

The background of the slide features a photograph of a university building with a central entrance and columns, framed by trees with yellow autumn leaves. A semi-transparent green rectangular overlay covers the upper portion of the image, serving as a background for the title text.

Introduction to the CSU Baseline Algorithm

David Bolme and Nayeem Teli

Colorado State University

Quick Survey

- Have you used the LRPCA baseline before?
- Are you familiar with the NIST/BEE protocol: Sigsets, Distance Matrix, etc?
- Are you a Python expert?
- Are you new to Python?
- Did you bring a laptop?
- Do you have the eval system already Installed?

Tutorial Goals

- Installation of Python and the CSU Baseline System
- Understanding the Tools: PyVision, Numpy, OpenCV
- Introduction to the Baseline Sytem
- Time with the experts: Question / Answer

System Architecture

Session 3

CSU Face Baseline
(LRPCA / LDA-IR)

Session 2

PyVision
(Face and Eye Detection)

Session 1

PIL

Scipy

OpenCV

Python

System Architecture

Session 3

CSU Face Baseline
(LRPCA / LDA-IR)

Session 2

PyVision
(Face and Eye Detection)

Session 1

PIL

Scipy

OpenCV

Python

Installation - DVD contents

- Parallels virtual machine
- Python, Numpy, Scipy, OpenCV, PIL
 - MacOS mpkg's (64-bit Lion and Snow Leopard)
 - Windows 7 (binary installers)
- CSUBaselineBeta20111004.zip (source)
- TutorialSource

<http://pyvision.sf.net/>

Session 1 Goals

- Walk through the installation process.
- Hands on help.
- Introduction to Python for computer vision.
 - PIL, NumPy, SciPy, OpenCV

Installation DVD / USB

- Install kit for Windows
- Install kit for MacOS (10.6)
- Parallels Virtual Appliance



Installation Requirements



PyVision

- Python
- Python Imaging Library (PIL)
- NumPy and SciPy
- OpenCV (ver 2.2 or 2.3)
- PyVision (Included)

Parallels Virtual Appliance

- Ubuntu 11.04
- python, scipy, numpy, pil,
- Eclipse with pydev&subclipse
- FireFox
- pyvision
- CSU Baseline
- libsvm
- data
- iPython

Quick Introduction Python

Benefits of Python

- Similar syntax as matlab
- Similar functionality to matlab through scipy.
- Interfaces to OpenCV, LibSVM, and many other open source libraries.
- Object Oriented
- Quick and easy prototyping

Indentation / Control

- Indentation determines block structure.
- Use colon ":" instead of braces
- Set your text editor to use spaces instead of tabs.
- Reference Counting / GC

```
def foo(a,b):  
    ''' A function that adds two numbers '''  
    return a + b  
  
# Count from 0 to 9  
i = 0  
while i < 10:  
  
    print "Number:",i,  
  
    if i % 2 == 0:  
        print "even"  
    else:  
        print "odd"  
  
    i = foo(i, 1)
```

Results

```
Number: 0 even  
Number: 1 odd  
Number: 2 even  
Number: 3 odd  
Number: 4 even  
Number: 5 odd  
Number: 6 even  
Number: 7 odd  
Number: 8 even  
Number: 9 odd
```


The “main” script

- Basic script structure.
- Executes from top to bottom.
- “__main__” if statement
- Arguments: sys.argv
- Functions “def”
- Classes “class”
 - “self” parameter

```
def add(a,b):  
    return a + b  
  
class Add:  
    def __init__(self,a,b):  
        self.val = a+b  
  
    def getValue(self):  
        return self.val  
  
if __name__ == '__main__':  
    # execute this code if this  
    # is the main script  
    print "Hello World!!!"  
    print "2 + 3 =",add(2,3)  
    my_obj = Add(2,3)  
    print "Add(2,3)=", my_obj.getValue()
```

