Find Mii Project and OpenCV Tutorial

Zixuan Wang

Overview

- FindMii Project
- OpenCV Introduction
- HighGUI
- Basic Operations
- AdaBoost
- Face Detection
- Optical Flow
- Template Matching
- Local Feature

FindMii Project

- It's the default project of this class.
  - Matlab
  - OpenCV
- If you prefer OpenCV, there is a tutorial later.
- Dates
  - Oct 16 (11:59pm): Finalizing team members and proposal submission.
  - Nov 6 (11:59pm): Project milestone.
  - Dec 3 (11:59pm): Code and writeup due.
  - Dec 6 (time 1-3 pm): Course project presentation.

FindMii Project

- 4 tasks
  - Task 1 : Find this Mii
  - Task 2 : Find 2 look-alikes
  - Task 3 : Find n odd Miis out
  - Task 4 : Find the fastest Mii
- 3 levels
  - Easy, medium, hard
- 3x4 = a lot
  - Min-requirement : 3 different tasks (1 level for each)

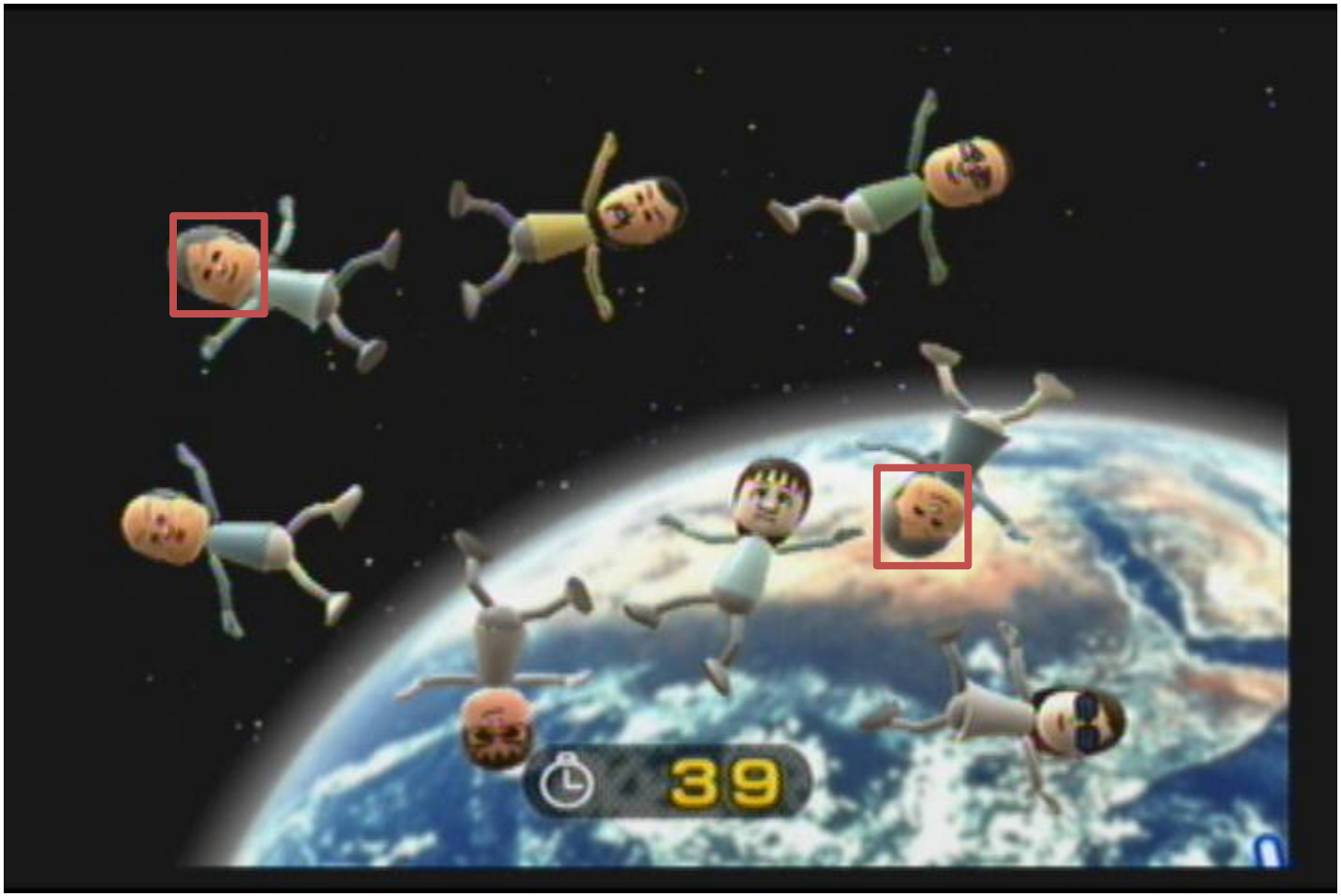
FindMii Data

- 4 .m files
  - ClickMii.m (visualize your results)
  - FindMiiMain.m (main script)
  - FindMiiTask1Level1.m (dummy function)
  - PlayMii.m (play video)
- opencvexample.cpp and Makefile (for OpenCV)
- Training data
  - /afs/ir/class/cs231a/findmii/data
- Ground truth
  - /afs/ir/class/cs231a/findmii/gt

Task 1: Find This Mii

Level 1 (Easy)

Task 2: Find 2 Look-alikes



Level 2 (Medium)

Task 3: Find n Odd Miis Out

Level 1 (Easy)

