d.hpp>.

Header files OpenCV C++ #6/6

- If we want to include every possible OpenCV function we can use the include file opencv.hpp but it comes with a drawback and that drawback is that the compile time will have a delay.
- Example: If you want to use includes for image processing you can include the opencv2/imgproc/imgproc.hpp instead of opencv.hpp, it will compile faster. You can find the .hpp file in …/modules/imgproc/include/opencv2/imgproc/imgproc.hpp.

First example: Display an image #1/15

- One of the most common utilities that OpenCV provides is reading from a wide array of video or image file types.

```
//Here is an example of an OpenCV program that loads an image from the disc
//and displays that image on the screen.

// As we said before the opencv.hpp  support every function of OpenCV
// but the compile process  is going slow
#include <opencv2/opencv.hpp>

int main(int argc, char** argv)
{

    cv::Mat img = cv::imread(argv[1], -1);
    if( img.empty())
    {
        return -1;
    }
    cv::namedWindow( "img_example1", cv::WINDOW_AUTOSIZE);
    cv::imshow("img_example1",img);
    cv::waitKey(0);
    cv::destroyWindow("img_example1");
    return 0;
}
```

First example: Display an image #2/15

- As we saw in the previous example we use in each start **cv::** that means we need to tell the compiler that we are talking about the **cv** namespace.
- cv namespace is where the functions of OpenCV exists.
- To avoid using all **th**e time cv:: we can use once this: "using namespace cv;".

First example: Display an image #3/15

- Now we will see an example using "using namespace cv;" in the first example, we use general include opencv.hpp but in this example, we will use only the necessary include file to improve the compile time.
- In this example, there is a risk of conflicting names with other potential namespaces. For example, if the function f() exist, say in the cv and std names that you must specify the function you are talking about by using cv::f() or std::f(). In this example, we will use the explicit from cv:: for the objects in the namespace of OpenCV. This is considered to be a better programming method.

First example: Display an image #4/15

```
 //Second example about how we display an image
// In this example instead of #include <opencv2/opencv.hpp
// which is slow to compile we use direct include file
//#include "opencv2/highgui/highgui.hpp" for faster compile processing
//also as you notice instead of using the namespace cv::  we use
//once using namespace cv;
#include "opencv2/highgui/highgui.hpp"

using namespace cv;

int main (int argc, char** argv){

        Mat img= imread( argv[1], -1);

        if( img.emty())
        {
                return -1;
        }

        nameWindow( "img_example2", cv::WINDOW_AUTOSIZE):
        imshow("img_example2", img);
        waitKey(0);

        desrtoyWindow("img_example2");
}
```

First example: Display an image #5/15

- If we compile and run via command line with a single argument, in the first example loads an image into memory and it displays it in the screen after that it waits from the user to press a Key to close a window and to exit.
- Definitely the built instructions are highly platform dependent. An example about how you can compile the second example in Unix: gcc -v img_example2.cpp -I/usr/local/include/ -L/usr/lib/ -lstdc++ -L/usr/local/lib -lopencv_highgui -lopencv_core - -o img_example2 .

First example: Display an image #6/15

- Lets explain the line that loads the image in the first example which is "cv:: Mat img = cv:: imread(argv[1],-1);". First of all, I would like to make a point, it is essentially for a good programmer to understand how much important is the error –handling code.
- The high-level routine cv::imread() can determine the file format that will be loaded based on the filename and it allocates automatically the memory that it needs for the image data structure.

First example: Display an image #7/15

The following image formats can be read by the function cv::imread():

- JPEG
- PNG
- BMP
- JPE
- DIB
- TIFF
- PBM
- SR
- RAS
- PPM
- PGM

First example: Display an image #8/15

- The structure that is returned is cv::Mat.
- The cv::Mat structure can handle all kind of images such as:
    1. floating-point-valued
    2. Multichannel
    3. Integer-valued
    4. Single channel
    5. etc..

First example: Display an image #9/15

- The next line is: " if(img.empty()) { return -1};". In this line we check if an image was read.
- The next function which is a high level function is: "cv::namedWindow()". This function can open a window on the screen which contains and displays an image.

First example: Display an image #10/15

- Since we explained previously the function "cv::namedWindow()" we can now explain the next line of the program that contains this function which is :"cv::namedWindow("img_example1", cv::WINDOW_AUTOSIZE);.
- This function also can assign a name to the window, the name we chose to give is "img_example1". The HighGUI which provide us this function can call the interact with this window by referring to it by it the name we gave.

First example: Display an image #11/15

- The argument that defines the properties of the window is the second argument to cv::nameWindow().
- We can set the value to default or we can set it to cv::WINDOW_AUTOSIZE.
- If we set the value to default the size of the window will be the same regardless of the image size. The image will fit within the window. If we use cv::WINDOW_AUTOSIZE will contract automatically or it will expand when the image is loaded so it will accommodate the true size of the image.

First example: Display an image #12/15

- The next line is : “cv::imshow("img_example1",img);”. With this function since we have an Image in the structure cv::Mat we can display the image in an existing window.
- This function can also create a window if we have no window.
- When we call this function the window will be redrawn with the image on it and it will resize itself if it was created by the flag : cv::WINDOW_AUTOSIZE.

First example: Display an image #13/15

- The next line that we will explain is: "cv::waitKey(0);".
- This function asks the program to stop and wait until we press a Key.
- If we give an argument that is positive the program will wait for a specific time and then it will be continued no matter if no Key is pressed.
- If we set the argument to a negative number or to 0 the program will wait till we press a key.