File: TutorialMain.py

Results

```
Hello World!!!
```

```
2 + 3 = 5
```

```
Add(2,3)= 5
```


Data and Types

- object - myobj = MyClass()
- int - 1
- float - 1.0
- str / buffer - "Hello World"
- list - [1,2.0,"three",myobj]
- dict - {"key": "val", 203:myobj}

```
if __name__ == '__main__':  
    print "2 + 3 =", 2 + 3  
    print "2. + 3. =", 2. + 3.  
    print "'2' + '3' =", '2' + '3'  
    print "(2,) + (3,) =", (2,) + (3,)  
    print "[2] + [3] =", [2] + [3]  
    print "dictionary:", {'two':2,3:'three',  
    (2,3):5}  
    print "int('2') + 3 =", int('2') + 3  
    print "'2' + 3 =", '2' + 3
```

Results

```
2 + 3 = 5
2. + 3. = 5.0
'2' + '3' = 23
(2,) + (3,) = (2, 3)
[2] + [3] = [2, 3]
dictionary: {3: 'three', (2, 3): 5, 'two': 2}
int('2') + 3 = 5
'2' + 3 =
```

Traceback (most recent call last):

File "/Users/bolme/Documents/workspace/FaceRec/src/experiments/tutorials/
TutorialTypes.py", line 9, in <module>

```
    print "'2' + 3 =", '2' + 3
```

TypeError: cannot concatenate 'str' and 'int' objects

Introspection and Help

- `print` - print object info
- `type(object)` - get the object type.
- `dir()` - list variables, functions, etc in current scope.
- `dir(object)` - list members/methods of object.
- `help(object/module)` - get help on an object, function, or module.

```
import numpy as np

a = np.array([1., 2., 3., 4.])

print a

print type(a)

print dir()

print dir(a)

help(a)
```

Results

```
[ 1.  2.  3.  4.]  
<type 'numpy.ndarray'>  
['__builtins__', '__doc__', '__file__', '__name__', '__package__', 'a', 'np']  
['T', '__abs__', '__add__', '__and__', ..., 'all', 'any', 'argmax', 'argmin',...]  
Help on ndarray object:
```

```
class ndarray(__builtin__.object)  
|   ndarray(shape, dtype=float, buffer=None, offset=0,  
|           strides=None, order=None)  
|  
|   An array object represents a multidimensional, homogeneous array  
|   of fixed-size items.  An associated data-type object describes the  
|   format of each element in the array (its byte-order, how many bytes it  
|   occupies in memory, whether it is an integer, a floating point number,  
|   or something else, etc.)
```


Standard Library Highlights

- Operating System - `os`, `os.path`
- Binary Data - `struct`
- Math - `math`, `cmath`, `random`
- Object Serialization - `pickle`
- XML - `ElementTree`, `dom`, `sax`
- Comma Sep. Value - `csv`
- Database - `bsddb`, `sqlite3`
- Compression: `zlib`, `bz2`, `zipfile`, `tarfile`
- Security: `md5`, `sha`, `ssl`
- Time: `time`, `calendar`, ...
- Multiple CPU: `multiprocessing`
- Networking: `socket`
- Web: `urllib`, `email`, `htmlib`, `ftplib`
- Other: `unittest`, `string`, `copy`, `sys`

Third Party Libraries

- Interfaces to C, Java, Matlab, R ...
- Web services, Databases, Networking, XML ...
- Scientific Computing, Machine Learning, cuda ...
- GUI: wxPython, Qt, Gnome, Cocoa, Windows...
- Bindings to most popular open source resources.

Organization

Session 3

CSU Face Baseline
(LRPCA / LDA-IR)

Session 2

PyVision
(Face and Eye Detection)

Session 1

PIL

Scipy

OpenCV

Python

Face Detection in OpenCV

```
#!/usr/bin/python
"""
```

This program is demonstration for face and object detection using haar-like features.
The program finds faces in a camera image or video stream and displays a red box around them.