Task 4: Find the Fastest Mii

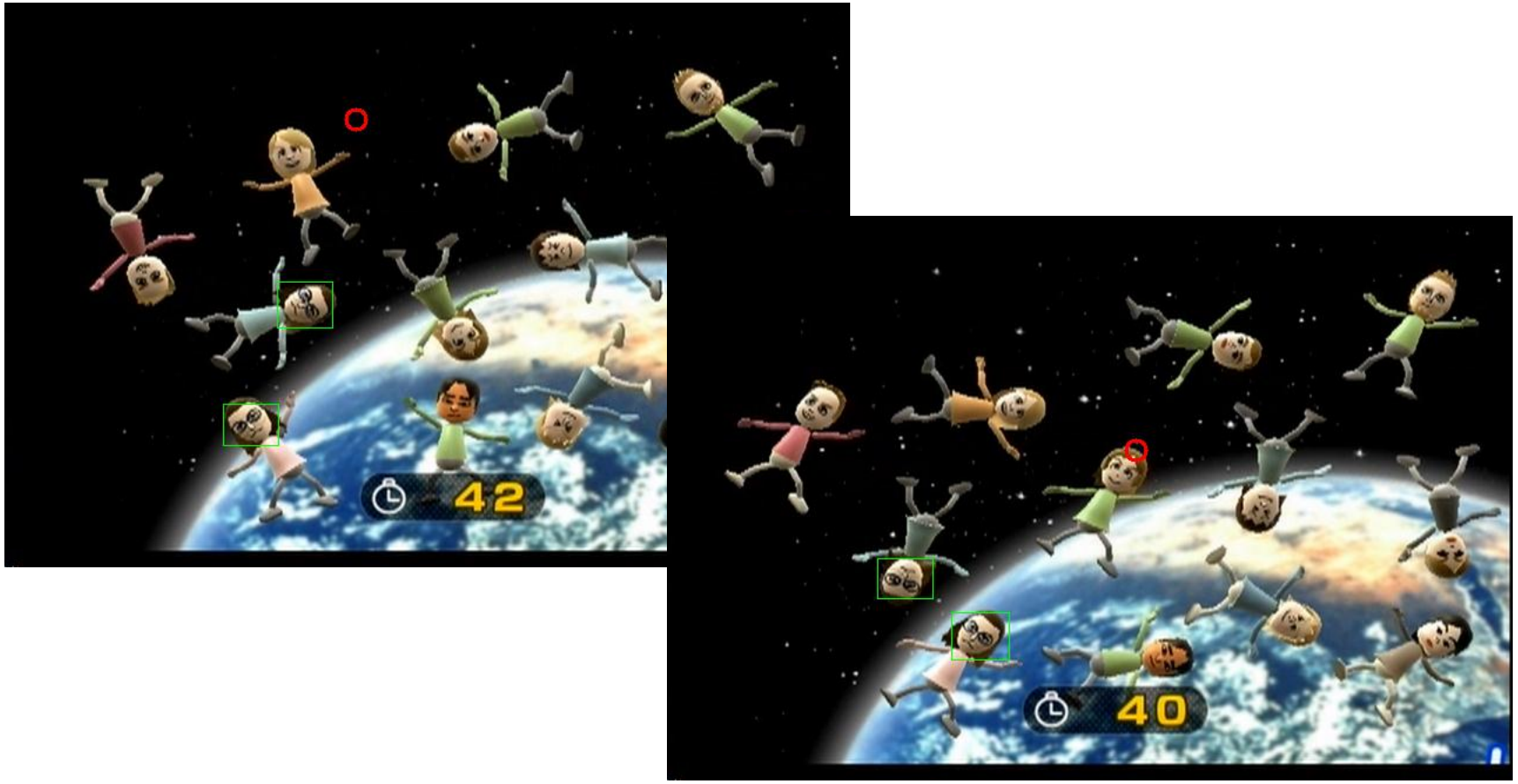
Level 3 (Hard)

Compile with OpenCV

- You program should compile and run properly on the ‘corn’ clusters, using the following header file and library file path:
  - /afs/ir/class/cs231a/opencv/include
  - /afs/ir/class/cs231a/opencv/lib
- Please follow the example given by Makefile and 'opencvexample.cpp' in the project AFS path.
- Windows executable or Visual Studio projects will NOT work.

Evaluations

```
click = [12  300  100;  66  400  200];
ClickMii('data/t2l2.avi', 'gt/t2l2.gt', click);
```



Scoring

- For a given task & level
  - A correct click on frame 1 is worth 1 point, each frame thereafter is discounted by 0.99. And you can NOT access (even read into memory) any frame after the one you made your final click for a task. Score is averaged over multiple clicks if applicable. For example, if a task requires two clicks, you return 1st click at frame 5, and 2nd click at frame 20 (both of them are correct), your score would be (0.99^4 + 0.99^19)/2 = 0.8934. In this case, you can NOT access any frame after frame 20.
- For a given task
  - The highest score you achieve on the 3 levels. When comparing the scores on different levels, scores on level 2 are multiplied by 1.5, and scores on level 3 are multiplied by 2. For example, for task 1, your score is 1.0 on level 1, and 0.77 on level 2, and you are not doing level 3. Your final score for task 1 would be max(1*1, 0.77*1.5, 0*2) = 1.155.
- For the challenge
  - Sum up your scores in highest 3 of the 4 tasks.

