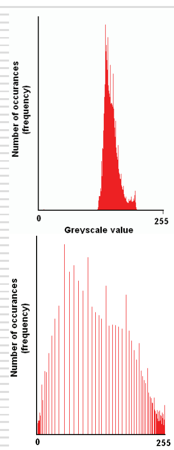
Introduction to OpenCV

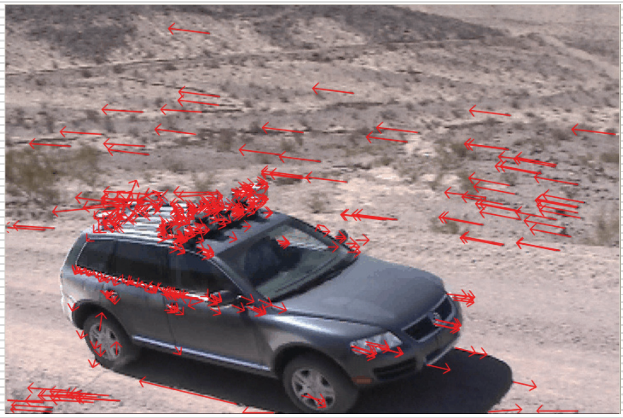
David Stavens
Stanford Artificial Intelligence Lab

Aside: Histogram Equalization



Images are from Wikipedia.

Today we'll code:



A fully functional sparse optical flow algorithm!

Plan

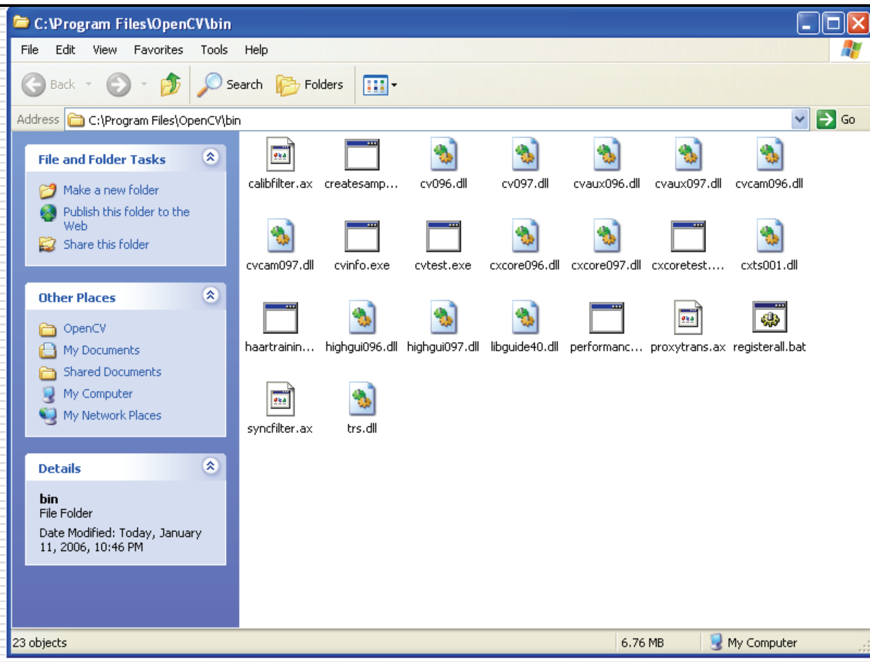
- OpenCV Basics
  - What is it?
  - How do you get started with it?
- Feature Finding and Optical Flow
  - A brief mathematical discussion.
- OpenCV Implementation of Optical Flow
  - Step by step.

What is OpenCV?