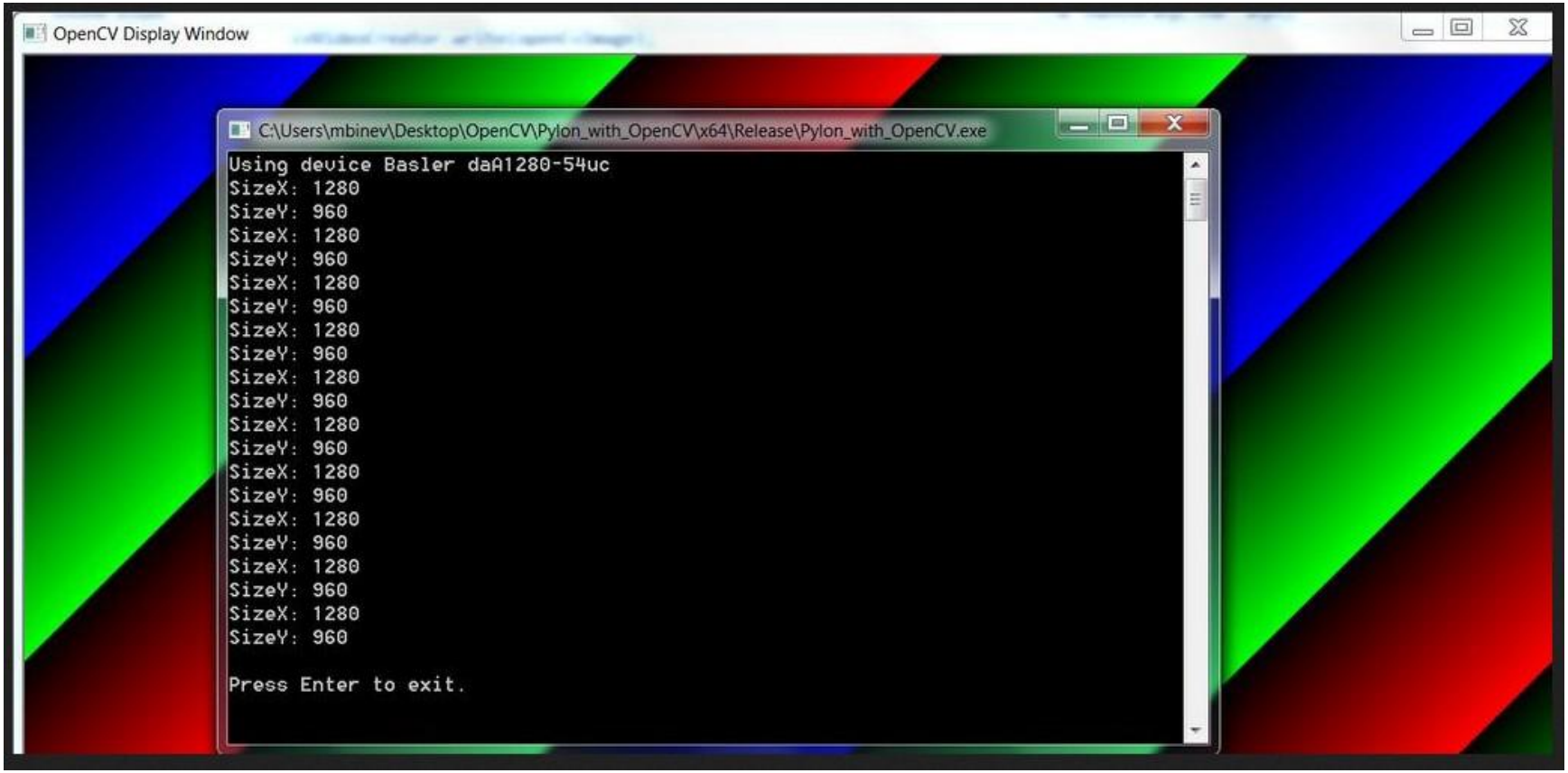
First example: Display an image #14/15

- There is the Standard Template Library (STL) which contains classes that make the images to automatically deallocated when they go out of scope. "cv::Mat" is similar to (STL).
- We can control this automatic deallocation with an internal reference counter.

First example: Display an image #15/15

- The final line of the code that we are going to explain is: "cv::destroyWindow("img_example1");".
- With this function, we can close the window and deallocate the associated memory usage.
- If we have short programs we are going to skip that step but if we have larger programs which might be more complex, we the programmers should make sure to tidy up the windows before they go out of the scope so we can avoid any memory leaks.

Display an image C



Source file: http://technodocbox.com/docs-images/71/66220294/images/11-1.jpg, last time visited 5/29/2018.

Second example: Video processing #1/6

- In video processing, we have a new issue that we face. That issue is that we need some kind of loop to read each frame in a row of time (sequence).
- Since we need that loop we also might need a method or a way to get out of that loop we created.

Second example: Video processing #2/6

- **Here is an example of playing video from a disk via OpenCV.**

```
// We include the both higui.hpp and imgproc.hpp
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/higui/higui.hpp"

int main(int argc, char** argv )
{
cv::namedWindow( "video_example1", cv::WINDOW_AUTOSIZE);
cv::VideoCapture  capture;
capture.open( string(argv[1]));

cv::Mat frame;
for(;;)
{
        capture >> frame;
// we use now if. if we run out of film
                if( frame.empty() )
                {
                        break;
                }
                cv::imshow("video_example1", frame);
                        if( cv::waitKey(33) >=0)
                        {
                                break;
                        }
}
        return 0;
}
```

Second example: Video processing #3/6

- In this example firstly we name the window "video_example1".
- Then we create a video capture object with the "cv::VideoCapture capture;" we gave the name capture in the object then it is instantiated.
- With the object we created can open and close video files of as many types as *ff*mpeg supports.

Second example: Video processing #4/6

- Lets see the lines "capture.open(string(argv[1]));" and "cv::Mat frame;".
- We give a string to the object that we created that contains the path and the filename of the video we want to open.
- The object will have all the information about the video file it read and the state information.

Second example:Video processing #5/6

- The "cv::Mat frame;" instantiates a data object to hold video frames.
- The code that we have inside the loop which is: "capture >> frame;","
  if(frame.empty()){break;}","
  cv::imshow("video_example1", frame);" , first it will read frame by frame the capture object stream then it will check the if the data of the video was read if it was not read it quits.
- When the video frame was read it can be displayed by "cv::imshow()".

Second example: Video processing #6/6

- Now we will explain the last line which is: "if(cv::waitKey(33) >=0){break;}", In this function, we can give the amount of time we want to wait we gave it 33 ms to wait.
- When the frame is completed displayed it waits 33 ms.
- In that time if we press a key we are going to exit from the loop.
- If we let the 33 ms to pass without pressing any key the program will execute the loop again.
- And last, when we exit, the allocated data is being released automatically when it goes out of scope.

Example for video: Moving around #1/23

- In the first example of the video example, we can see that in the video player we cannot move around quickly within the video.
- To solve that problem in this example we are going to insert a slider track bar. This will allow us to move quickly within the video.
- Also, we are going to give the user more control. We are going to let the user single step the video by pressing "t" from the keyboard and we are going to allow the user to go into the run mode by pressing "r" from. When the user jumps in a new location in the video with the track bar that we will create, we can pause there in single step mode.

Example for video: Moving around #2/23

- The function we are going to call for creating the track bar is "createTrackbar()".
- This function also indicates which window we are going to set the track bar to appear in.
- For this to work well we are going to need a callback that will do the relocation.

Example for video: Moving around #3/23

```
 // This is the second example for OpenCV: video.
//In this example we are going to add a track bar slider
//in the basic window viewer that will allow us to
// move around within the video file.
#include <fstream>
#include <iostream>
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highui/highgui.hpp"

using namespace std;
// we also could use the
// using namespace cv to
// avoid using cv:: all the time.
int g_dontset=0, g_run=1;

// we  set the slider's possition.

int g_slider_position=0;

//now we create the object g_ capture.

cv::VideoCapture g_capture;
```

Example for video: Moving around #4/23

```
// we create the track bar slide function.

void onTrackbarSlide(int position, void *)
{
   // we set the position to the object we created.
        g_capture.set( cv::CAP_PROP_POS_FRAMES,
   position);

        if( !g_dontsep)
                g_run=1;
                g_dontset=0;

}
```

Example for video: Moving around #5/23

```
 int main( int argc, char** argv )
{

    cv::nameWindow("video_example2", cv::WINDOW_AUTOSIZE);
    g_capture.open(string (argv[1]) );
    int tmph=(int) g_capture.get(cv::CAP_PROP_FRAME_HEIGHT);
    int frames=(int) g_capture.get(cv::CAP_PROP_FRAME_COUNT);
    int tmpw=(int) g_capture.get(cv::CAP_PROP_FRAME_WIDTH);

    cout << "The video has :" << frames << "frames of the dimensions
    (" <<tmpw <<","<< tmph <<")."<< endl;

    cv::createTrackbar("Position",
    "video_example2",&g_slider_position,frames,onTrackbarSlide);

    cv::Mat frame;
```

Example for video: Moving around #6/23

```
for(;;)
    {
                                    if( g_run !=0)
                                    {
                                            g_capture >> frame;
                                            if(frame.empty())
                                            {
                                            break;
                                            }
                                            int current_position = (int)
    g_capture.get(cv::CAP_PROP_POS_FRAMES);
                                            g_dontset=1;

                                            cv::setTrackbarpos("Position","video_example2",
    current_position);
                                            c::imshow("video_example2",frame);

                                            g_run=1;
                                    }
    }
```

Example for video: Moving around #7/23

```
 char key=(char) cv::waitKey(15);
      // we set the key for single step.

      if( key== 't')
      {
               g_run=1;
               cout<<"Single step, run =" <<g_run << endl;
      }
      // now we are going to set the key for run mode
      if (key=='r')
      {
               g_run=-1;
                        cout << "Run mode, run =" << g_run << endl;
      }
      if( key==27)
      {
               break;
      }
return (0);
}
```

Example for video: Moving around #8/23

- The point is to add a global variable that represents the position of the trackbar.
- After that, we add a callback to inform the variable and then to relocate the read position in the video.
- The call that creates the track bar it also attaches the callback and we are ready to go.
- It is important to notice that some mpeg and Avi encodings do not allow us to move backward in the video.