OpenCV – Introduction

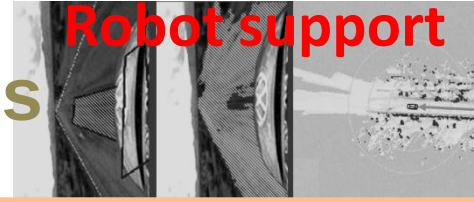
- OpenCV stands for the Open Source Computer Vision Library.
  - It was founded at Intel in 1999, went through some lean years after the .com bust but is now under active development, now receiving ongoing support from Willow Garage.
- OpenCV is free for commercial and research use.
  - It has a BSD license. The library runs across many platforms and actively supports Linux, Windows and Mac OS.
- OpenCV was founded to advance the field of computer vision.
  - It gives everyone a reliable, real time infrastructure to build on. It collects and makes available the most useful algorithms.

OpenCV - Features

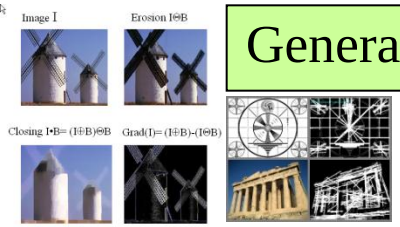
- Cross Platform
  - Windows, Linux, Mac OS
- Portable
  - iPhone
  - Android.
- Language Support
  - C/C++
  - Python

OpenCV Overview: > 500 functions

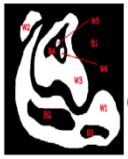
opencv.willowgarage.com

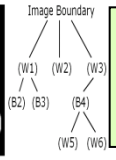


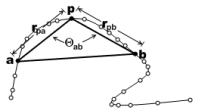
General Image Processing Functions



Geometric Descriptors







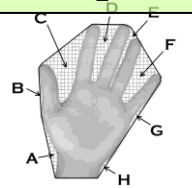
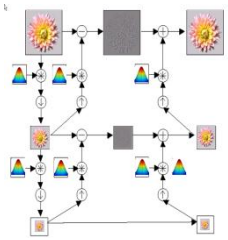
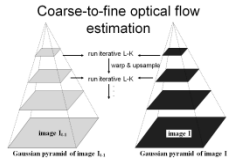
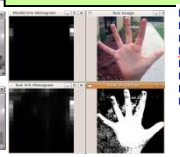


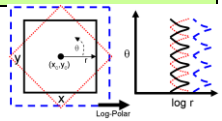
Image Pyramids



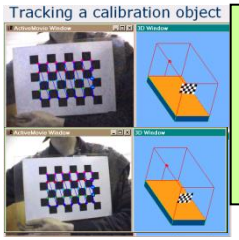


Segmentation

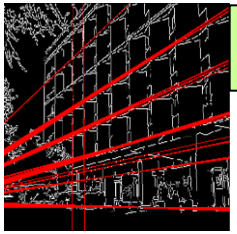


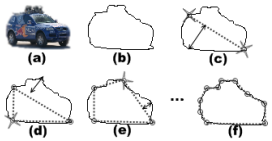


Camera Calibration, Stereo, 3D

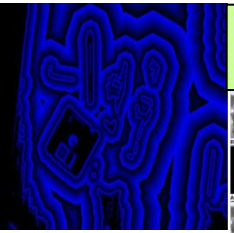


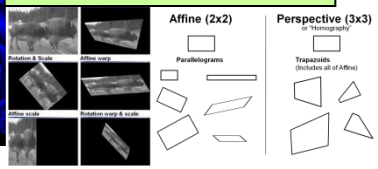
Features



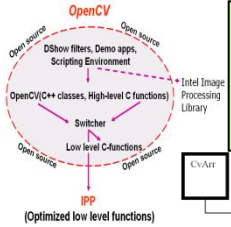


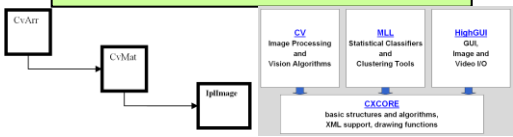
Transforms



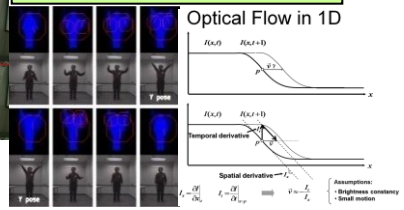


Utilities and Data Structures



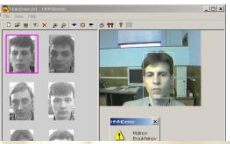


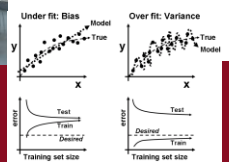
Tracking



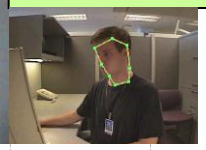
Machine Learning:

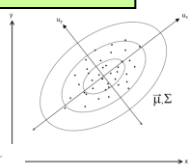
- Detection,
- Recognition

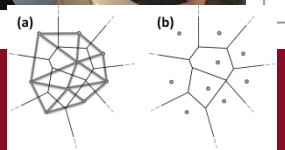




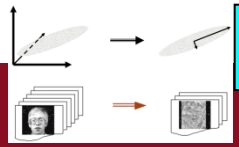
Fitting





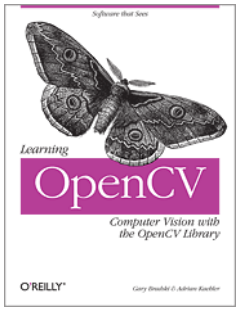


Matrix Math



OpenCV – Getting Started

- Download OpenCV
  - http://opencv.org
  - Install from macports/aptitude
- Setting up
  - Comprehensive guide on setting up OpenCV in various environments at the official wiki.
- Online Reference:
  - http://docs.opencv.org
- Two books