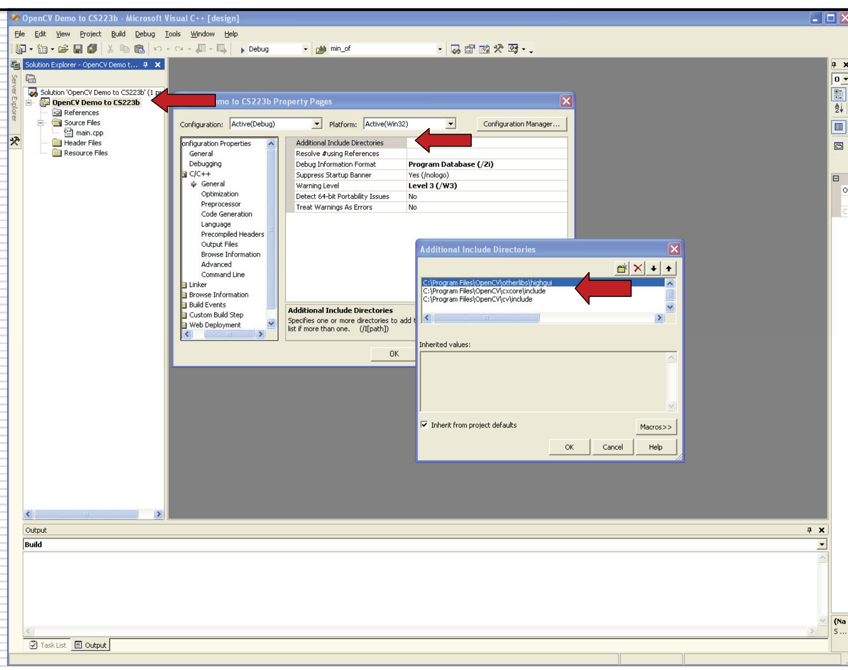
- Created/Maintained by Intel.
- Really four libraries in one:
  - "CV" – Computer Vision Algorithms
    - All the vision algorithms.
  - "CVAUX" – Experimental/Beta
  - "CXCORE" – Linear Algebra
    - Raw matrix support, etc.
  - "HIGHGUI" – Media/Window Handling
    - Read/write AVIs, window displays, etc.
- Check out the samples directory!

Installing OpenCV

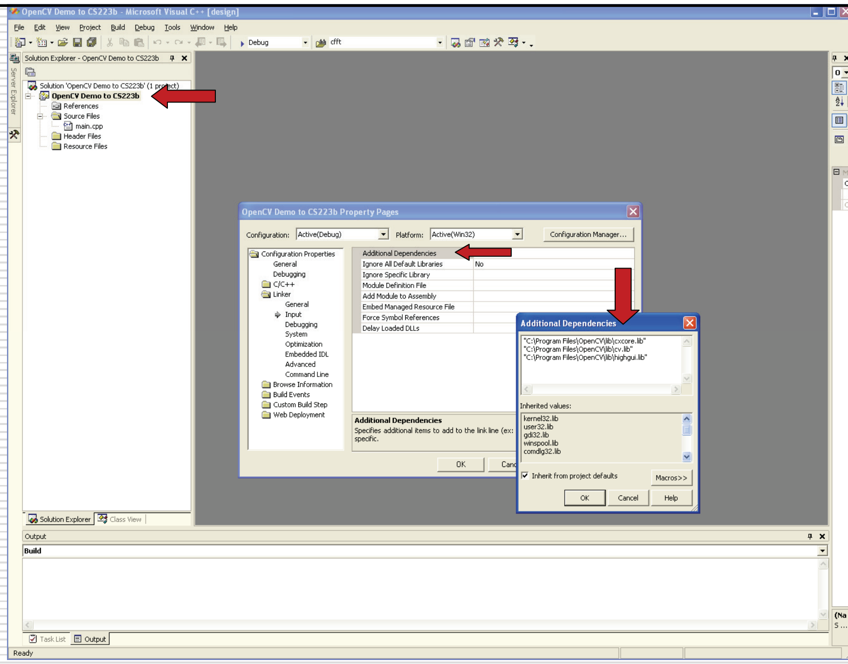
- Download from:
  - http://sourceforge.net/projects/opencvlibrary/
- Be sure to get Version 1.0.0.
- Windows version comes with an installer.
- Linux: (Install ffMPEG first!)
  - gunzip opencv-1.0.0.tar.gz; tar –xvf opencv-1.0.0.tar
  - cd opencv-1.0.0; ./configure --prefix=/usr; make
  - make install [as root]