Original C implementation by: ?
Python implementation by: Roman Stanchak, James Bowman
"""

```
import sys
import cv
from optparse import OptionParser
```

```
# Parameters for haar detection
# From the API:
# The default parameters (scale_factor=2,
# min_neighbors=3, flags=0) are tuned
# for accurate yet slow object detection. For a
# faster operation on real video
# images the settings are:
# scale_factor=1.2, min_neighbors=2,
# flags=CV_HAAR_DO_CANNY_PRUNING,
# min_size=<minimum possible face size
```

```
min_size = (20, 20)
```

```
image_scale = 2
haar_scale = 1.2
min_neighbors = 2
haar_flags = 0
```

```
def detect_and_draw(img, cascade):
    # allocate temporary images
    gray = cv.CreateImage((img.width, img.height), 8,
1)
    small_img = cv.CreateImage((cv.Round(img.width /
image_scale),
                                cv.Round (img.height /
image_scale)), 8, 1)
```

```
    # convert color input image to grayscale
    cv.CvtColor(img, gray, cv.CV_BGR2GRAY)
```

```
    # scale input image for faster processing
    cv.Resize(gray, small_img, cv.CV_INTER_LINEAR)

    cv.EqualizeHist(small_img, small_img)
```

```
    if(cascade):
        t = cv.GetTickCount()
        faces = cv.HaarDetectObjects(small_img,
cascade, cv.CreateMemStorage(0),
                                haar_scale,
```


Face Detection in OpenCV

```
cascade, cv.CreateMemStorage(0),
                                haar_scale,
min_neighbors, haar_flags, min_size)
    t = cv.GetTickCount() - t
    print "detection time = %gms" % (t/
(cv.GetTickFrequency()*1000.))
    if faces:
        for ((x, y, w, h), n) in faces:
            # the input to cv.HaarDetectObjects
            # bounding box of each face and
            # convert it to two CvPoints
            pt1 = (int(x * image_scale), int(y *
image_scale))
            pt2 = (int((x + w) * image_scale),
int((y + h) * image_scale))
            cv.Rectangle(img, pt1, pt2,
cv.RGB(255, 0, 0), 3, 8, 0)

    cv.ShowImage("result", img)

if __name__ == '__main__':

    print "hello world"

    parser = OptionParser(usage = "usage: %prog
[options] [filename|camera_index]")
```

```
parser.add_option("-c", "--cascade",
action="store", dest="cascade", type="str",
help="Haar cascade file, default %default", default =
"../data/haarcascades/
haarcascade_frontalface_alt.xml")
(options, args) = parser.parse_args()

print "load cascade"
cascade = cv.Load(options.cascade)

print "Print help"
if len(args) != 1:
    parser.print_help()
    sys.exit(1)

input_name = args[0]
print input_name
if input_name.isdigit():
    capture =
cv.CreateCameraCapture(int(input_name))
else:
    capture = None

cv.NamedWindow("result", 1)

if capture:
    frame_copy = None
```

Face Detection in OpenCV

```
while True:
    frame = cv.QueryFrame(capture)
    if not frame:
        cv.WaitKey(0)
        break
    if not frame_copy:
        frame_copy =
cv.CreateImage((frame.width, frame.height),
cv.IPL_DEPTH_8U, frame.nChannels)
    if frame.origin == cv.IPL_ORIGIN_TL:
        cv.Copy(frame, frame_copy)
    else:
        cv.Flip(frame, frame_copy, 0)

    detect_and_draw(frame_copy, cascade)

    if cv.WaitKey(10) >= 0:
        break
else:
    image = cv.LoadImage(input_name, 1)
    detect_and_draw(image, cascade)
    cv.WaitKey(0)

cv.DestroyWindow("result")
```