Example for video: Moving around #9/23

- Now we are going to explain the global variables which is :"int g_dontset=0, g_run=1;","int g_slider_position=0;" and "cv::VideoCapture g_capture;".
- To keep the trackbar position state we use "int g_slider_position=0;".
- The access that the callback will need to the capture object we created which is g_capture is needed to be promoted to a global variable.

Example for video: Moving around #10/23

- The global variable g_run is displaying the new frames as long it is different from zero.
- To indicate how many frames are displayed before stopping we need a positive number on the other hand if we have a negative number the system runs in continuous video mode.

Example for video: Moving around #11/23

- More analytically If we want to jump to a new location in the video, we need to click on the trackbar.
- So we will leave the video paused there in the single step. To do that we need to set the g_run =1. Then we are going to have a little problem.

Example for video: Moving around #12/23

- The problem is while the video advances we also want the slider trackbar's position in the display window to advance due to our location in the video.
- To achieve that we need to have the main program call the trackbar callback function to keep informing (updating) the position of the slider every time we get a new frame.
- Although we don't want this calls to the trackbar callback to put us in the single step mode.
- To avoid that we need the g_dontset which is also a global variable. This variable allows us to get informed about the trackbar position without starting the single-step mode.

Example for video: Moving around #13/23

- Now we are going to explain the function "void onTrackbarSlide(int position, void *)" which contains the following code: "

```
g_capture.set(
cv::CAP_PROP_POS_FRAMES, position);

        if( !g_dontsep)
        g_run=1;
        g_dontset=0;"
```

Example for video: Moving around #14/23

- We define the callback routine to be used when we want to slide the trackbar.
- A 32-bit integer will be pass through that routine and it is going to be the new trackbar position.
- In this call back we use the newly requested position in "g_capture.set(cv::CAP_PROP_POS_FRAMES, position);" to advance the video playback in the new position.

Example for video: Moving around #15/23

- We use the if () statement to set the program into single-step mode right after the next new frame that comes in.
- This is happening only when the callback starts by our click and not if the call was from the main function which sets the g_dontset.

Example for video: Moving around #16/23

- In the future, we are going to see more often the call "g_capture.set()" with the counterpart of it which is "g_capture.get()".
- With that routines we can configure lots of properties of the object we created via "cv::VideoCapture g_capture;".
- We can pass the argument "cv::CAP_PROP_POS_FRAMES", which mean that we would like to set the read position in units of frames.
- When the new video position is requested, it automatically will handle issues like the possibility that the requested frame is not actually the key-frame then it will start from the previous key-frame and it will fast-forward up to the requested frame without troubling us with the details.

Example for video: Moving around #17/23

- Lets see the next lines: ” int tmph=(int) g_capture.get(cv::CAP_PROP_FRAME_HEIGHT);”,” int frames=(int) g_capture.get(cv::CAP_PROP_FRAME_COUNT);”,” int tmpw=(int) g_capture.get(cv::CAP_PROP_FRAME_WIDTH);”,” cout << "The video has :" << frames << "frames of the dimensions (" <<tmpw <<","<< tmph <<")."<< endl;” .

Example for video: Moving around #18/23

- To determine the number of frames in the video and the width and height of the video images we use the g_capture.get().
- Then with the cout, we print these numbers.
- To calibrate the slider trackbar in the next step we will need the number of the frames in the video.

Example for video: Moving around #19/23

- We create the trackbar itself we use the function "cv::createTrackbar()" here is the full line of code: "
  cv::createTrackbar("Position", "video_example2",&g_slider_position,frames,onTrackbarSlide);".
- In the first place, we have "Position" is actually the label we gave to the trackbar.
- Then we specify the window we want to put the trackbar.
- After that, we give the variable that is going to bound to the trackbar.
- Then we have the number of frames in the video which is actually the max value of the trackbar.
- At last, we have the callback when the slider is moved it can also be NULL if we don't want one.

Example for video: Moving around #20/23

- Now we will explain the following code: “if(g_run !=0) {g_capture >> frame; if(frame.empty()) {break;} int current_position = (int) g_capture.get(cv::CAP_PROP_POS_FRAMES); g_dontsecv::setTrackbarpos("Position"," video_example2", current_position); c::imshow("video_example2",frame); g_run=1; }“

Example for video: Moving around #21/23

- This code is in a loop. While we read and display the video we can also get the current position in the video.
- We are setting the g_dontset because we want the next trackbar callback to not put us in single step mode.
- After that, we will invoke the trackbar callback to inform the position of the trackbar which had been displayed to us.
- And last we decrease g_run by 1 because that will keep us in single mode or it will allow the video to run depending on its previous state.

Example for video: Moving around #22/23

- Now we will see the last part of our example which is the key settings.

  “char key=(char) cv::waitKey(15); if(key== 't')

  {g_run=1; cout<<"Single step, run ="<< g_run<<endl; } if (key=='r')

  { g_run=-1; cout << "Run mode, run =" << g_run << endl;} if(key==27) {break;}”.

Example for video: Moving around #23/23

- In the last code, we told to the program that if we press "t" from the keyboard we can go into single step mode. Notice that here we set g_run into 1 which means that will allow reading of a single frame.
- After that, we told the program that if we press "r" from the keyboard we will go into continuous video mode. Notice that we set g_run into -1 and keeps decreasing and that leaves it negative for any conceivable video size.
- In the last " if" we say to the program that is we press "Esc" the program will terminate. Notice that in small programs we have omitted the cleaning up step if the window storage with the use of "cv::destroyWindow()".

OpenCV: Video

Source file: https://docs.opencv.org/2.4/_images/outputVideoInput.png, last time visited 5/29/2018.

Example for transformation (Gaussian) #1/6

- In this example, we are going to make an easy operation on the frames of the video as it plays.
- One easy operation is to make an image smooth, efficiently will reduce the information content of the image.
- To succeed that it needs to convolving it with a Gaussian or other kernel functions similar to Gaussian.