HighGUI

- HighGUI
- Image I/O, rendering
- Processing keyboard and other events, timeouts
- Trackbars
- Mouse callbacks
- Video I/O

HighGUI: OpenCV Functions

- void cv::namedWindow(const string& winname, int flags=WINDOW_AUTOSIZE);
  - Creates window accessed by its name. Window handles repaint, resize events. Its position is remembered in registry.
- void cv::destroyWindow(const string& winname);
- void cv::imshow(const string& winname, cv::Mat& mat);
  - Copies the image to window buffer, then repaints it when necessary. {8u|16s|32s|32f}{C1|3|4} are supported.
  - Only the whole window contents can be modified. Dynamic updates of parts of the window are done using operations on images, drawing functions etc.

HighGUI: OpenCV Functions

- Mat imread(const string& filename, int flags=1);
  - loads image from file, converts to color or grayscale, if need, and returns it (or returns empty cv::Mat()).
  - image format is determined by the file contents.
- bool imwrite(const string& filename, Mat& image);
  - saves image to file, image format is determined from extension.
- Example: convert JPEG to PNG

```
cv::Mat img = cv::imread("picture.jpeg");
if(!img.empty()) cv::imwrite( "picture.png", img );
```

HighGUI Sample Code

- Example code: load an image from disk and display it on the screen.

```
#include <opencv2/opencv.hpp>
int main( int argc, char* argv[] ) {
    cv::Mat image = cv::imread( argv[1] );
    cv::namedWindow( “Example1”,CV_WINDOW_AUTOSIZE );
    cv::imshow( “Example1”, image );
    cv::waitKey(0);
    cv::destroyWindow( “Example1” );
    return 0;
}
```

HighGUI Sample Code

- Example code: load a video file and process each frame.

```
#include <opencv2/opencv.hpp>
int main( int argc, char* argv[] ) {
    cv::VideoCapture capture("filename.avi");
    if (!capture.isOpened())     return 1;
    cv::Mat frame;
    while (true) {
        capture >> frame; if(!frame.data) break;
        //process the frame here
    }
    capture.release();
    return 0;
}
```

Key Classes

Key OpenCV Classes

`Point_`	Template 2D point class
`Point3_`	Template 3D point class
`Size_`	Template size (width, height) class
`Vec`	Template short vector class
`Scalar`	4-element vector
`Rect`	Rectangle
`Range`	Integer value range
`Mat`	2D dense array (used as both a matrix or an image)
`MatND`	Multi-dimensional dense array
`SparseMat`	Multi-dimensional sparse array
`Ptr`	Template smart pointer class

Manipulate an Image (Mat)

Matrix Basics

Create a matrix

```
Mat image(240, 320, CV_8UC3);
```

[Re]allocate a pre-declared matrix

```
image.create(480, 640, CV_8UC3);
```

Create a matrix initialized with a constant

```
Mat A33(3, 3, CV_32F, Scalar(5));
Mat B33(3, 3, CV_32F); B33 = Scalar(5);
Mat C33 = Mat::ones(3, 3, CV_32F)*5.;
Mat D33 = Mat::zeros(3, 3, CV_32F) + 5.;
```

Create a matrix initialized with specified values

```
double a = CV_PI/3;
Mat A22 = (Mat_<float>(2, 2) <<
  cos(a), -sin(a), sin(a), cos(a));
float B22data[] = {cos(a), -sin(a), sin(a), cos(a)};
Mat B22 = Mat(2, 2, CV_32F, B22data).clone();
```

Initialize a random matrix

```
randu(image, Scalar(0), Scalar(256)); // uniform dist
randn(image, Scalar(128), Scalar(10)); // Gaussian dist
```

Convert matrix to/from other structures (without copying the data)

```
Mat image_alias = image;
float* Idata=new float[480*640*3];
Mat I(480, 640, CV_32FC3, Idata);
vector<Point> iptvec(10);
Mat iP(iptvec); // iP – 10x1 CV_32SC2 matrix
IplImage* oldC0 = cvCreateImage(cvSize(320,240),16,1);
Mat newC = cvarrToMat(oldC0);
IplImage oldC1 = newC; CvMat oldC2 = newC;
```

... (with copying the data)

```
Mat newC2 = cvarrToMat(oldC0).clone();
vector<Point2f> ptvec = Mat_<Point2f>(iP);
```

Access matrix elements

```
A33.at<float>(i,j) = A33.at<float>(j,i)+1;
Mat dyImage(image.size(), image.type());
for(int y = 1; y < image.rows-1; y++) {
  Vec3b* prevRow = image.ptr<Vec3b>(y-1);
  Vec3b* nextRow = image.ptr<Vec3b>(y+1);
  for(int x = 0; y < image.cols; x++)
    for(int c = 0; c < 3; c++)
     dyImage.at<Vec3b>(y,x)[c] =
     saturate_cast<uchar>(
     nextRow[x][c] - prevRow[x][c]);
}
Mat_<Vec3b>::iterator it = image.begin<Vec3b>(),
  itEnd = image.end<Vec3b>();
for(; it != itEnd; ++it)
  (*it)[1] ^= 255;
```

Mat does reference counting, so it does the right thing when it goes out of scope you can also easily make STL vectors or maps out of Mat.