Face Detection Demo

- PyVision Philosophy:
 - PyVision is designed for researchers.
 - Algorithms should have simple interfaces and intelligent defaults.
 - Support standard datatypes.
 - Using OpenCV, SciPy, and PIL together should be easy.
 - Results should be easy to understand and debug.

```
import pyvision as pv
import pyvision.face.CascadeDetector as cd

if __name__ == '__main__':

    detector = cd.CascadeDetector()

    cam = pv.Webcam()
    while True:
        frame = cam.query()
        rects = detector(frame)
        for rect in rects:
            frame.annotateRect(rect)
        frame.show()
```

File: TutorialFaceDetect.py

Eye Detection

- Read image from disk:
pv.Image().
- bw image to make annotations stand out.
- Thicker detection rectangle using Polygon and width=4.
- Also detect eyes.

```
import pyvision as pv
import pyvision.face.CascadeDetector as cd
import pyvision.face.FilterEyeLocator as ed

face_detect = cd.CascadeDetector()
eye_detect = ed.FilterEyeLocator()

im = pv.Image("face.png", bw_annotate=True)

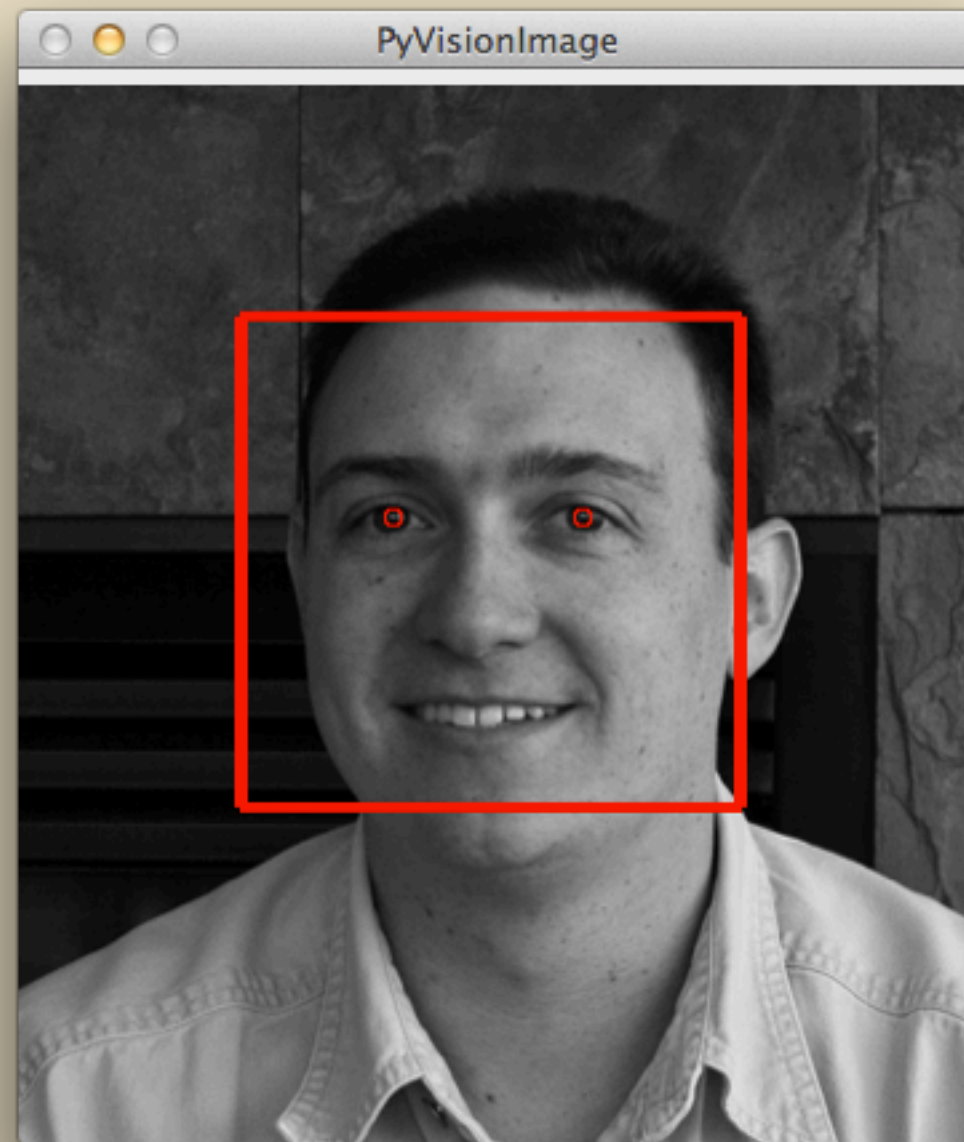
faces = face_detect(im)
eyes = eye_detect(im, faces)

for face, eye1, eye2 in eyes:
    im.annotatePolygon(face.asPolygon(),
                       width=4)
    im.annotatePoints([eye1, eye2])

im.show(delay=0)
```