Example for transformation (Gaussian) #2/6

```
 //In this program  firstly we will load
//and then we will smooth  an image
// before  it;;s displayed on the screen.

// we will use the header file opencv.hpp
//Notice that this header it includes all OpenCV
//functions and is slow to compile.

#include <opencv2/opencv.hpp>

void img_example3( const cv::Mat  & image )
{


      //we are going to create a few windows
      // to show both input and output images in.

      cv::nameWindow("img_example3-in", cv::WINDOW_AUTOSIZE);

      cv::nameWindow("img_example3-out",cv::WINDOW_AUTOSIZE);
```

Example for transformation (Gaussian) #3/6

```
//We will create a window to show
    // inpout of the image.
    cv::imshow("img_example3",image);

    // Now we will create the image  to hold
    // the smoothed output.
    cv::Mat out;
    //To do smoothing we could use:
    // bliateralFilter(),GausianBlur(); , medianBlur(),blur();

    cv::GausianBlur( image, out, cv::Size(5,5),3,3);
    cv::GaussianBlur(out,out, cv::Size(5,5),3,3);

    // Now in the  output window
    // we are going to show the  smoothed image
    cv::imshow("img_example3", out);
            CV::waitKey(0);
}
```

Example for transformation (Gaussian) #4/6

- We will start from the second call since the first is similar to the previous example.
- With the next call, we can allocate another image structure.
- After we create the object we will instantiate an output matrix, out that will automatically relocate, resize and dellocate itself as necessary as it is used.

Example for transformation (Gaussian) #5/6

- Notice that we use consecutive calls to cv::GaussianBlur().
- In the first call, the image input is blurred by a convolution filter which is a 5X5 Gaussian and it writes out.
- We should give in odd numbers the size of the Gaussian kernel, for example, cv::Size(5,5) which is computed at the center pixel of that area.

Example for transformation (Gaussian) #6/6

- Out is used in the next call to cv::GausianBlur() as input and output while the temporary storage is assigned in this case for us.
- The result is that we have double- blurred image displayed.
- The routine after that waits for to push from a keyboard the input before cleaning up allocated data and terminating as it goes out of scope.

Example for a complex transformation (Gaussian) #1/9

- In this example are going to use a function that uses Gaussian blurring to down-sample an image.
- Notice that to form a scale space we need to down-sample the image a few times. This is also known as an image pyramid. We use that often in computer vision so we can handle the scales that are changing in an object or a scene that is observed.

Example for a complex transformation (Gaussian) #2/9

- According to signal processing and the Sampling Theorem of Nyquist-Shannon down-sampling, a signal is equivalent to convolving with a series of delta functions. Notice that in our example the signal is the image that we are sampling every other pixel.
- That kind of sampling is introducing us high frequencies into the resulting signal which in our case it is an image.

Example for a complex transformation (Gaussian) #3/9

- If we want to avoid all of this we need to run first a high-pass filter over the signal.
- We are doing this to band- limit the signal's frequencies to make sure that they are all bellow the sampling frequency.
- The function that allows us to do down-sampling and Gaussian blurring in OpenCV is: "cv::pyrDown()".

Example for a complex transformation (Gaussian) #4/9

```
 // In this example we are going to learn cv::pyrDown()
//With this function  we are going to create an image
// that is half the height and width of the input image.
//We are going to include the general header file
// of OpenCV which is opencv.hpp.

#include <opencv2/opencv.hpp>

int main(int argc, char ** argv )
{
        cv::Mat f_image, s_image;
        cv::namedWindow("transformation1",cv::WINDOW_AUTOSIZE);
        cv::namedWindow("transformation2", cv::WINDOW_AUTOSIZE);
        img= cv::imread(argv[1]);
        cv::imshow("traansformation1", f_image);
        cv::pyrDown(f_image,s_image);
        cv::imshow("transformation2",s_image);
        cv::waitKey(0);
return 0;
}
```

Example for a complex transformation (Gaussian) #5/9

- In the next example of complex transformation, we are going to see about Canny edge detector.
- To do that with OpenCV we are going to use the function "cv::Canny();".
- The full size of the input image will be generated by edge detector into an image.

Example for a complex transformation (Gaussian) #6/9

- To do that we need only a single channel image to write to.
- To solve this we convert it to a gray-scale single channel image.
- We are going to use cv::cvtColor() with flag for converting blue,Green,Red images to grays-cale, "cv::COLOR_BRG2GRAY".

Example for a complex transformation (Gaussian) #7/9

```
      // In this example the canny edge detector
//writes its output to a single channel
//the grey scale.

#include <opencv2/opencv.hpp>

int main (int argc, char** argv)
{
      cv::Ma image_rgb, image_gray, image_canny;

      cv::nameWindow("Gray", cv::WINDOW_AUTOSIZE);
      cv::nameWindow("Canny", cb::WINDOW_AUTOSIZE);

      image_rgb=cv::imred(argv[1]);
      cv::cvtColor(image_rgb,image_gray,cv::COLOR_BGR2GRAY);
      cv::imshow("Gray",image_gray);
      cv::Canny(image_gray,image_canny, 10,100,3,true);
      cv::imshow("Canny", image_canny);
      cv::waitKey(0);
}
```

Example for a complex transformation (Gaussian) #8/9

- We can combine in a simple image pipeline both the Canny subroutine and the pyramid down operator.

```
cv::cvtColor(image_rgb, image_gray, cv::BGR2GRAY);
cv::pyrDown(image_gray, image_pyr);
cv::pyrDown(image_pyr, image_pyr2);
cv::Canny(image_pyr2 , image_canny, 10, 100, 3, true);
```

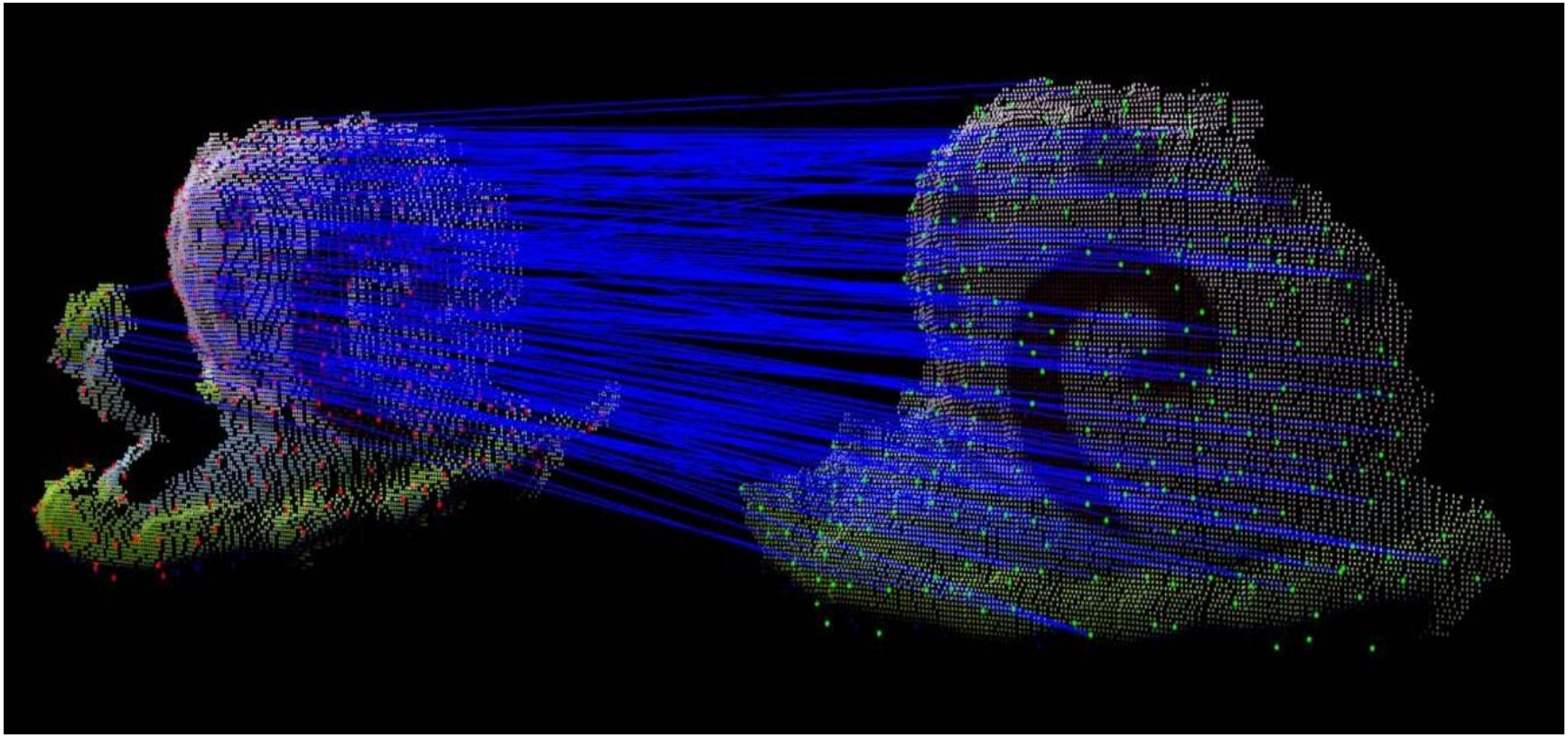
Example for a complex transformation (Gaussian) #9/9

```
//This example is connecting to the previous example
//and it is made for  getting and setting pixels.

int x=15,y=33;

cv::Vec3b intensity =image_rgb.at< cv::Vec3b>(y,x);
uchar blue =intensity[0];
uchar green=intensity[1];
uchar red=intensity[2];
std::cout << "At (x,y)= (" << x<<"," <<y<<"): (RGB)= (" <<(unsigned int)
     red<<","<<(unsigned int) green<<","<<(unsigned int) blue<< ")" << std:: endl;
std::cout<<"Greay pixel there is :" <<(unsigned int)image_gray.at<uchar>(y,x)<<std::endl;
x= x/4;
y=y/4;
std::cout <<"Pyramid pixel is :"<<(unsigned int) image_pyr.at<uchar>(y,x)<<std::endl;
// here we set the canny pixel to 128
image_cny.at<uchar>(x,y)=128;
```

