A Tutorial on OpenCV

PPT: Chihwei Lin

Speaker: Chun-Jui Lai

Outline

- Introduction
- Environment Settings
- Modules In OpenCV
- Examples
  - Image
  - video

Introduction

- What is OpenCV?
  -- The latest version is 3.0

OpenCV is a Open Source Computer Vision Library. It consists of a set of C functions and a few C + + classes, many common algorithms for image processing and computer vision

Download OpenCV

- The Newest update is version 3.0
  --You can download the release version for Unix,
  ios, Windows or Android
  --It has C++, C, Python and Java interfaces
- OpenCV is free for non-commercial and commercial applications

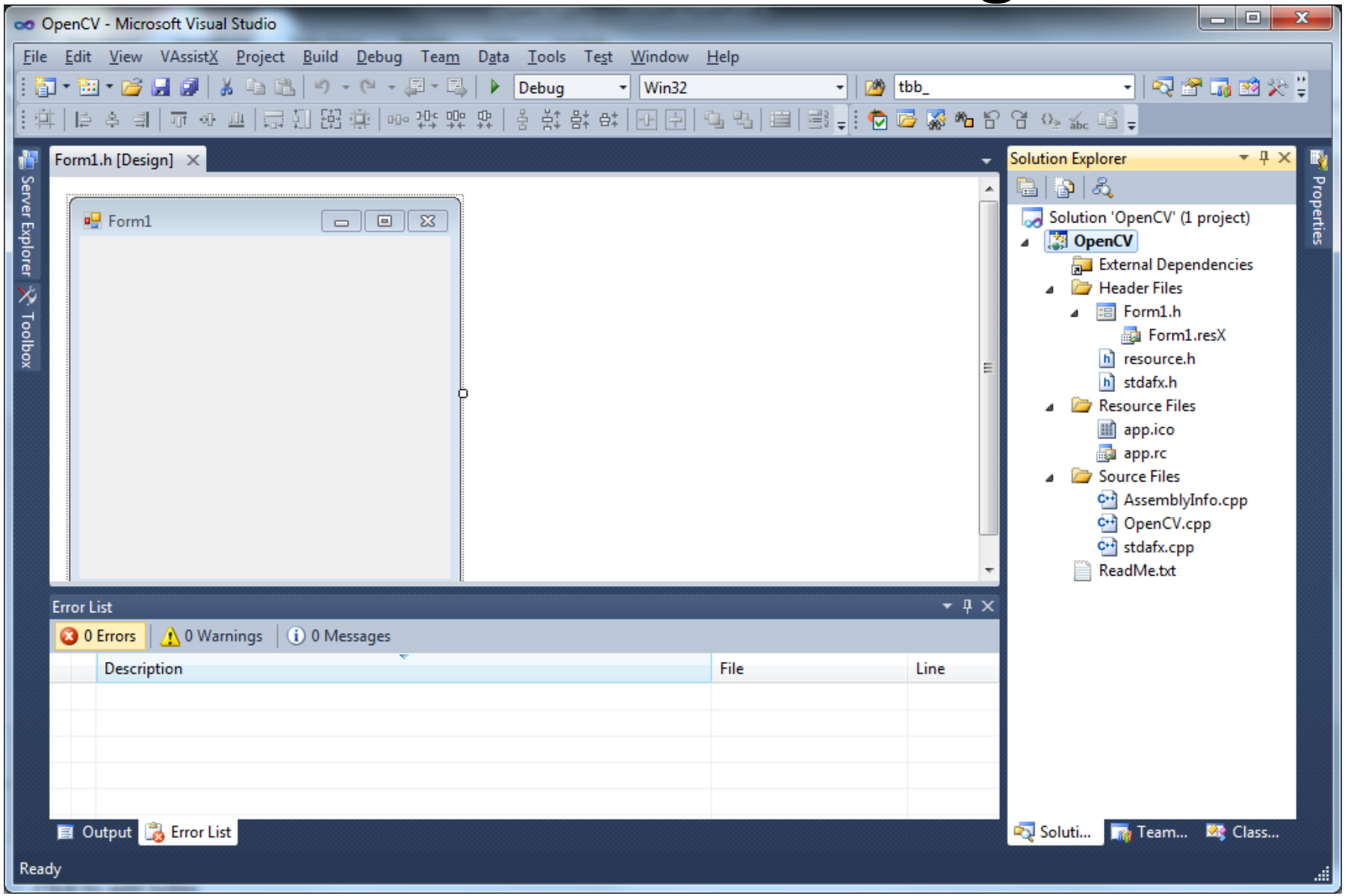
Applications

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and Recognition
- Stereopsis Stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Motion tracking Augmented reality