File: TutorialEyeDetect.py

Result



What PyVision Provides

- Read and convert common data types: video, image, matrix, rects, points, ...
- Common computer vision functions: preprocessing, transforms, detectors, interest points, motion detection, surf, lda, pca, svm, ...
- Analysis tools: Annotation, Plots, Logs, Montage...

PyVision Directory Structure

Points, Rects, Images, Videos, ...

- `pv.Point` - A point (x,y,[z,[w]])
- `pv.Rect` - A rect (x,y,w,h)
- `pv.Image` - JPG, PNG, TIF, ...
- `pv.Video` - AVI, MOV, M4V, ...
- `pv.Webcam` - USB Webcams
- `pv.FFMPEGVideo`
- `pv.VideoFromFileList`
- `pv.VideoFromImages`

Affine Transformations

- `affine = pv.AffineTransform(matrix,size)`
- `new_im = affine(old_im)`
- `new_pts = affine(old_pts)`
- `both = affine1*affine2`
- `affine.invert(pts)`
- Helper Functions:
`pv.AffineFromPoints,`
`pv.AffineFromRect,`
`pv.Affine[Scale,Rotate,Trans...]`
- `pv.PerspectiveTransform`

```
import pyvision as pv

if __name__ == '__main__':
    im = pv.Image("face.png")
    eye1, eye2 = pv.Point(140,165), ...
    out1, out2 = pv.Point(64,128), ...

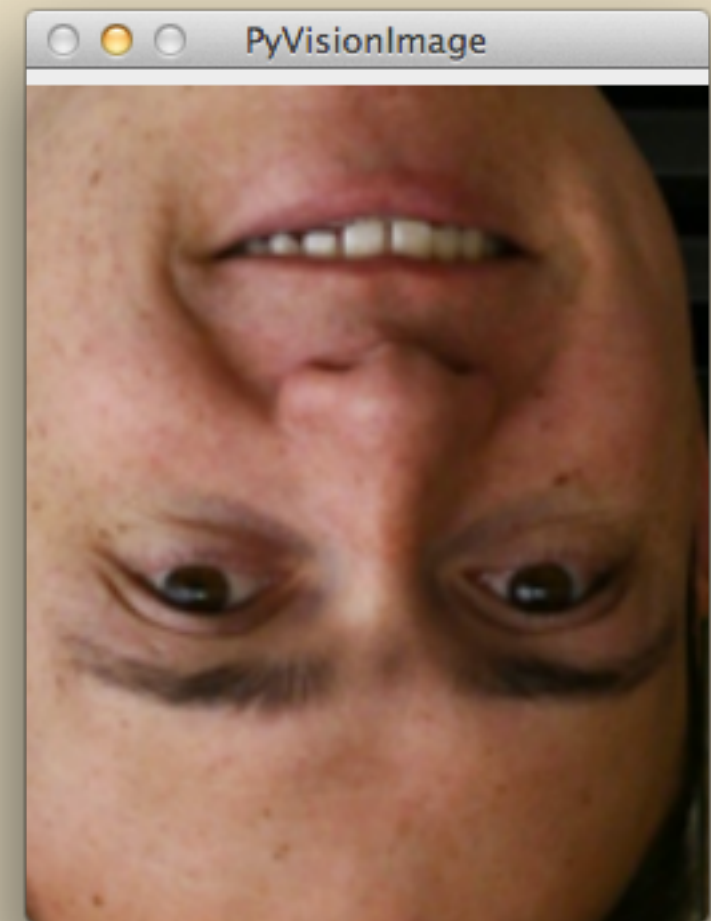
    im.annotatePoints([eye1, eye2])
    im.show(delay=0)

    affine = pv.AffineFromPoints(eye1, eye2,
                                out1, out2, (256, 320))
    tile = affine(im)
    tile.show(delay=0)

    affine = pv.AffineRotate(3.1415, (256, 320),
                             center=pv.Point(128, 160))*affine;
    tile = affine(im)
    tile.show(delay=0)
```