Computer vision transformation



Source file: http://imagelab.ing.unimore.it/imagelab2015/_images/3dmatching.jpg, last time visited 5/29/2018.

Example: Input from a camera #1/4

- In the world of computers "VISION" stands for many things.
- We can analyze frames that are still which are loaded from elsewhere.
- We also can analyze videos that are being read by a disk.
- In the most advanced situations, we can work with real-time data. That data can be obtained by camera devices.

Example: Input from a camera #2/4

- The OpenCV library gives us tools to handle real-time data from a camera.
- Reading from a disc or a camera can be done in the same way which is "cv::VideoCapture object_name".
- If we want to read from a disk we need to give the path of the file "Path/filename", if we want to read from a camera you give a camera ID number if have one camera that is connected to the system we use 0.

Example: Input from a camera #3/4

- The value -1 is the default value and refers to "pick one".
- This works well when we have only one camera.
- In this example, we will see how to capture a video file from a camera or a file.

Example: Input from a camera #4/4

```
//the object can load videos
// both from a file or a camera.
#include <iostream>
#include<opencv2/opencv.hpp>

int main( int argc , char** argv)
{               cv::nameWindow("Capture_camera", cv::WINDOW_AUTOSIZE);
                cv::VideoCapture capture;
                //open the camera
                if(argc==1)
                { capture.open(0); }
                else
                { capture.open(argv[1]); }
      // cheack the success
      if(!capture.isOpened())
      { std::cerr<< "Failed to open  capature."<< std::endl;
                return -1; }
      // this is the main core of the program
      …
}
```

Example: How to write to an Avi file #1/4

- We write videos in Avi files because we want to record the streaming output or disparate captured images to an output video stream.
- We can create a capture device that allows us to grab frames from a video stream once at a time.
- We can also create a writer device that will allow us to place frames in a video. The object we are going to use is: "cv::VideoWriter" and finally we use the "cv::VideoWriter.release()" method.

Example: How to write to an Avi file #2/4

```
//In this program we are going to read
//a color in the video and
//write out the lo-polar- transformed video.
#include <iostream>
#include <opencv2/opencv.hpp>
int main( int argc, char* argv[])
{
      cv::namedWindow("Example_avi", cv::WINDOW_AUTOSIZE);
      cv::namedWindow("Log_polar", cv::WINDOW_AUTOSIZE);
      //if we want to use a camera to capture
      // we must give a camera id which must be declared as integer(int).
      cv::VideoCapture capture(argv[1]);
      double fps=caprure.get(cv::CAP_PROP_FPS);
      cv::Size size( (int)capture.get(cv::CAP_PROP_FRAME_WIDTH),
      (int) capture.get(cv::CAP_PROP_FRAME_HEIGHT) );
      cv::VideoWriter writer;
      writer.open(argv[2], CV_FOURCC('M','J','P','G'), fps ,size);
      cv::Mat logpolar_frame, bgr_frame;
```

Example: How to write to an Avi file #3/4

```
for(;;)
    {
            capture >> bgr_frame;
            if(bgr_frame.empty())
            {break;}
           cv::imshow("Example_avi",bgr_frame);
           //1.inpute color frame 2. output log-polar frame 3.centerpoint
           //for transformation of log-polar 4.x 5.y 6.scale parameter Magnitude
           //7. Fill  outliers with 0.
           cv::logpolar(bgr_frame,logpolar_frame,cv::Point2f(bgr_frame.cols/2,
           bgr_frame.rows/2),40,cv::WARP_FILL_OUTLIERS);
           cv::imshow("Log_polar",logpolar_frame);
           writer<<logpolar_frame;
           char key= cv::waitKey(10);
           if(key==27) //27=Esc
           {break;}
    }
    capture.release();
}
```

Example: How to write to an Avi file #4/4

- In this program, we opened a video and read some properties of it such us frames per second, the height and the width of an image.
- Then we read frame by frame the video from "cv::VideoReader object".
- We convert the frame to log-polar format.
- We write the long- polar frames to a new video file one at a time till the user quits by pressing Esc or till none left to write.

OpenCV basics with python: Reading and writing an image file #1/5

- In python we have the following OpenCV functions: "imread()" and "imwrite()".
- This functions supports various formats of still images.
- In many systems, the formats may vary but all the systems have a standard format that format is BMP.

OpenCV basics with python: Reading and writing an image file #2/5

- To use OpenCV functions in python we need to import cv2.

```
import cv2

image= cv2.imread('image1.png')
cv2.imwrite('image1.jpg',image)
```

OpenCV basics with python: Reading and writing an image file #3/5

- The imread() function returns an image in Blue Green Red(BGR) color format. Notice that if the file uses a grayscale format it will do the same function.
- The color that BGR represents is the same with RGB (Red Green Blue) the only difference is that the byte order is reversed.

OpenCV basics with python: Reading and writing an image file #4/5

- In imread() we can specify the mode we want to use so it can be : CV_LOAD_IMAGE_COLOR for BGR, CV_LOAD_IMAGE_GRAYSCALE for grayscale and CV_LOAD_IMAGE_UNCHANGED.
- It does not matter the mode we have because imread() discards any alpha channel (transparency).