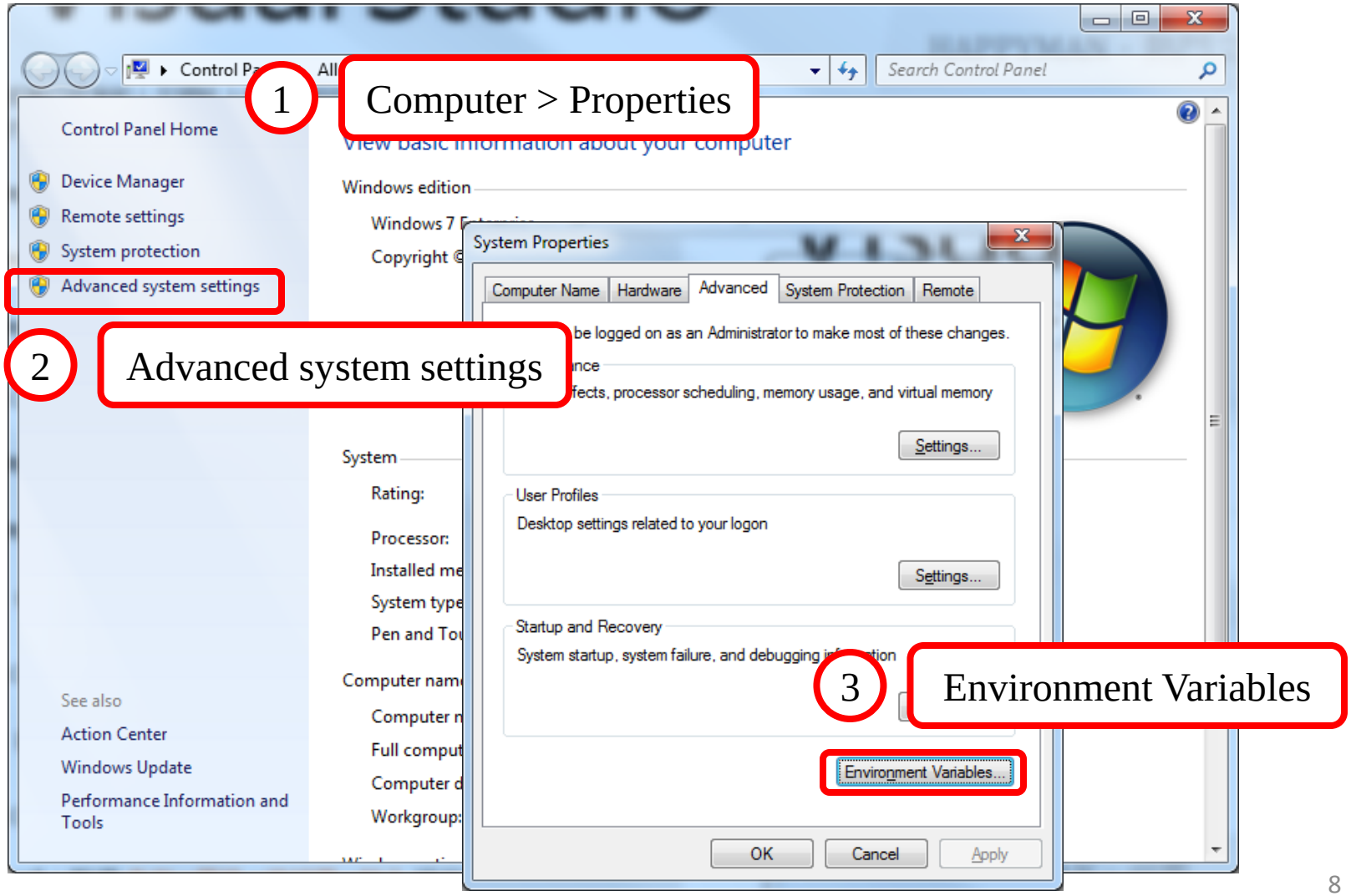
Machine Learning Library

- Boosting (meta-algorithm)
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)