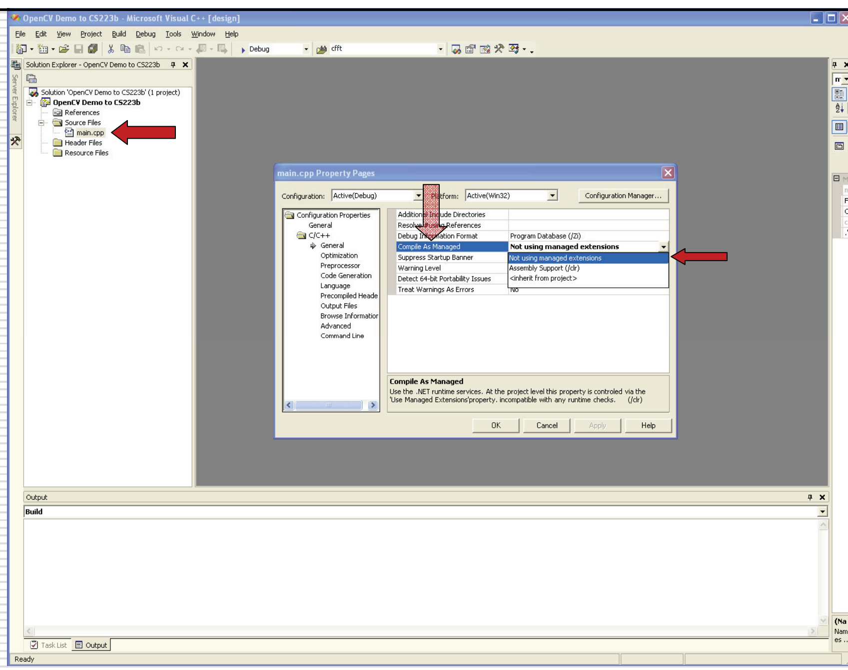
Copy all the DLLs in \OpenCV\bin to \WINDOWS\System32.



Tell Visual Studio where the includes are. (Import a C file first.)



Tell Visual Studio to link against cxcore.lib, cv.lib, and highgui.lib.

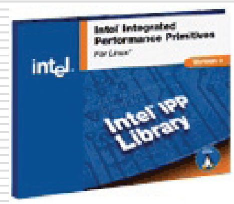


Tell Visual Studio to disable managed extensions.

Better Performance: ICC and IPL

- Intel C/C++ Compiler

- Intel Integrated Performance Primitives



- ~30 – 50% Speed Up

Plan

- ✓ OpenCV Basics
  - ✓ What is it?
  - ✓ How do you get started with it?
- Feature Finding and Optical Flow
  - A brief mathematical discussion.
- OpenCV Implementation of Optical Flow
  - Step by step.

Optical Flow: Overview

- Given a set of points in an image, find those same points in another image.
- or, given point $[u_x, u_y]^T$ in image I_1 find the point $[u_x + \delta_x, u_y + \delta_y]^T$ in image I_2 that minimizes ε:

$$\varepsilon(\delta_x, \delta_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} \left(I_1(x,y) - I_2(x+\delta_x, y+\delta_y) \right)$$

- (the Σ/w's are needed due to the aperture problem)

Optical Flow: Utility

- Tracking points ("features") across multiple images is a fundamental operation in many computer vision applications:
  - To find an object from one image in another.
  - To determine how an object/camera moved.
  - To resolve depth from a single camera.
- Very useful for the 223b competition.
  - Determine motion. Estimate speed.
- But what are good features to track?

Finding Features: Overview

- Intuitively, a good feature needs at least:
  - Texture (or ambiguity in tracking)
  - Corner (or aperture problem)
- But what does this mean formally?

$$\begin{bmatrix} \sum_{neighborhood} \left(\frac{\partial I}{\partial x}\right)^2 & \sum_{neighborhood} \frac{\partial^2 I}{\partial x \partial y} \\ \sum_{neighborhood} \frac{\partial^2 I}{\partial x \partial y} & \sum_{neighborhood} \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

- A good feature has big eigenvalues, implies:
  - Texture
  - Corner
- Shi/Tomasi. Intuitive result really part of motion equation. High eigenvalues imply reliable solvability. Nice!