AdaBoost

- Boosting
  - Covered in Lecture 2 (Sept 27).
  - Can a set of weak learners create a single strong learner?
  - Weak classifiers weighted by its accuracy.
- Adaptive Boosting
  - Subsequent classifiers focus on instances misclassified by previous classifiers.
  - Formulated by Yoav Freund and Robert Schapire.
- Y. Freund, R.E. Schapire. A Decision-theoretic Generalization of On-line Learningand an Application to Boosting. Journal of Computer and System Sciences. 1997

Face Detection

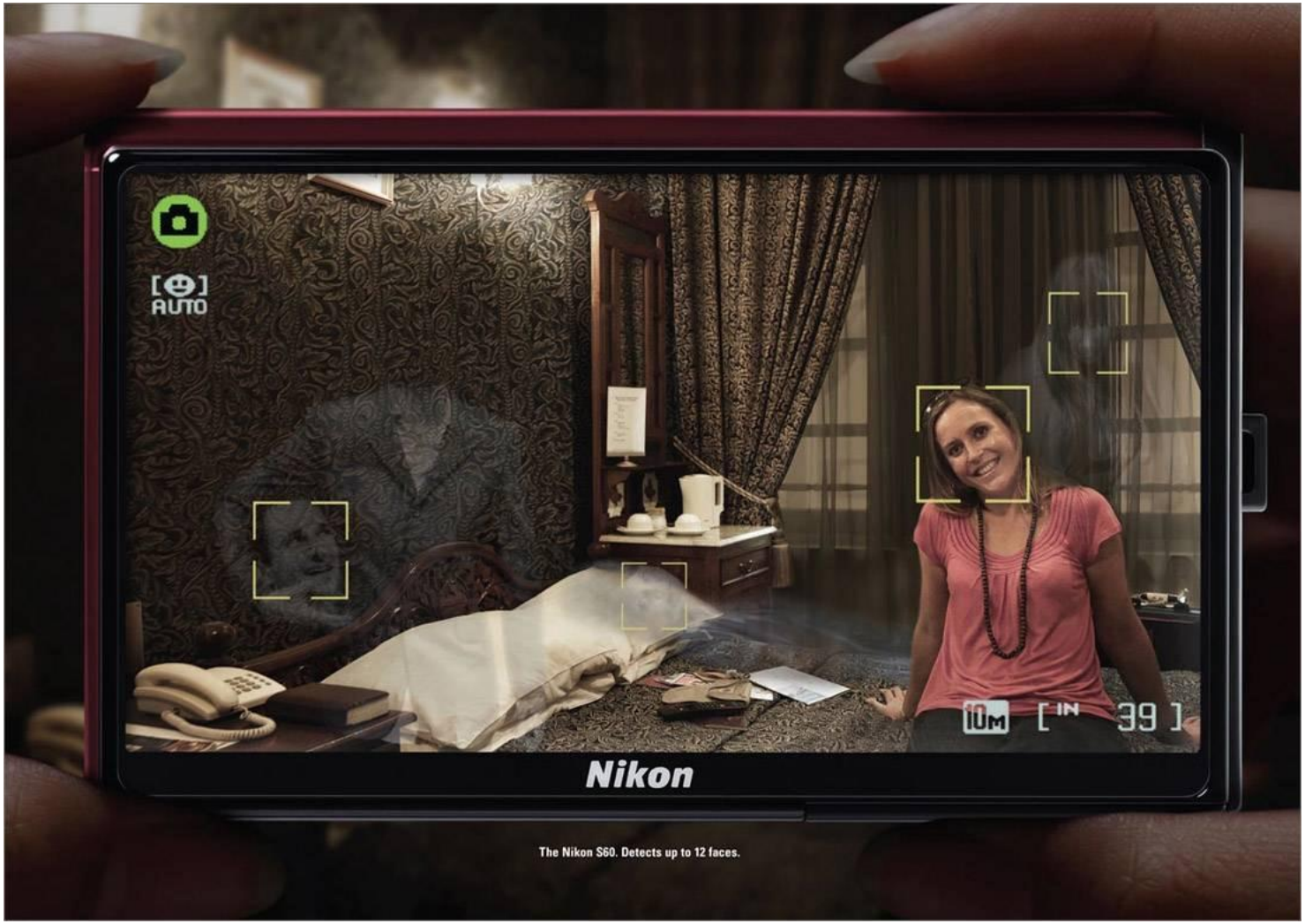
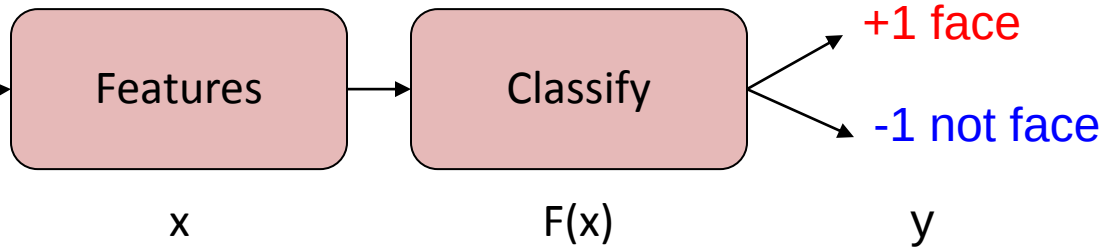


Photo Courtesy Nikon Corp.