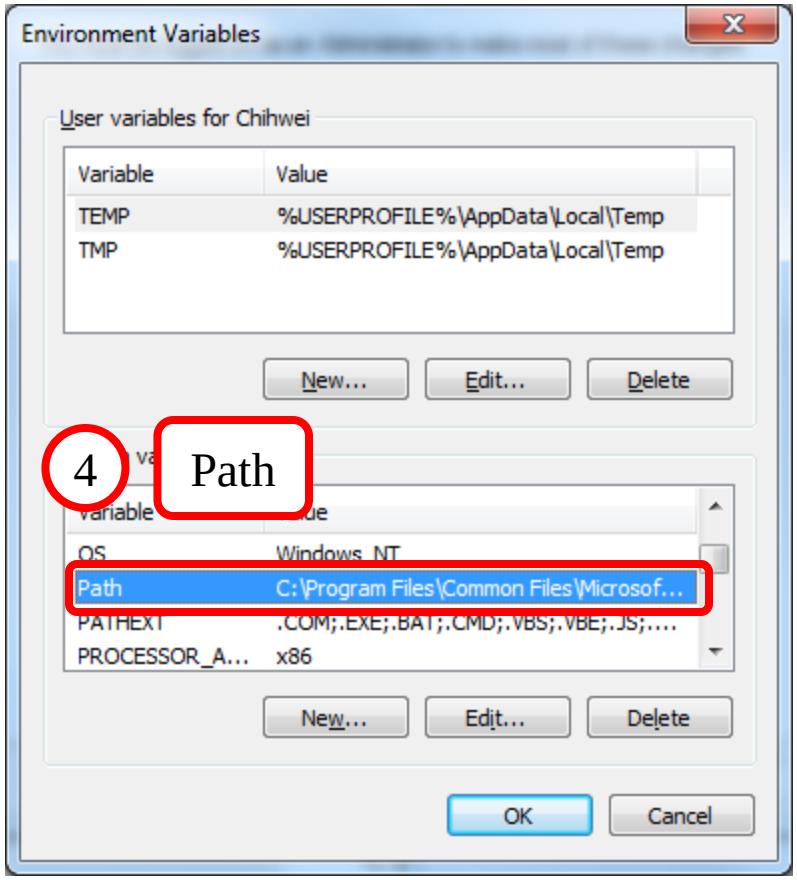
Environment Settings

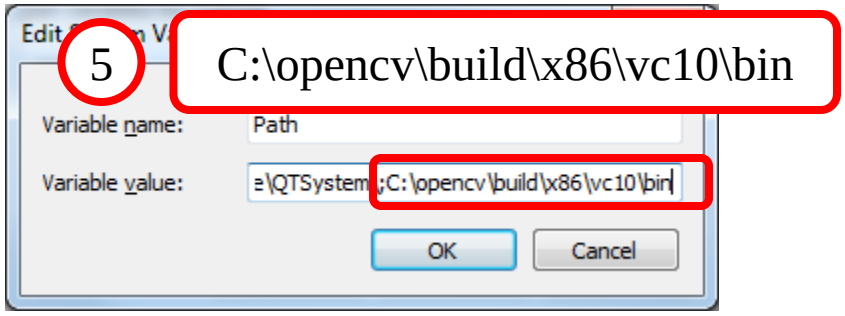


Environment Settings

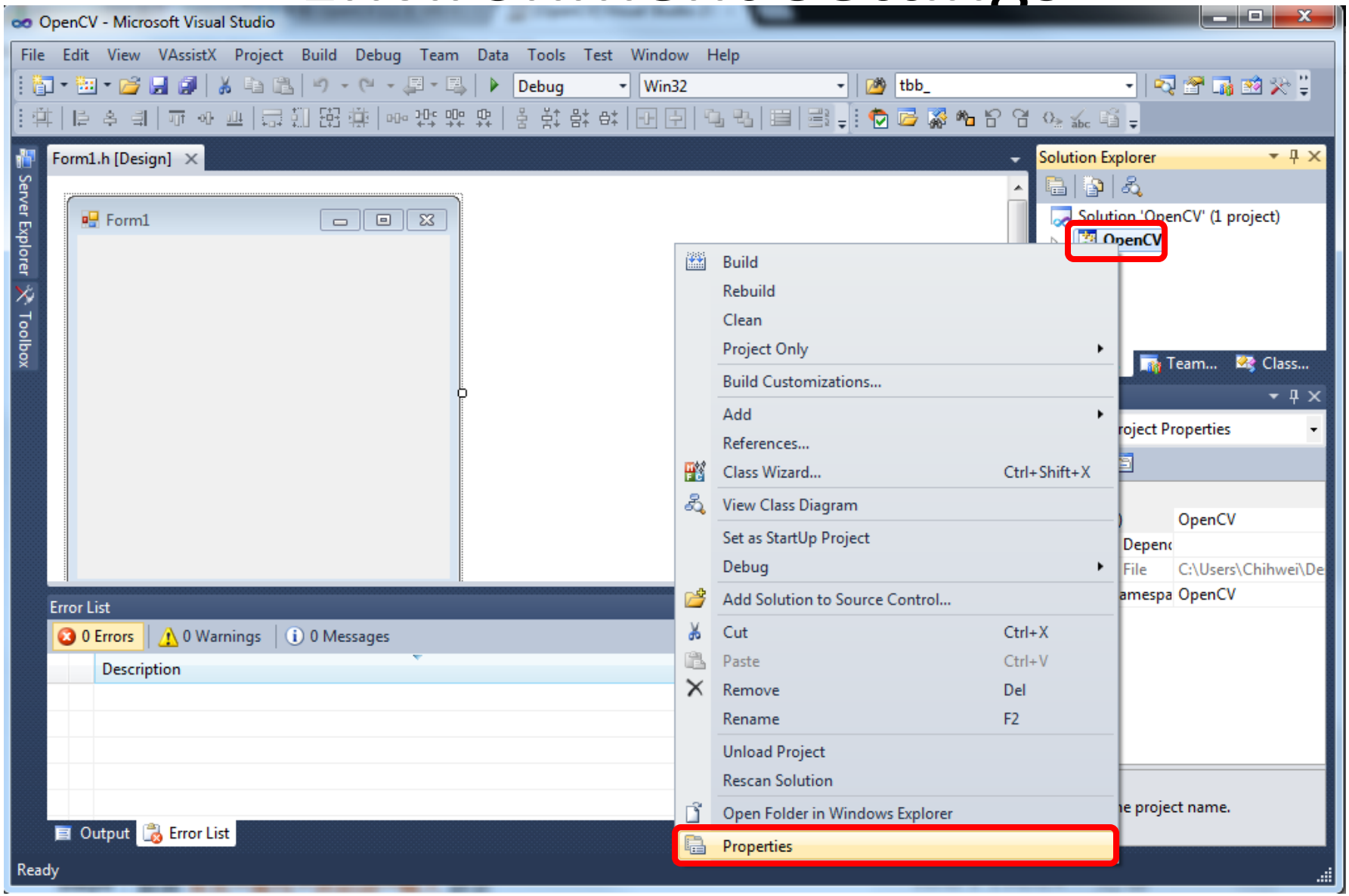


Environment Settings

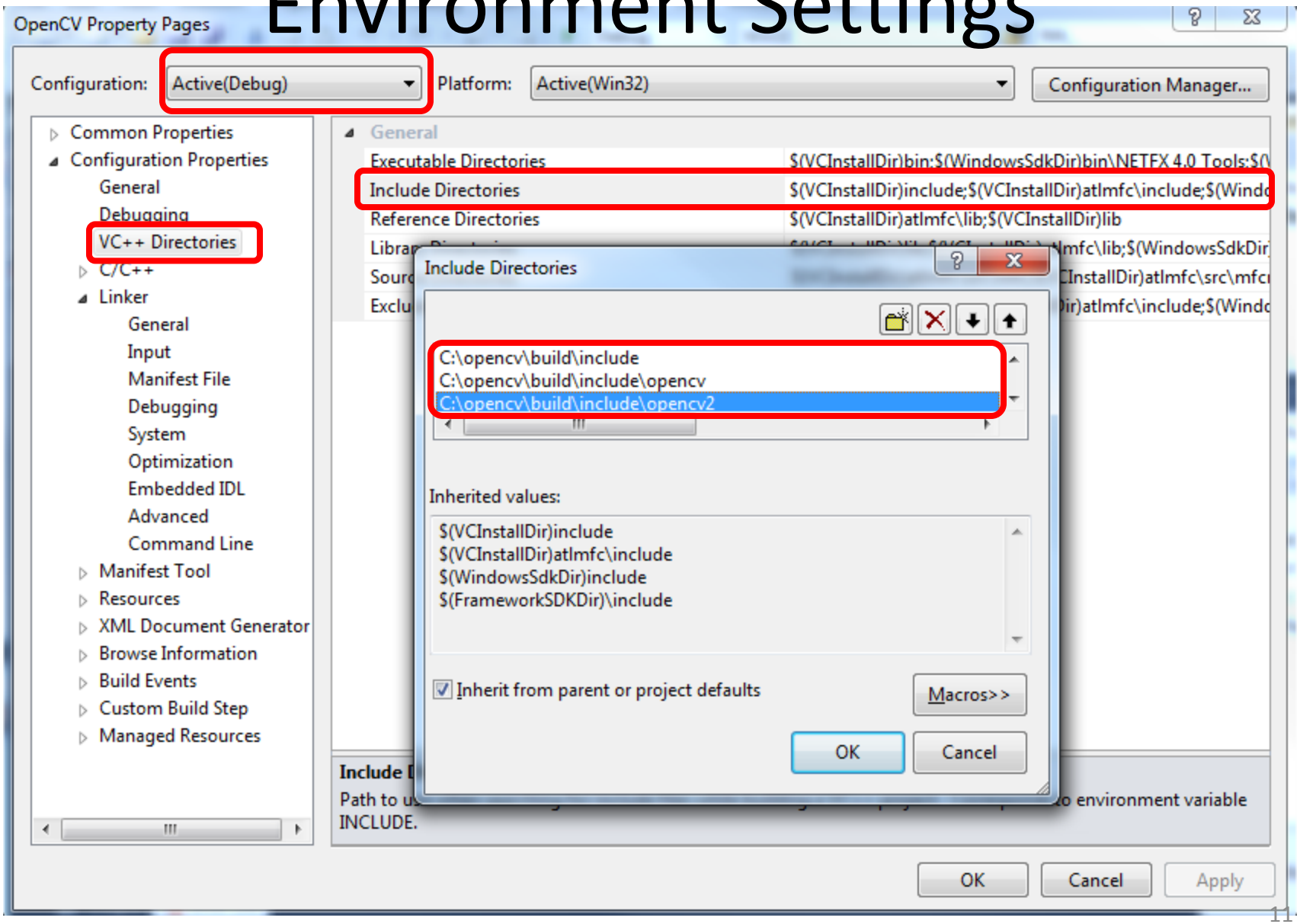




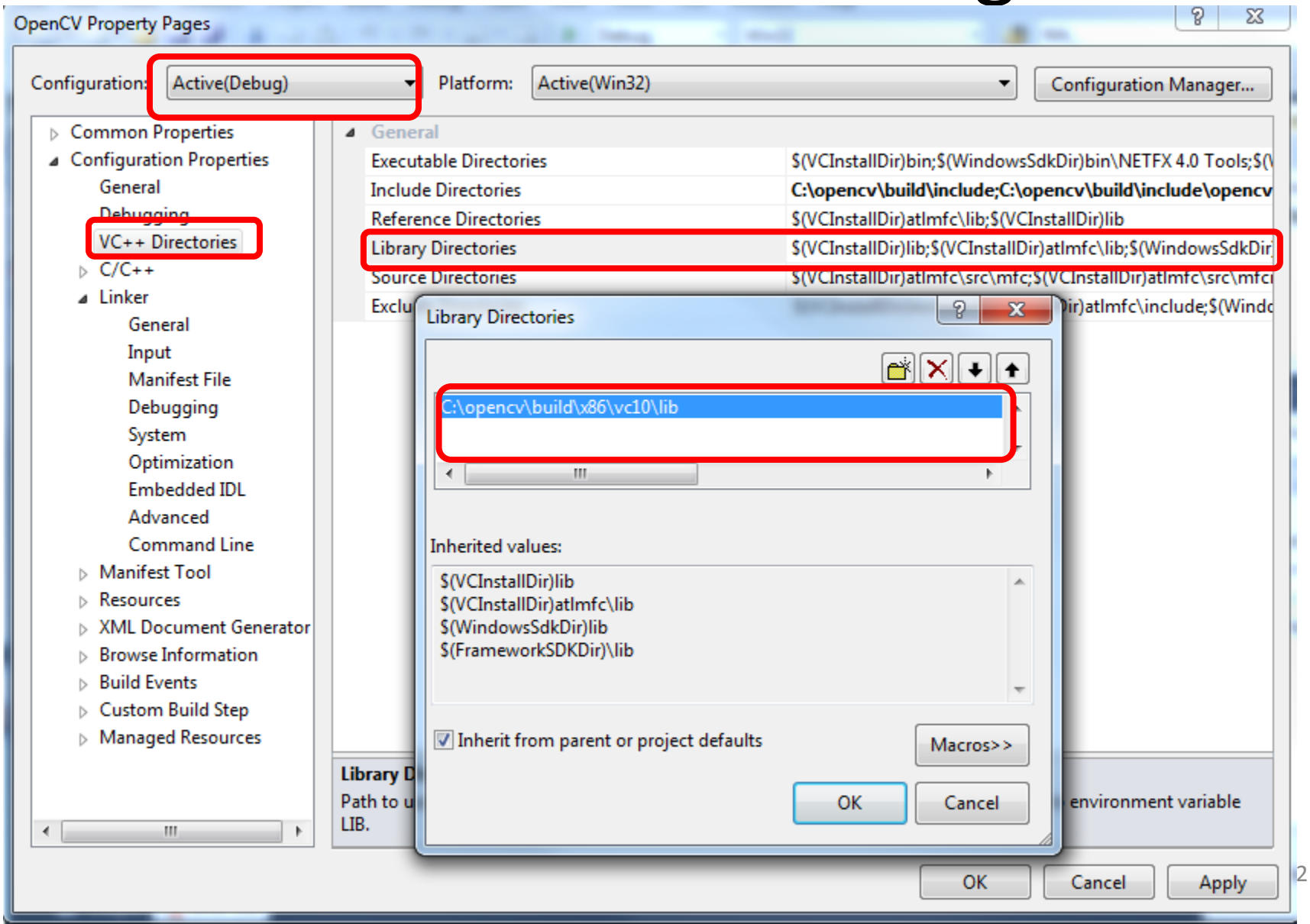
Environment Settings



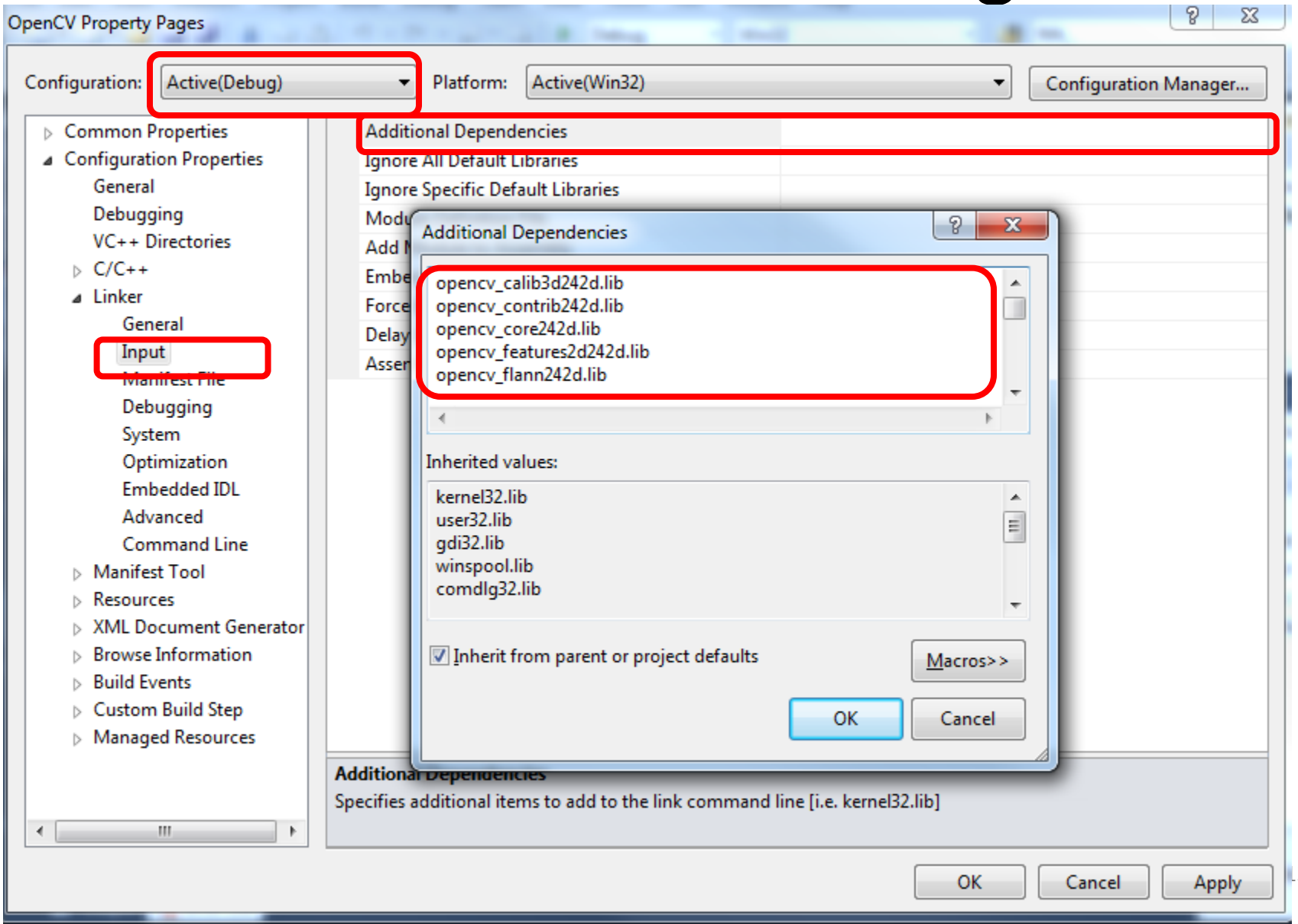
Environment Settings



Environment Settings



Environment Settings



Environment Settings

- Step 1: download OpenCV 2.4.11
- Step 2: unzip to C:\opencv\
- Step 3: Environment Variables
  - Computer > Properties > Advanced system settings > Environment Variables > System variables > Path > C:\opencv\build\x86\vc10\bin
- Step 4: To construct a project in Visual Studio 2010

Environment Settings

- Step 5:select project > right click > properties > VC++ Directories
  - Include Directories
    - C:\opencv\build\include
    - C:\opencv\build\include\opencv
    - C:\opencv\build\include\opencv2
  - Library Directories
    - C:\opencv\build\x86\vc10\lib

Environment Settings

- Step 6:select project > right click > properties > Linker > input(d->debug mode)