Plan

- ✓ OpenCV Basics
  - ✓ What is it?
  - ✓ How do you get started with it?
- ✓ Feature Finding and Optical Flow
  - ✓ A brief mathematical discussion.
- OpenCV Implementation of Optical Flow
  - Step by step.

So now let's code it!

- Beauty of OpenCV:
  - All of the Above = Two Function Calls
  - Plus some support code :-)
- Let's step through the pieces.
- These slides provide the high-level.
  - Full implementation with extensive comments:
    - http://ai.stanford.edu/~dstavens/cs223b

ai.stanford.edu/~dstavens/cs223b

- Three versions of the code:
  - optical_flow_demo.cpp.windows
    - For Windows, full functionality.
  - optical_flow_demo.cpp.linux.limited_api
    - OpenCV for Linux missing some functions.
  - optical_flow_demo.cpp.linux.full_api
    - For Mac OS X? Full functionality?
    - Also for Linux if/when API complete.

Step 1: Open Input Video

```
CvCapture *input_video =
    cvCaptureFromFile("filename.avi");
```

- Failure modes:
  - The file doesn't exist.
  - The AVI uses a codec OpenCV can't read.
    - Codecs like MJPEG and Cinepak are good.
    - DV, in particular, is bad.

Step 2: Read AVI Properties

```
CvSize frame_size;
frame_size.height =
    cvGetCaptureProperty( input_video,
    CV_CAP_PROP_FRAME_HEIGHT );
```

- Similar construction for getting the width and the number of frames.
  - See the handout.

Step 3: Create a Window

cvNamedWindow(“Optical Flow”, CV_WINDOW_AUTOSIZE);

- We will put our output here for visualization and debugging.

Step 4: Loop Through Frames

- Go to frame N:
  cvSetCaptureProperty(input_video, CV_CAP_PROP_POS_FRAMES, N);
- Get frame N:
  IplImage *frame = cvQueryFrame(input_video);
  - Important: cvQueryFrame *always* returns a pointer to the same location in memory.

Step 5: Convert/Allocate

- Convert input frame to 8-bit mono:

```
IplImage *frame1 =
    cvCreateImage( cvSize(width, height),
        IPL_DEPTH_8U, 1);
cvConvertImage( frame, frame1 );
```

- Actually need third argument to conversion: CV_CVTIMG_FLIP.

Step 6: Run Shi and Tomasi

```
CvPoint2D32f frame1_features[N];
cvGoodFeaturesToTrack(
    frame1, eig_image, temp_image,
    frame1_features, &N, .01, .01, NULL);
```

- Allocate eig,temp as in handout.
- On return frame1_features is full and N is the number of features found.

Step 7: Run Optical Flow

```
char optical_flow_found_feature[];
float optical_flow_feature_error[];
CvTermCriteria term =
    cvTermCriteria( CV_TERMCRIT_ITER |
    CV_TERMCRIT_EPS, 20, .3 );

cvCalcOpticalFlowPyrLK( … );
```

- 13 arguments total. All of the above.
  - Both frames, both feature arrays, etc.
- See full implementation in handout.

Step 8: Visualize the Output

```
CvPoint p, q;
p.x = 1; p.y = 1; q.x = 2; q.y = 2;
CvScalar line_color;
line_color = CV_RGB(255, 0, 0);
int line_thickness = 1;

cvLine(frame1, p,q, line_color, line_thickness, CV_AA, 0);
cvShowImage("Optical Flow", frame1);
```

- CV_AA means draw the line antialiased.
- 0 means there are no fractional bits.