File: TutorialAffine.py

Results



Tools For Understanding Vision

Image Annotation

- Implemented in PIL.
- Annotate images with points, rects, circles, ellipses, polygons, lines, and labels.
- A separate copy of the image is created within the object just for annotations.
- Supports colors and other drawing options: color = "red" or "#FF0000"

```
import pyvision as pv
import scipy as sp
if __name__ == '__main__':
    im = pv.Image(sp.zeros((128,128)))

    pts = [pv.Point(48,55),pv.Point(80,55)]
    im.annotatePoints(pts)

    ellipse = pv.CenteredRect(64,64,96,96)
    im.annotateEllipse(ellipse)

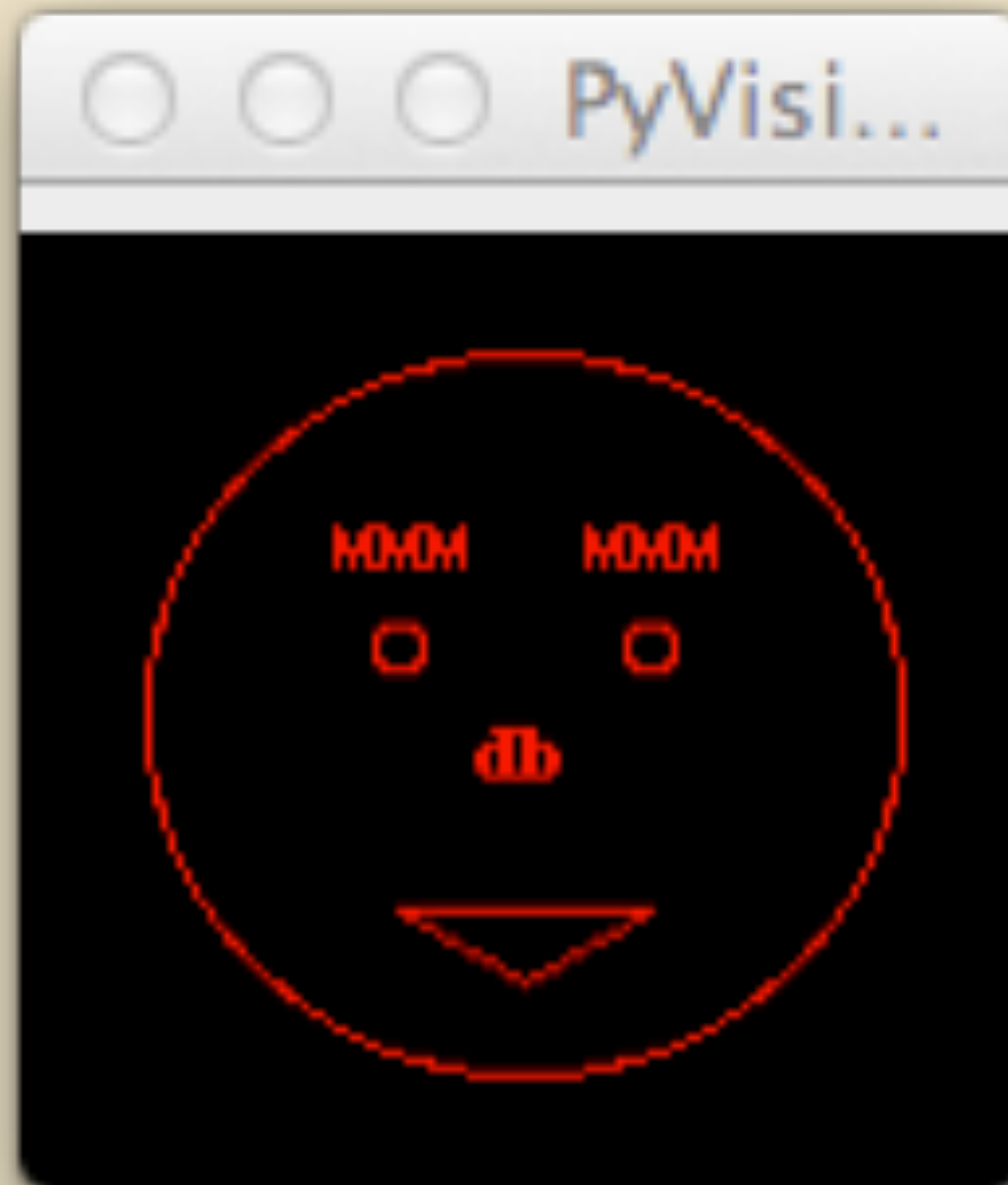
    im.annotatePolygon([pv.Point(48,90),
                        pv.Point(80,90),pv.Point(64,100)])

    im.annotateLabel(pv.Point(40,36), "MMM")
    im.annotateLabel(pv.Point(72,36), "MMM")
    im.annotateLabel(pv.Point(58,64), "db")

    im.show(delay=0)
```