OpenCV basics with python: Reading and writing an image file #5/5

- The imwrite() function needs an image to be in grayscale or BGR format with a number of bits for each channel that the output format supports.

```
#Notice that a bmp image requires 8bits per channel
#png requiews 8 or 16 bits per channel
import cv2
grayimage=cb2.imread('image2.png',
    cv2.CV_LOAD_IMAGE_GRAYSCALE)
cv2.imwrite('iamge2gray.png', grayimage)
```

OpenCV basics with python: converting between raw bytes and image #1/6

Generally, we know that a byte is an integer and its range is from 0 to 255. In the real-time graphics applications, a pixel is represented by one byte per channel but there are other representations which are possible too.

OpenCV basics with python: converting between raw bytes and image #2/6

- Actually when we talk about an OpenCV image we refer to 2D or 3D array and the type of it is numpy.array.
- For example, an 8-bit grayscale image is actually a 2D array that contains byte values.
- Another example is a 24-bit BGR image which is a 3D array that also contains byte values.

OpenCV basics with python: converting between raw bytes and image #3/6

- To access that values we use expressions such as image[0,0,0] or image[0,0].
- In the first index, we have the y coordinate or row 0 which is being on the top.
- As for the second is the x coordinate or column 0 which is being left most.
- If we have a third index it will represent the color channel.

OpenCV basics with python: converting between raw bytes and image #4/6

- Example: if we have an 8-bit grayscale image with a white pixel in the upper left corner we will have image[0,0] which is 255.
- On the other hand for a 24-bit BGR image with a blue pixel image[0,0] is [255,0,0].
- An alternative way to express image[0,0]=128 in python is image.setitem((0,0),128). We use the second method mostly to a single pixel operation.

OpenCV basics with python: converting between raw bytes and image #5/6

- We can cast an image that has 8 bit per channel to a standard python "byteArray", this is one-dimensional and it will be like this: "byteArray = bytearray(image) ".
- The "byteArray" has bytes in a specific order.
- After we cast then we reshape it to get a "numpy.array". Here is how it will go : "grayImage = numpy.array(grayByteArray).reshape(height, width) ","bgrImage = numpy.array(bgrByteArray).reshape(height, width, 3)".

OpenCV basics with python: converting between raw bytes and image #6/6

```
#convert bytearray
#This script creates a pair of randomly
#generated images ins script's directory
import os
import numpy
import cv2

#os.urandom() is a function that generates random raw bytes
#we then covert them to numpy array using reshape.
randomByteArray=byteArray(os.urandom(120000))

flatNumpyArray = numpy.array(randomByteArray)

#To make a 400X300 grayscale image
#we need to convert the array.
grayImage=flatNumpyArray.reshape(300,400)
cv2.imwrite('RandomGray.png',grayImage)


#now we are going to do the same for a color image 400X100
bgrImage=flatNumpyArray.reshape(100,400,3)
cv2.imwrite('RandomColor.png', bgrImage)
```

OpenCV basics with python: Reading and writing a video file #1/3

- In python, OpenCV has classes that support various video file formats. That class is: "VideoCapture" and "VideoWriter".
- The supported formats vary in every system but all of them should always include AVI.
- In VideoCapture class we have the method "read()". This method may be polled for new frames until reaching the end of the video file.

OpenCV basics with python: Reading and writing a video file #2/3

- In the class VideoWriter, we have the method "write()" that appends the image to the file in the VideoWriter.
- We need to be cautious with the arguments in VideoWriter class constructor.
- The file name of the video must be specified because any preexisting with the same name will be overwritten.
- Notice that we must specify the codec too.

OpenCV basics with python: Reading and writing a video file #3/3

```
 #In this Script we will
#read frames from an Avi file
#then it writes them to another
#avi file with YUV encoding.

import cv2

videoCapture=cv2.VideoCapture('Inputvideo.avi')
fps=videoCapture.get(cv2.cv.CV_CAP_PROP_FPS)
size=(int(videoCapture.get(cv2.cv.CV_CAP_PROP_FRAME_WIDTH)),
      int(videoCapture.get(cv2.cv.CV_CAP_PROP_FRAME_HEIGHT)))

videoWriter=cv2.VideoWriter( 'Outputvideo.avi',cv2.cv.CV_FOURCC('I','4','2','0'),fps,size)

succsess,frame=videoCapture.read()
#This Loop goes till there are no frames left.
while success:

            videoWirter.write(frame)
      succsess,frame=videoCapture.read()
```

OpenCV basics with python: Capturing camera frames #1/5

It is the same as reading and writing from a file the only difference is that we need to pass the video camera's index instead of a video file name.

OpenCV basics with python: Capturing camera frames #2/5

```
#capturing camera frames
import cv2

cameraCapture=cv2.VideoCapture(0)
fps =30
size=(int(cameraCapture.get(cv2.cv.CV_CAP_PROP_FRAME_WIDTH),
      int(cameraCapture.get(cv2.cv.CV_CAP_PROP_FRAME_HEIGHT)))

videoWriter =
    cv2.VideoWriter('Ourputvideo.avi'cv2.cv.CV_FOURCC('I','4','2','0'),fps,size)

success, frame= cameraCapture.read()
numFramesRemaining=10 *fps-1
while success and numFramesRemaining>0:
    videoWriter.write(frame)
    success, frame = cameraCapture.read()
    numFramesRemaining -=1
```

OpenCV basics with python: Capturing camera frames #3/5

- The get() method does not return an accurate value for the camera's frame rate it returns always 0.
- We need to make an assumption about the frame rate or measure it using a camera. If we do that we will create an appropriate VideoWriter class for the camera.

OpenCV basics with python: Capturing camera frames #4/5

- OpenCV does not provide any way of querying the number of the cameras or the properties of the cameras.
- If we are going to construct a VideoCapture class with an invalid index it will not yield any frames and the read() method will return (false, none).

OpenCV basics with python: Capturing camera frames #5/5

- **To synchronize a set of cameras or a multi-head camera we use the methods grab() and retrieve().**

```
#for a set of cameras we use

success0 = cameraCapture0.grab()
success1 = cameraCapture1.grab()
if success0 and success1:
    frame0 = cameraCapture0.retrieve()
    frame1 = cameraCapture1.retrieve()

#For a multi head camera we use

success = multiHeadCameraCapture.grab()
if success:
    frame0 = multiHeadCameraCapture.retrieve(channel = 0)
    frame1 = multiHeadCameraCapture.retrieve(channel = 1)
```

OpenCV basics with python: Displaying camera frames in a window #1/2

- In this example we will see how:
    1. To name a window
    2. To create and redrawn
    3. To destroy a Window
    4. Mouse input

OpenCV basics with python: Displaying camera frames in a window #2/2

```
import cv2

clicked=False
def onMouse(event,x,y,flags, param);

global clicked

if event == cv2.cv.CV_EVENT_LBUTTONUP:
      clicked =True

cameraCapture           = cv2.VideoCapture(0)
cv2.namedWindow('Example')
cv2.setMouseCallback('Example', onMouse)

print 'Showing camera feed. Click on the window or press any key to stop.'
success,frame =cameraCapture.read()
while success and cv2.waitKey(1) == -1 and not clicked:
      cv2.imshow('Example',frame)
      success,frame =cameraCapture.read()

cv2.destroyWindow('Example')
```

Module B

In this lecture we will see some advanced topics of OpenCV, more specifically we are going to see:

- Feature extraction
- Background subtraction
- Object Detection

Feature extraction #1/11

- There is an image pattern that is helping us to describe what we see in the image this image pattern we call it "Image feature".
- For example, if we have an image of a cat it's feature could be the cat eye.
- In computer vision, the features are used for transforming visual information into the vector space.