  - Additional Dependencies
    - opencv_calib3d2411d.lib
    - opencv_contrib2411d.lib
    - opencv_core2411d.lib
    - opencv_features2d2411d.lib
    - opencv_flann2411d.lib
    - opencv_gpu2411d.lib
    - opencv_highgui2411d.lib
    - opencv_imgproc2411d.lib
    - opencv_legacy2411d.lib
    - opencv_ml2411d.lib
    - opencv_nonfree2411d.lib
    - opencv_objdetect2411d.lib
    - opencv_photo2411d.lib
    - opencv_stitching2411d.lib
    - opencv_ts2411d.lib
    - opencv_video2411d.lib
    - opencv_videostab2411d.lib

Modules In OpenCV

OpenCV has several modules

Core

- A compact module defining basic data structures
  - basic functions used by all other modules.
- The data structures of the new version 2.x were substantially changes

Core – Basic Structures

- DataType
- Point_
- Point3_
- Size_
- Rect_
- Scalar_
- Range
- Ptr
- Mat
- …

Core – Mat

```
Mat image;
```

- Defined Mat classes to represent the matrix, replacing the previous CvMat and IplImage
- The class Mat represents an n-dimensional dense numerical single-channel or multi-channel array
  - It can be used to store
    real or complex-valued vectors and matrices
  - grayscale or color images
  - voxel volumes
  - vector fields
  - point clouds
  - Tensors
  - histograms
- **Mat handles all the memory automatically**

Core – Mat

```
class CV_EXPORTS Mat
{
public:
    // ... a lot of methods ...
    ...

    /*! includes several bit-fields:
         - the magic signature
         - continuity flag
         - depth
         - number of channels
     */
    int flags;
    //! the array dimensionality, >= 2
    int dims;
    //! the number of rows and columns or (-1, -1) when the array has more than 2 dimensions
    int rows, cols;
    //! pointer to the data
    uchar* data;

    //! pointer to the reference counter;
    // when array points to user-allocated data, the pointer is NULL
    int* refcount;

    // other members
    ...
};
```

```
if(! image.data )                              // Check for invalid input
{
    cout <<  "Could not open or find the image" << std::endl ;
    return -1;
}
```