File: --Example.py--

Result



Logs, Tables, Timers

- `pv.ImageLog` - A collection of images, tables, plots, etc that is saved to disk for later analysis.
- `pv.Table` - Tabular data that support pretty printing and csv.
- `pv.Timer` - Time functions and processes.
- `pv.Plot` - line and scatter plots.
- `pv.ImageMontage` and `pv.VideoMontage` - groups of images.

```
import pyvision as pv

ilog = pv.ImageLog()
im = pv.Image("baboon.jpg")
ilog(im, "Baboon")

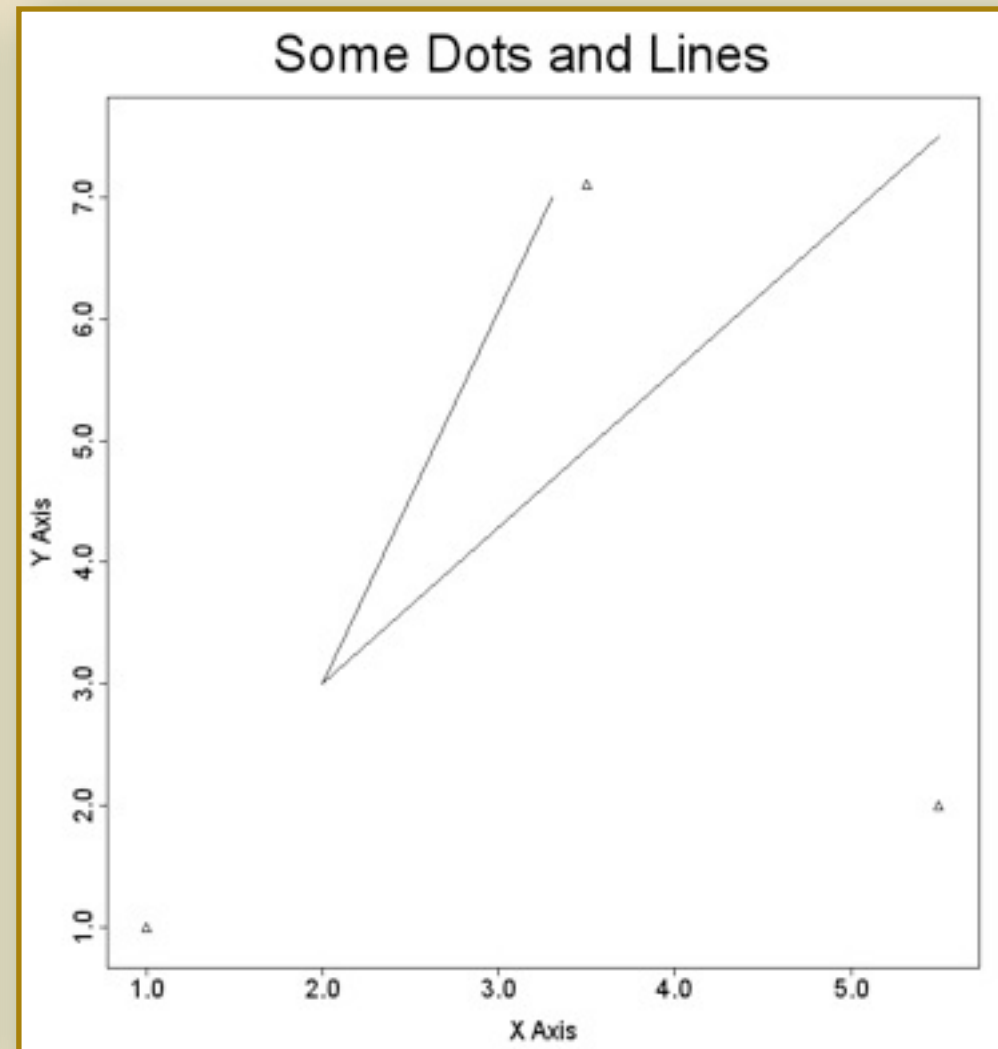
table = pv.Table()
table[1, "image"] = im.filename
table[1, "width"] = im.size[0]
table[1, "height"] = im.size[1]
ilog(table, "ImageData")
print table

plot = pv.Plot(title="Some Dots and Lines");
plot.points([[3.5, 7.1], [1, 1], [5.5, 2]], shape=2)
plot.lines([[5.5, 7.5], [2, 3], [3.3, 7]])
ilog(plot, "MyPlot")

ilog.show()
```

File: TutorialLogsPlots.py

Results



000001_ImageData.csv

Search in Sheet

Home Layout Tables Charts

Edit Font Alignment Number