Face Detection



- Slide a window over the image
  - Also change the size of the window
- Extract features for each window
- Classify each window into face/non-face

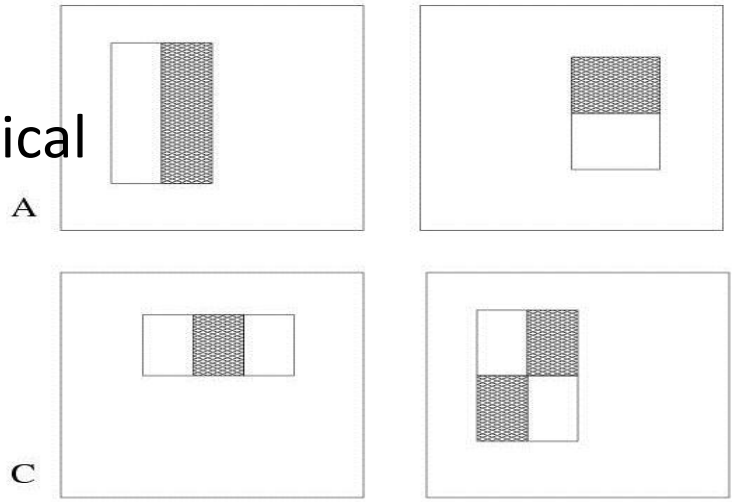
Face Detection: Viola-Jones

- Robust and fast
- Introduced by Paul Viola and Michael Jones
  - http://research.microsoft.com/~viola/Pubs/Detect/violaJones_CVPR2001.pdf
- Haar-like Features

- Use AdaBoost to select good classifiers and combine them

Face Detection: Viola-Jones

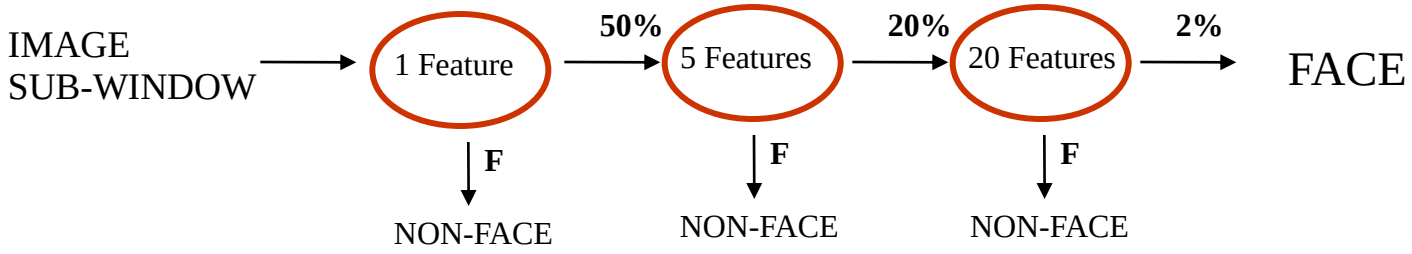
- Haar-like features
  - Rectangular blocks, white or black
  - 3 types of features:
    - two rectangles: horizontal/vertical
    - three rectangles
    - four rectangles
  - In 24x24 window: 180,000 possible features
  - Simple weak classifier



Face Detection: AdaBoost

- AdaBoost
  - Given set of “weak” classifiers:
    - Pick best one.
    - Reweight training examples, so that misclassified images have larger weight.
    - Reiterate; then linearly combine resulting classifiers.
- Result
  - A classifier with 200 rectangle features was learned using AdaBoost.
  - 95% correct detection on test set with 1 in 14084 false positives.
  - Less susceptible to over-fitting.

Face Detection: Cascaded Classifier



- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - using data from previous stage.
- A 20 feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)

Face Detection: Non-Maximum Suppression

- Thin out the multiple responses.
- Suppress spurious responses.