Step 9: Make an AVI output

```
CvVideoWriter *video_writer =
    cvCreateVideoWriter( "output.avi",
    -1, frames_per_second, cvSize(w,h) );
```

- ("-1" pops up a nice GUI.)

```
cvWriteFrame(video_writer, frame);
```

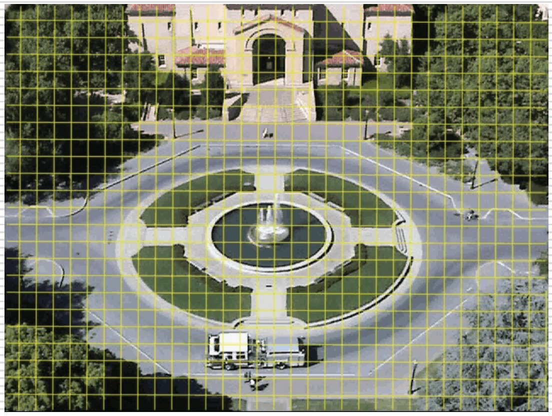
  - Just like cvShowImage(window, frame);

```
cvReleaseVideoWriter(&video_writer);
```

Let's watch the result:

(Stanley before turning blue.)

That's the first step for…



Stavens, Lookingbill, Lieb, Thrun; CS223b 2004; ICRA 2005

Corresponding functions…

Wed, Jan. 18	Thrun	McCullough 115	Features I: Image Processing, Filters, edges, corners, Hough transform	cvSobel, cvLaplace, cvCanny, cvCornerHarris, cvGoodFeaturesToTrack, cvHoughLines2, cvHoughCircles
Mon, Jan. 23	Thrun	McCullough 115	Features II: Fourier, Phase, Pyramids, SIFT features, log-polar	cvWarpAffine, cvWarpPerspective, cvLogPolar, cvPyrSegmentation
Wed, Jan. 25	Thrun	McCullough 115	Camera Calibration	cvCalibrateCamera2, cvFindExtrinsicCameraParams2, cvFindChessboardCorners, cvUndistort2, cvFindHomography, cvProjectPoints2

Corresponding functions…

Mon, Jan. 30	Thrun	McCullough 115	Stereopsis I: Epipolar geometry, feature-based correspondence, dense stereo		Homework	
Wed, Feb. 1	Thrun	McCullough 115	Stereopsis II: active illumination, layered, volumetric, symmetry			
Mon, Feb. 6	Thrun	McCullough 115	Optical Flow			
Wed, Feb. 8	Thrun	McCullough 115	Affine Structure from Motion, Correspondence, RANSAC			

cvFindFundamentalMat,
cvComputeCorrespondEpilines,
cvConvertPointsHomogenious,
cvCalcOpticalFlowHS,
cvCalcOpticalFlowLK

cvCalcOpticalFlowPyrLK,
cvFindFundamentalMat (RANSAC)

Corresponding functions…

Mon, Feb. 13	Rick Szeliski (Microsoft Research)	McCullough 115	Structure from Motion II		Homework assignment III due at 11:59pm PST	
Wed, Feb. 15	Rick Szeliski (Microsoft Research)	McCullough 115	Image Stitching, Mosaicking			
Mon, Feb. 27	Thrun	McCullough 115	Tracking: Kalman filter and condensation			

cvMatchTemplate,
cvMatchShapes, cvCalcEMD2,
cvMatchContourTrees

cvKalmanPredict,
cvConDensation, cvAcc
cvMeanShift, cvCamShift

Corresponding functions…

Wed, Mar. 1	Thrun	McCullough 115	Markov Random Fields for 3-D reconstruction	
Mon, Mar. 6	Thrun	McCullough 115	Segmentation and Grouping (K-means, graph cuts), Snakes and active contours	cvSnakeImage, cvKMeans2, cvSeqPartition, cvCalcSubdivVoronoi2D, cvCreateSubdivDelaunay2D
Wed, Mar. 8	Gary Bradski, Intel Research and Stanford	McCullough 115	Object Detection and ClassificationUsing Machine Learning	cvHaarDetectObjects

A few closing thoughts…

- Feel free to ask questions!
  - david.stavens@ai.stanford.edu
  - My office: Gates 254
- Good luck!! 223b is fun!! :-)