Feature extraction #2/11

- Transforming a visual information to the vector space is allowing us to perform mathematical operations on them.
- This gives us a way to find similarities between images or objects on the images.
- To do that actions there are two ways first way is with neural nets and the second is image descriptors.

Feature extraction #3/11

- There are plenty algorithms for feature extraction, most of that algorithms are based on image gradient. Some of the most popular algorithms for feature extraction are :
  - ORB
  - SIFT
  - BRIEF
  - SURF
- In this presentation, we will say a few things only about SURF.

Feature extraction #4/11

- The Speeded –Up Robust Features (SURF) we can say that is the evolution of SIFT feature.
- SURF originally proposed by Bay et al in 2006.
- The goal of SURF creators was to replace SIFT features with more computationally efficient techniques that could give similar or better performance in primarily recognition tasks.

Feature extraction #5/11

- Indeed the result was not only faster to compute but also in many cases the simpler nature results in greater robustness to change in operation or lightning that is observed with SIFT features.
- The SURF feature is depended on computations that can be greatly accelerated.
- The technique we use for that purpose is "Integral image technique".

Feature extraction #6/11

- In SURF we need to define a key point in terms of the local Hessian at a given point.
- Hessian is a matrix of second- order derivatives.

- The best implementation of SURF in OpenCV is using cv::Feature2D interface we re going to see an example in C++.

Feature extraction #7/11

```
class cv::xfeatures2d::SURF : public cv::Feature2D{
    static Ptr<SURF> create (
            double hessianThreshold=100,
    // This is the number of pyramid octaves
            int n0ctaves =4,
     //This is the number of images in each octave
            int n0ctaveLayers=3,
    //false: 64-element, true : 128-element descriptors
    bool extented = fasle ,
    //true : do not compute orientations .W/out is much faster.
    bool upright = false);
            // descriptor size 128 or 64
            int descriptorSize() const;
    // type od the descriptor CV_32F
    int descriptorType() const;
    …

};
typedef SURF SurfFeatureDetector;
typedef SURF SurfDescriporExtractor;
```

Feature extraction #8/11

- In the cv::xfeatures2d::SURF : object the constructor method create() has 5 arguments.
- The first argument "hessianThreshold" sets the value of the threshold for the determinant of the Hessian that is needed as a local extremum to be considered as key-point.
- A typical value for the constructor is 1500 but we assigned the constructor's value by default 100.

Feature extraction #9/11

- The extended parameter is telling the feature extractor to use the extended feature set.
- The parameter upright indicates that orientations should be treated as "vertical" we also can call that "U-SURF".
- The final arguments n0ctaves and n0ctaveLayers are analogous to corresponding arguments for "cv::xfeatures2d::SIFT()".

Feature extraction #10/11

- The n0ctaves argument determines the number of doublings of scale that will be searched for key points.
- For each octave, a few kernels will be evaluated.
- The default value for n0ctaveLayers is 3 but in new research has proved that it is increasing it to 4.

Feature extraction #11/11

- The descriptorSize() and descriptorType() methods are returning the numbers of elements in descriptor vector and the type of the descriptor vector.
- Some extra functions of "cv::xfeatures2d::SURF" are methods that set and retrieves the parameters of the algorithm on the fly.

Background subtraction #1/7

- Background subtraction is a key image-processing operation for many applications mostly for video security.
- If we want to perform background subtraction we need to learn a model of the background.
- This model is compared to the current image and then the background parts are subtracted away. That objects are presumably new foreground objects.

Background subtraction #2/7

- For simple scenes, the background modeling methods are suffering from an assumption which is often violated: that the behavior of every pixel in the image is statistically independent of the behavior of others.
- A solution to make the surrounding pixels into account we need to learn a model which is multipart.

Background subtraction #3/7

- To define background and foreground we need a high-level scene in which we define multiple levels between foreground and background states.
- Also, we need a timing-based method of slowly relegating unmoving foreground patches to background patches.
- Finally, we have to detect and create a new model when we have a global change in the scene.

Background subtraction #4/7

- A simple background subtraction method is that subtracts one frame from another and then label any difference that is big enough in the foreground.
- We create a process that tends to catch the edges of moving objects.

Background subtraction #5/7

An example: if we have three single-channel images: frametime1, frametime2, and foreground. The image with frametime1 is filled with an older grayscale image and frametime2 with the current grayscale image.

Background subtraction #6/7

- In the following code, we are going to detect the absolute value of foreground differences in frameForeground.

```
cv::absdiff(
//It is the first input array.
frametime1,
//It is the second input array
frametime2,
//It is the result array.
frameForeground);
```

Background subtraction #7/7

- **Pixel values every time exhibit fluctuations and noise so we need to ignore set "set to 0' small differences "say, less than 15" and then mark the rest as big differences "set to 255".**

```
cv::threshold(
//input image
frameForefround,
//result image
frameForefround,
//threshold value
15,
//max value for upward operations
255,
//Type of threshold to use
cv::THRESH_BINARY
);
```

Object Detection #1/11

- Object Detection is a process that determines if an image contains any particular object also the localization of the object we want to find in pixel space.
- We will see some methods, which is helping us to detect objects.

Object Detection #2/11

- The first method is called " Cascade classifier".
- This method generalizes the "Viola and Jones" algorithm, for face detection.
- The second method is called "Soft cascade".
- This evolved algorithm uses a more robust classification than the " Cascade classifier".

Object Detection #3/11

- These methods can be used for object detection not only for faces but for other object classes too.
- In conclusion objects with rich texture and grid structure are well respond in these methods.
- Notice that these methods involve other stages that inset an input learning or post-process the input of "The learning algorithm".

Object Detection #4/11

- The cascade classifier is a tree-based technique. This technique was built on a concept, the name of the name of that concept is "boosted rejection cascade".
- A technique for face detection first developed by "Paul Viola and Michael Jones".
- That technique is known as the "Viola-Jones detector".

Object Detection #5/11

- The Viola-Jones detector operates in two layers.
- We call the first layer "feature detector". In this layer, we encapsulate and modularize the feature computation.
- In the last layer we have the cascade boosted. It uses differences and sums over rectangular regions of the computed features.

Object Detection #6/11

- By default, the classifier uses Haar-like features. Haar wavelet is the first known wavelet basis and it was proposed by Alfred Haar in 1909.
- There are two feature sets supported. These sets include both the Haar wavelet features and the alternative feature "LBP".

Object Detection #7/11

- Now we are going to give an example of face detection. We will use detectAnddraw() to detect faces and draw their locations in different – colored rectangles on the image.
- In this example notice that a previously trained classifier cascade has been loaded.
- In the end, the memory for detected faces is going to be created.