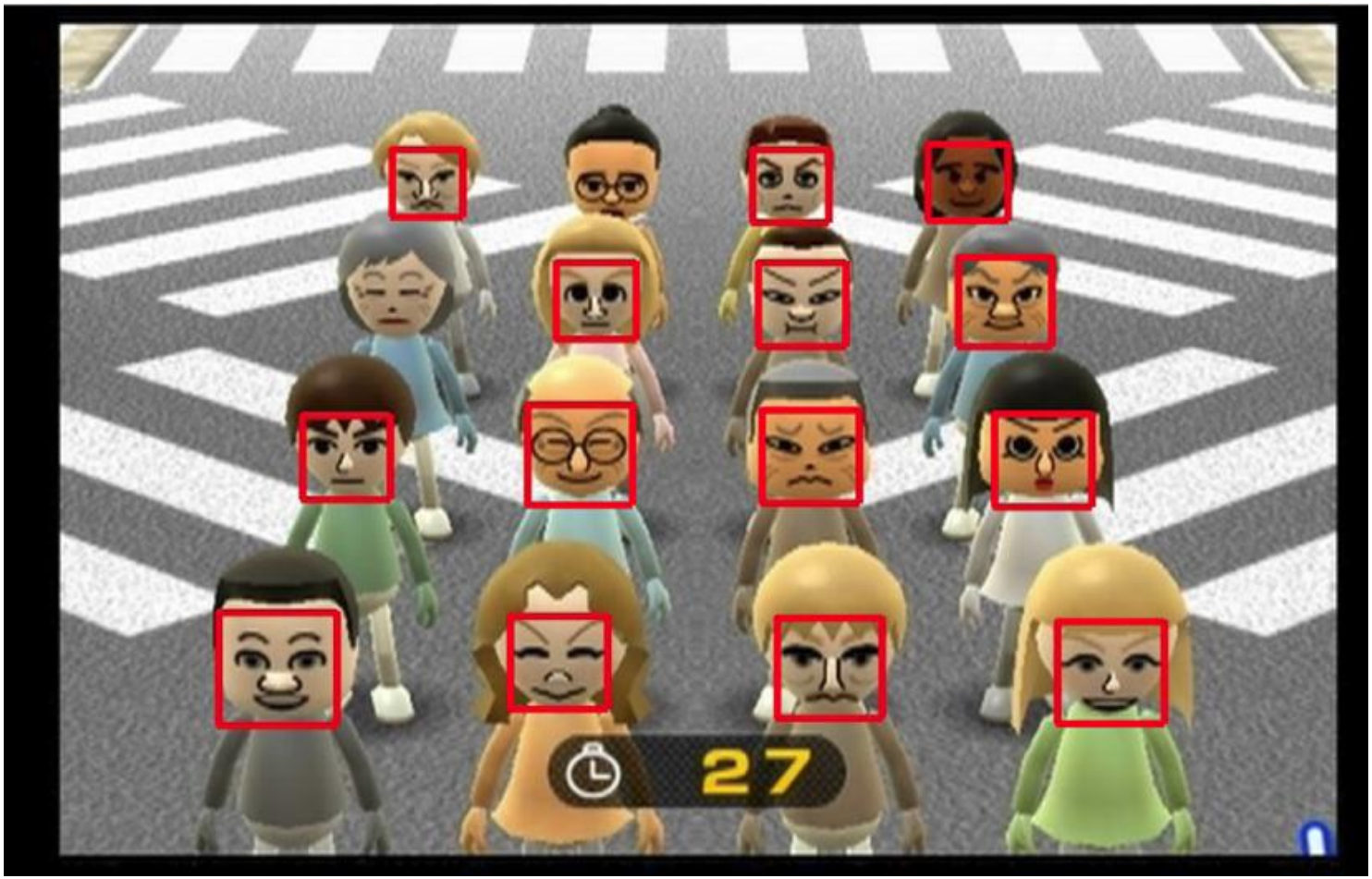
Face Detection

- Good news – OpenCV comes with an implementation of Viola-Jones!
- Training
  - The training face images are provided.
  - opencv_traincascade (supports Harr and LBP features).
  - Details are at http://docs.opencv.org/trunk/doc/user_guide/ug_traincascade.html
- Detection
  - A good reference: http://docs.opencv.org/doc/tutorials/objdetect/cascade_classifier/cascade_classifier.html

Face Detection: OpenCV Functions

- bool CascadeClassifier::load(const string& filename);
  - Loads a classifier from a file.
- void CascadeClassifier::detectMultiScale(const Mat& image, vector<Rect>& objects, double scaleFactor=1.1, int minNeighbors=3, int flags=0, Size minSize=Size(), Size maxSize=Size());
  - Detects objects of different sizes in the input image. The detected objects are returned as a list of rectangles.

Face Detection: Example

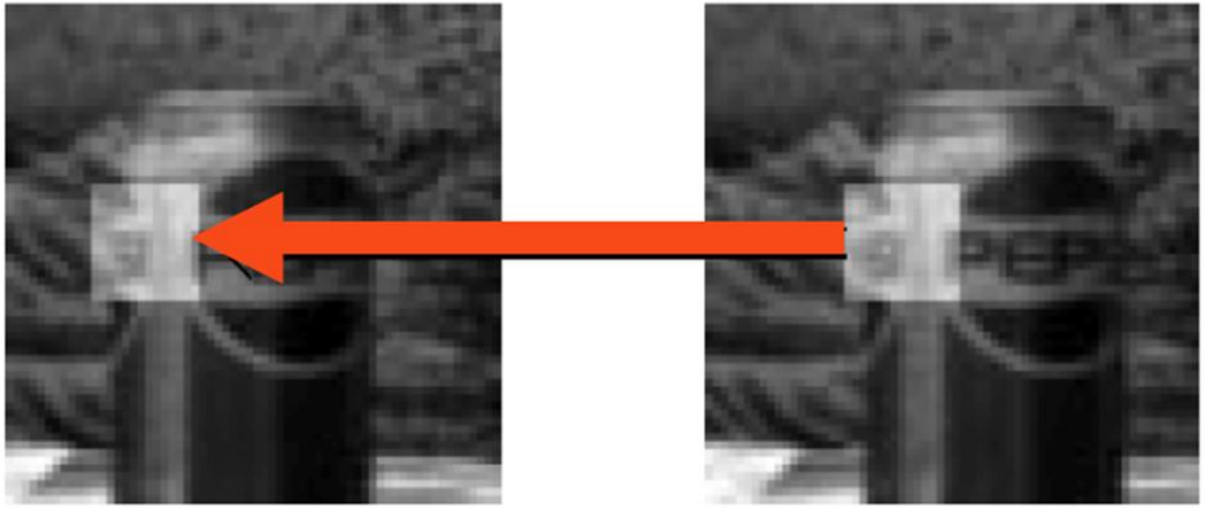


Optical Flow – Theory

- Optical flow: the apparent motion of brightness patterns in the image.
  - Will be covered in Lecture 13 (Nov 8)
  - May not be the same as the actual motion
  - Human vision does optical flow analysis all the time – being aware of movement around them.
- Use cases:
  - Find objects from one frame in other frames.
  - Determine the speed and direction of movement of objects.
  - Determine the structure of the environment.