Core – Mat

- Data formats
  - Matrix elements can be of the following types

    `CV_8U` - 8-bit unsigned integers (`0..255`)
    `CV_8S` - 8-bit signed integers (`-128..127`)
    `CV_16U` - 16-bit unsigned integers (`0..65535`)
    `CV_16S` - 16-bit signed integers (`-32768..32767`)
    `CV_32S` - 32-bit signed integers (`-2147483648..2147483647`)
    `CV_32F` - 32-bit floating-point numbers (`-FLT_MAX..FLT_MAX, INF, NAN`)
    `CV_64F` - 64-bit floating-point numbers (`-DBL_MAX..DBL_MAX, INF, NAN`)

  - Multi-channel types can be specified using following options

    `CV_8UC1` ... `CV_64FC4` constants (for a number of channels from 1 to 4)

Core – Point

- Point_ template class <- CvPoint,CvPoint2D32f
- Point3_ template class <- CvPoint2D32f

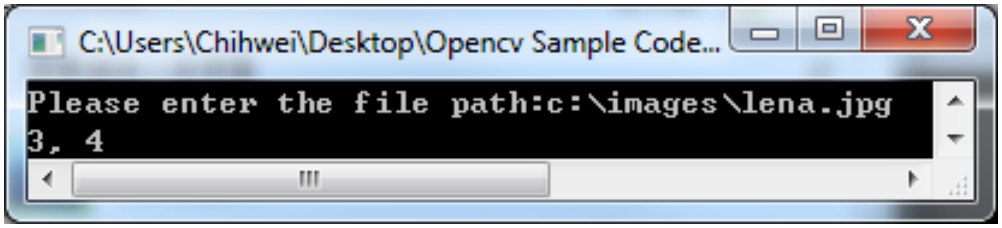
Core-Point

- *class* Point_
  - For convenience, the following type aliases are defined:

```
typedef Point_<int> Point2i;
typedef Point2i Point;
typedef Point_<float> Point2f;
typedef Point_<double> Point2d;
```

  - Example

```
Point2f a(0.3f, 0.f), b(0.f, 0.4f);
Point pt = (a + b)*10.f;
cout << pt.x << ", " << pt.y << endl;
```



imgproc

- an image processing module
  - linear and non-linear image filtering
    - GaussianBlur、Smooth、 Sobel
  - geometrical image transformations
    - resize, affine and perspective warping, generic table-based remapping
  - color space conversion
  - histograms
  - an so on.

Imgproc - GaussianBlur

- GaussianBlur
  - Smoothes an image using a Gaussian filter
  - void GaussianBlur(InputArray **src**, OutputArray **dst**, Size **ksize**, double **sigmaX**, double **sigmaY**=0, int**borderType**=BORDER_DEFAULT)
  - **Parameters:**
    - **src** – Source image:The depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F
    - **dst** – Destination image of the same size and type as src .
    - **ksize** – Gaussian kernel size.
    - **sigmaX** – Gaussian kernel standard deviation in X direction.
    - **sigmaY** – Gaussian kernel standard deviation in Y direction.
    - **borderType** – Pixel extrapolation method.

Imgproc - resize

- resize
  - Resizes an image
  - void resize(InputArray **src**, OutputArray **dst**, Size **dsize**, double **fx**=0, double**fy**=0, int **interpolation**=INTER_LINEAR)
    - **Parameters:**
      - **src** – Source image.
      - **dst** – Destination image
      - **interpolation** –Interpolation method:
        - » **INTER_NEAREST** - a nearest-neighbor interpolation
        - » **INTER_LINEAR** - a bilinear interpolation (used by default)
        - » **INTER_CUBIC** - a bicubic interpolation

video

- a video analysis module
  - motion estimation
  - background subtraction
    - BackgroundSubtractorMOG
  - object tracking algorithms.