Object Detection #8/11

```
 //In this example we are going
//to  detect and draw faces
// faces are the objects we detect in this example.
#include <opencv2/opencv.hpp>
#include <fstream>
#include <iostream>
//input image, preload classifier,  resize image by ..
void detectAndDraw( cv::Mat image, cv::Ptr<cv::CascadeClassifier> classifier, double scale=1.3)
{ enum {CYAN,BLUE,AQUA,  GREEN};

                static cv::Scalar colors[]=
                {
                                        cv::Scalar(0,255,255),
                                        cv::Scalar(0,0,255),
                                        cv::Scalar(0,128,255),
                                        cv::Scalar(0,255,0),
                };
```

Object Detection #9/11

```
//Preparation of  the image
    cv::Mat gray(image.size(),CV_8UC1);
    cv::May small_img(cvSize(cvRound(image.cols/scale),
    cvRound(image.rows/scale)),CV_8UC1);
    cv::cvtcolor(image,gray,cv::BGR2GRAY);
    cv::resize(gray, small_img, cv::INTER_LINEAR);
    cv:: equalizeHist(small_img,small_img);
    //This is how we detect objects is there are any.
    vector<cv::Rect> objects;
// 1) The image we input, 2)variable for the resu;ts, 3) scalefactor,
//4) the min number of neighbors, 5) An old format for cascades only
//6) Throw away detections smaller than this.
    classifier ->
    detectMultiScale(small_img,objects,1.1,2,cv::HAAR_DO_CANNY_PRUNING,cv::Size(30,30);
//Found and draw
for (vector <cv::rect>:: iterator r=objects.begin(); r!=objects.end; ++r)
{
    Rect r_ =(*r)*scale;
    cv:: rectangle(image,r_,colors[i%4]);
}
}
```

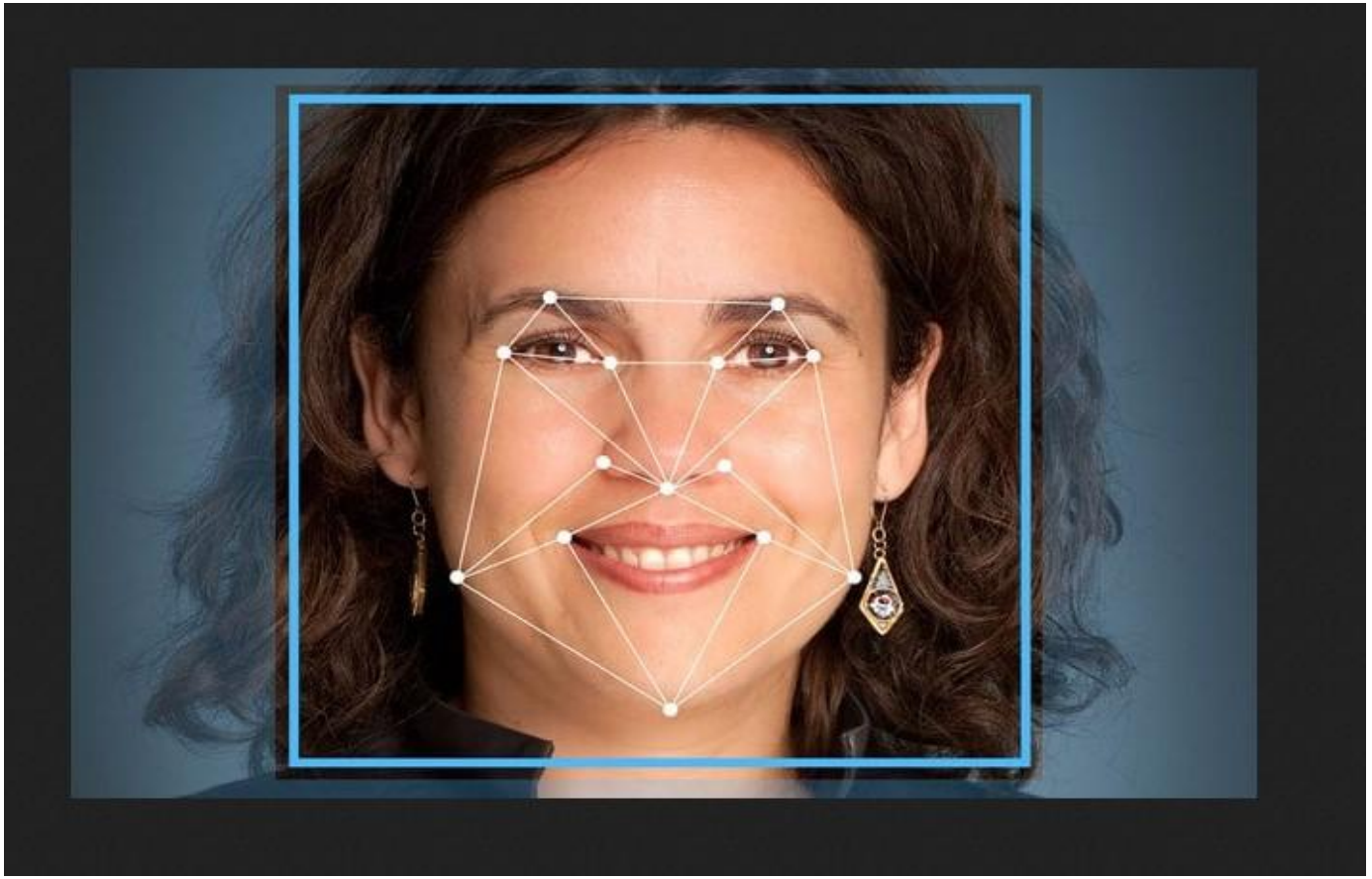
Object Detection #10/11

- In this program the detectAndDraw() function has a static vector of colors colors[].
- These colors are allowed to be indexed to draw the faces we found in deferent colors.
- Since this classifier works on grayscale images, we need to convert the BGR image into grayscale via "cv::cvtColor()", after that optionally resized via "cv::resize()".

Object Detection #11/11

- After that, we have histogram equalization via "cv::equalizeHist()", which speeds out the values of brightness.
- This is important because the integral image features are based on differences of rectangle regions this means that if the histogram is not balanced, the differences might be skewed by overall lighting or exposure of the test images.
- The real object detection is taking place above the loop and the loop steps into the found object rectangle regions and then draws them with the colors we declare using the "cv::rectangle()".

Face Detection



Source file: https://d3l69s690g8302.cloudfront.net/wp-content/uploads/2016/11/18123046/google-glass-facial-recognition.jpg, last time visited 5/29/2018.

Conclusion

OpenCV can be used in various domains such as medicine, robotics, secure systems and much more. It is essential to say that it has great capability with Windows and Linux. You can start from the basics and then move to more advance subjects. In this presentation we made a simple reference about computer vision and OpenCV. If you are willing to learn more about you can check the sources of this presentations to find more materials to study.

Sources

- **Computer vision From Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Computer_vision, last time visited on 5/12/2018.**
- **BOOK: Learning OpenCV 3 COMPUTER VISION in C++ WITH THE OPENCV LIBRARY by Adrian Kaehler & Gary Bradski.**
- **Computer vision from Osprey Informatics: http://www.ospreyinformatics.com/computer-vision/ , last time visited on 5/12/2018.**
- **MASARYK UNIVERSITY FACULTY OF INFORMATICS, Event Detection in video , from Master's Thesis Filip Nálepa : https://is.muni.cz/th/359760/fi_m/thesis.pdf, last time visited on 5/12/2018.**
- **OpenCV From Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/OpenCV , last time visited on 5/12/2018.**
- **BOOK: Joseph Howse - OpenCV Computer Vision with Python – 2013**
- **Feature extraction and similar image search with OpenCV for newbies, Author Andrey Nikishaev : https://medium.com/machine-learning-world/feature-extraction-and-similar-image-search-with-opencv-for-newbies-3c59796bf774 , last time visited on 5/12/2018.**
- **Learn OpenCV ubuntu installation: https://www.learnopencv.com/install-opencv3-on-ubuntu/, last time visited on 5/12/2018.**