Optical Flow – Theory (Lucas-Kanade)

- Key assumption
  - Brightness constancy



  - $I(x+u,y+v,t+1)=I(x,y,t)$
  - Image measurements in a small region remain the same although their location may change.

Optical Flow – Theory (Lucas-Kanade)

- From Brightness Constancy
  - $I(x+u, y+v, t+1) = I(x, y, t)$
  - $\Rightarrow I(x+u, y+v, t+1) \approx I(x, y, t) + I_x \cdot u + I_y \cdot v$
  - $\Rightarrow \nabla I \cdot (u, v) = -I_t$
- From Spatial Coherence
  - Assume pixel's neighbors have the same (u,v)
  - If we use a 5x5 window, that gives us 25 equations per pixel

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

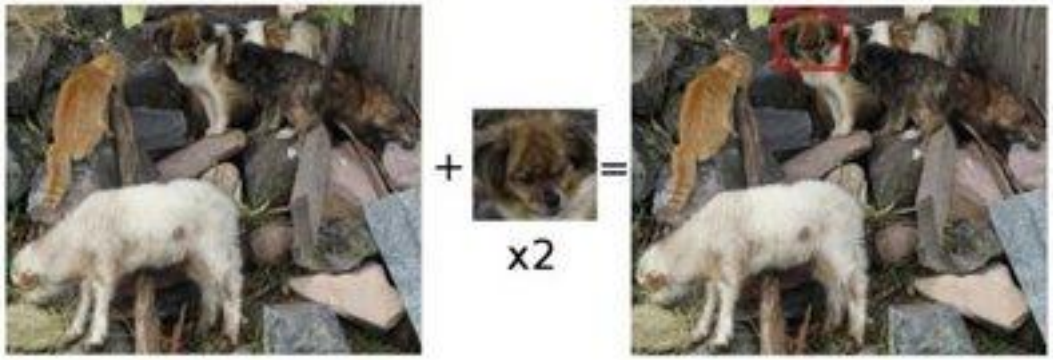
Optical Flow: OpenCV Functions

- void goodFeaturesToTrack(Mat& image, vector<Point2f>& corners, int maxCorners, double qualityLevel, double minDistance);
  - Determines strong corners on an image.
- void cornerSubPix(Mat& image, vector<Point2f>& corners, Size winSize, Size zeroZone, TermCriteria criteria);
  - Refines the corner locations.
- void calcOpticalFlowPyrLK(Mat& prevImg, Mat& nextImg, vector<Point2f>& prevPts, vector<Point2f>& nextPts, vector<uchar>& status, vector<float>& err);
  - Calculates an optical flow for a sparse feature set using the iterative Lucas-Kanade method with pyramids.

Optical Flow - Example

Template Matching

- Template matching is a technique for finding areas of an image that match (are similar) to a template image (patch).



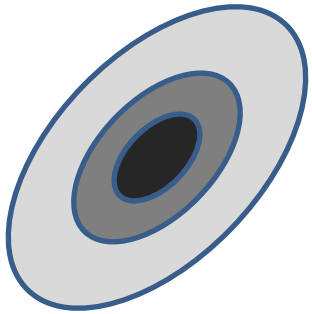
- We need two primary components:
  - Source image (I): The image in which we expect to find a match to the template image.
  - Template image (T): The patch image which will be compared to the template image.
- Our goal is to detect the highest matching area.

Template Match: OpenCV Functions

- void matchTemplate(Mat& image, Mat& templ, Mat& result, int method);
  - The function slides through image, compares the overlapped patches of size w*h against templ using the specified method and stores the comparison results in result.
  - After the function finishes the comparison, the best matches can be found as global minimums (when CV_TM_SQDIFF was used) or maximums (when CV_TM_CCORR or CV_TM_CCOEFF was used) using the minMaxLoc() function.
- void minMaxLoc(Mat& src, double* minVal, double* maxVal=0, Point* minLoc=0, Point* maxLoc=0, InputArray mask=noArray());
  - Finds the global minimum and maximum in an array.

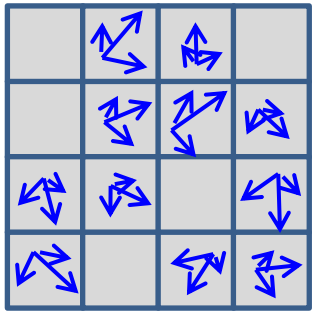
Local Features

Detection:



- Detectors available
- SIFT
- SURF
- FAST
- ORB
- MSER

Description:



- Descriptors available
- SIFT
- SURF
- ORB
- BRIEF
- FREAK

Local Features Sample Code

- Detect interest points and extract descriptors from the given image.

```
cv::Mat img = cv::imread(“test.jpg”);

cv::Ptr<cv::FeatureDetector> detector = cv::createDetector( “FAST” );

cv::Ptr<cv::DescriptorExtractor> descriptorExtractor = cv::createDescriptorExtractor( “SURF” );

std::vector<cv::KeyPoint> keypoints;

detector->detect( img, keypoints );

cv::Mat descriptors;

descriptorExtractor->compute( img, keypoints, descriptors );
```

Local Features Sample Code

- Match descriptors from two images.
- Assume that the descriptors of two images are stored in descriptor1 and descriptor2 already.

```
cv::Ptr<cv::DescriptorMatcher> descriptorMatcher =
cv::createDescriptorMatcher( "BruteForce" );

std::vector<cv::DMatch> matches;

descriptorMatcher->add( descriptors2 );

descriptorMatcher->match( descriptors1, matches );
```

Good Luck!