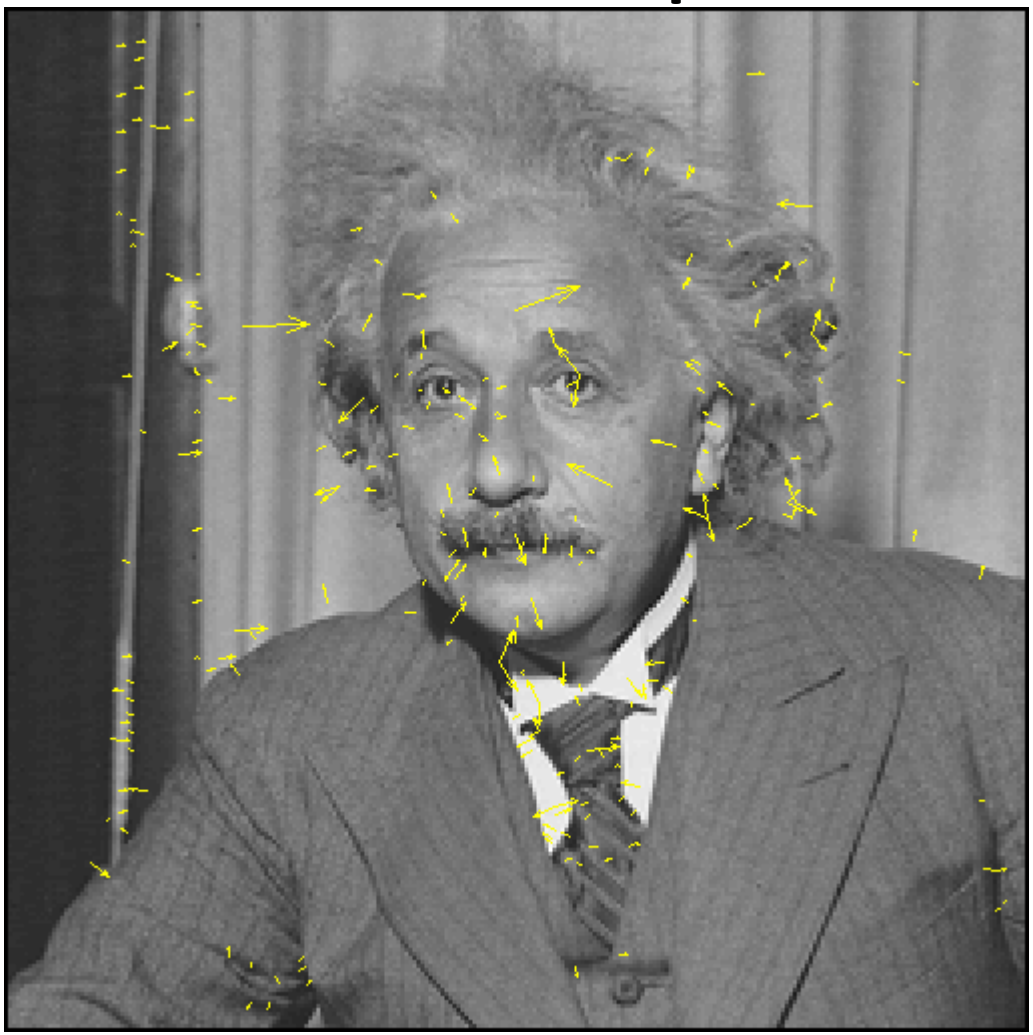
features2d

- salient feature detectors
  - FAST
  - SIFT
- descriptors
- descriptor matchers.

Features2d-SIFT

- Extract features and computes their descriptors sing SIFT algorithm
- void SIFT::operator()(InputArray **img**, InputArray **mask**, vector<KeyPoint>& **keypoints**, OutputArray **descriptors**, bool**useProvidedKeypoints**=false)
- **Parameters:**
  - **img** – Input 8-bit grayscale image
  - **mask** – Optional input mask that marks the regions where we should detect features.
  - **keypoints** – The input/output vector of keypoints
  - **descriptors** – The output matrix of descriptors. Pass cv::noArray() if you do not need them.
  - **useProvidedKeypoints** – Boolean flag. If it is true, the keypoint detector is not run.

SIFT descriptor



Features2d-FAST

- Detects corners using the FAST algorithm
- **C++:** void FAST(InputArray **image**, vector<KeyPoint>& **keypoints**, int **threshold**, bool **nonmaxSupression**=true)
- **Parameters:**
  - **image** – Image where keypoints (corners) are detected.
  - **keypoints** – Keypoints detected on the image.
  - **threshold** – Threshold on difference between intensity of the central pixel and pixels on a circle around this pixel.
  - **nonmaxSupression** – If it is true, non-maximum suppression is applied to detected corners (keypoints).

Features2d-FAST

- Sample of FAST

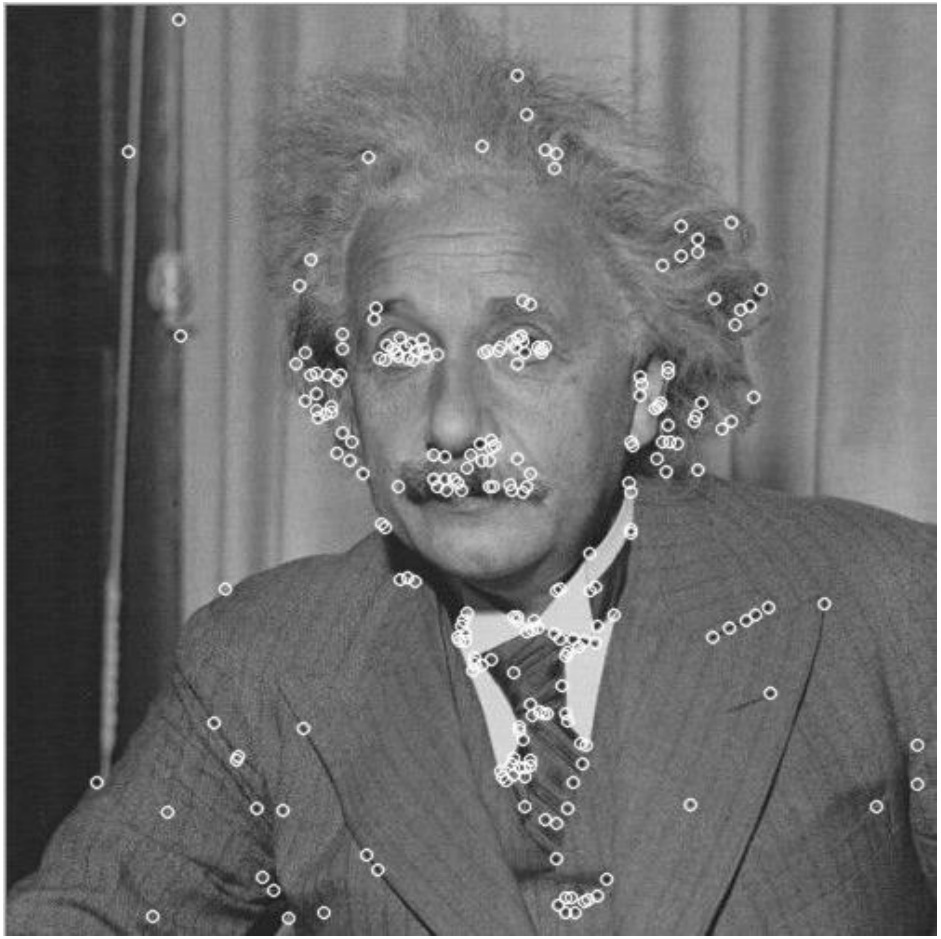
```
#include <vector>

using namespace cv;

void main()
{
    Mat image;
    image = imread("church01.jpg");
    // vector of keyPoints
    std::vector<KeyPoint> keyPoints;
    // construction of the fast feature detector object
    FastFeatureDetector fast(40);   // 檢測的閥值为40
    // feature point detection
    fast.detect(image,keyPoints);
    drawKeypoints(image, keyPoints, image, Scalar::all(255), DrawMatchesFlags::DRAW_OVER_OUTIMG);
    imshow("FAST feature", image);
    cvWaitKey(0);
}
```

Features2d-FAST

- Result of FAST



calib3d

- basic multiple-view geometry algorithms
  - single and stereo camera calibration
    - findHomography
  - elements of 3D reconstruction.

Calib3d-findHomography

- findHomography
  - Finds a perspective transformation between two planes.
  - **C++:** Mat findHomography(InputArray **srcPoints**, InputArray **dstPoints**, int **method**=0, double **ransacReprojThreshold**=3, OutputArray**mask**=noArray())
  - **Parameters:**
    - **srcPoints** – Coordinates of the points in the original plane
    - **dstPoints** – Coordinates of the points in the target plane
    - **method** –Method used to computed a homography matrix. The following methods are possible:
      - **0** - a regular method using all the points
      - **CV_RANSAC** - RANSAC-based robust method
      - **CV_LMEDS** - Least-Median robust method
    - **mask** – Optional output mask set by a robust method

objdetect

- detection of objects
- instances of the predefined classes
  - CascadeClassifier
  - for example: faces, eyes, mugs, people, cars, and so on

highgui

- an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities

Highgui - Reading and Writing Images

- Imdecode
- Imencode
- Imread
- imwrite

Highgui - Reading and Writing Video

- VideoCapture
- Open
- isOpened
- release

- grab
- retrieve
- read
- get
- set

Highgui – Other User Interfaces

- createTrackbar
- getTrackbarPos
- setTrackbarPos
- Imshow
- namedWindow
- destroyWindow
- destroyAllWindows
- MoveWindow
- ResizeWindow
- SetMouseCallback
- waitKey

Two ways of using cv namespace

- All the OpenCV classes and functions are placed into the **cv namespace**
  - use the cv::specifier

```
#include "opencv2/core/core.hpp"
...
cv::Mat H = cv::findHomography(points1, points2, CV_RANSAC, 5);
...
```

  - using namespace cv

```
#include "opencv2/core/core.hpp"
using namespace cv;
...
Mat H = findHomography(points1, points2, CV_RANSAC, 5 );
...
```

Example - Image

```
#include "stdafx.h"
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>

using namespace cv;
using namespace std;


int main(int argc, char** argv)
{
    string filepath;
    cout<<"Please enter the file path:";
    cin>>filepath;

    Mat image;
    image = imread(filepath, CV_LOAD_IMAGE_COLOR);   // Read the file

    if(! image.data )                              // Check for invalid input
    {
        cout <<  "Could not open or find the image" << std::endl ;
        return -1;
    }

    cv::namedWindow( "Display window", CV_WINDOW_AUTOSIZE );// Create a window for display.
    imshow( "Display window", image );                   // Show our image inside it.

    cv::waitKey(0);                                          // Wait for a keystroke in the window
    return 0;
}
```

imread

- Loads an image from a file

```
Mat image;
image = imread(filepath, CV_LOAD_IMAGE_COLOR);   // Read the file
```

  - **C++:** Mat *imread*(const string& **filename**, int **flags**=1)
- **filename** – Name of file to be loaded.

imread

- **C++:** Mat *imread*(const string& **filename**, int **flags**=1)
  - **flags** –Flags specifying the color type of a loaded image:
    - **> 0** Return a 3-channel color image
      - CV_LOAD_IMAGE_COLOR
    - **= 0** Return a grayscale image
      - CV_LOAD_IMAGE_GRAYSCALE
    - **< 0** Return the loaded image as is
      - CV_LOAD_IMAGE_ANYCOLOR

imshow

- Displays an image in the specified window.

```
// Create a window for display.
cv::namedWindow( "Display window", CV_WINDOW_AUTOSIZE );
// Show our image inside it.
imshow( "Display window", image );
```

- **C++:** void imshow(const string& **winname**, InputArray **mat**)
  - Parameters:
    - **winname** – Name of the window.
    - **image** – Image to be shown.

namedWindow

- Creates a window.
- **C++:** void namedWindow(const string& **winname**, int **flags**=WINDOW_AUTOSIZE)
  - Parameters:
    - **winname** – Name of the window in the window caption that may be used as a window identifier.
    - **flags** – Flags of the window.

Example - Video

```
#include "stdafx.h"
#include "Form1.h"
#include "opencv2/opencv.hpp"

using namespace OpenCV;
using namespace std;
void Form1::start_Click(System::Object^ sender, System::EventArgs^ e)
{
    IplImage pImg;
    Graphics ^graph1;
    graph1 = this->CreateGraphics();
    cv::VideoCapture cap(0);

    try{
        for (;;)
        {
            cv::Mat frame;
            cap >> frame;
            cv::imshow("Video", frame);
            pImg = IplImage(frame);
            Bitmap ^bmp = gcnew Bitmap( pImg.width, pImg.height, pImg.widthStep,
                          System::Drawing::Imaging::PixelFormat::Format24bppRgb, (IntPtr)pImg.imageData );
            graph1->DrawImage( bmp, 0, 0 );
            delete bmp;
            if(cv::waitKey(30) >= 0) break;
        }
    }
    catch(cv::Exception& e)
    {
        MessageBox::Show("Exception");
    }
```

VideoCapture

- Class for video capturing from video files or cameras.
  - **C++:** VideoCapture::VideoCapture(const string& **filename**)
  - **C++:** VideoCapture::VideoCapture(int **device**)

VideoCapture

```
#include "opencv2/opencv.hpp"

using namespace cv;

int main(int, char**)
{
    VideoCapture cap(0); // open the default camera
    if !cap.isOpened()   // check if we succeeded
        return -1;

    Mat edges;
    namedWindow("edges",1);
    for(;;)
    {
        Mat frame;
        cap >> frame; // get a new frame from camera
        cvtColor(frame, edges, CV_BGR2GRAY);
        GaussianBlur(edges, edges, Size(7,7), 1.5, 1.5);
        Canny(edges, edges, 0, 30, 3);
        imshow("edges", edges);
        if(waitKey(30) >= 0) break;
    }
    // the camera will be deinitialized automatically in VideoCapture destructor
    return 0;
}
```

VideoCapture::isOpened

- Returns true if video capturing has been initialized already.
- **C++:** bool VideoCapture::isOpened()

Reference

- OpenCV installation:
- http://docs.opencv.org/doc/tutorials/introduction/table_of_content_introduction/table_of_content_introduction.html#table-of-content-introduction
- http://yester-place.blogspot.tw/2008/06/opencv.html
- http://blog.csdn.net/yang_xian521/article/details/6894228
- http://docs.opencv.org/index.html