Paste Calibri (Body) 12 General

B I U A

Align

Cond Form

	A	B	C	D	E	F
1	row	image	width	height		
2	1	baboon.jpg	512	512		
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						

000001_ImageDat

Normal View Ready

PIL, SciPy, and OpenCV

- Use `im.as<Format>` to get PIL, Scipy, and OpenCV images.
- Perform operations using preferred library.
- Convert back using `pv.Image()`
- Note: Scipy format matrices are transposed so that `mat[x,y]` correspond to the x and y image axis.

File: TutorialThresh.py

```
import pyvision as pv
import PIL, cv
ilog = pv.ImageLog()

im = pv.Image("baboon.jpg")

pil = im.asPIL()
gray = pil.convert('L')
thresh = PIL.Image.eval(gray, lambda x:
255*(x>127.5) )
ilog(pv.Image(thresh),"PILThresh")

mat = im.asMatrix2D()
thresh = mat > 127.5
ilog(pv.Image(1.0*thresh),"ScipyThresh")

cvim = im.asOpenCVBW()
dest=cv.CreateImage(im.size,cv.IPL_DEPTH_8U,1)
cv.CmpS(cvim,127.5,dest,cv.CV_CMP_GT)
ilog(pv.Image(dest),"OpenCVThresh")

ilog.show()
```


Results

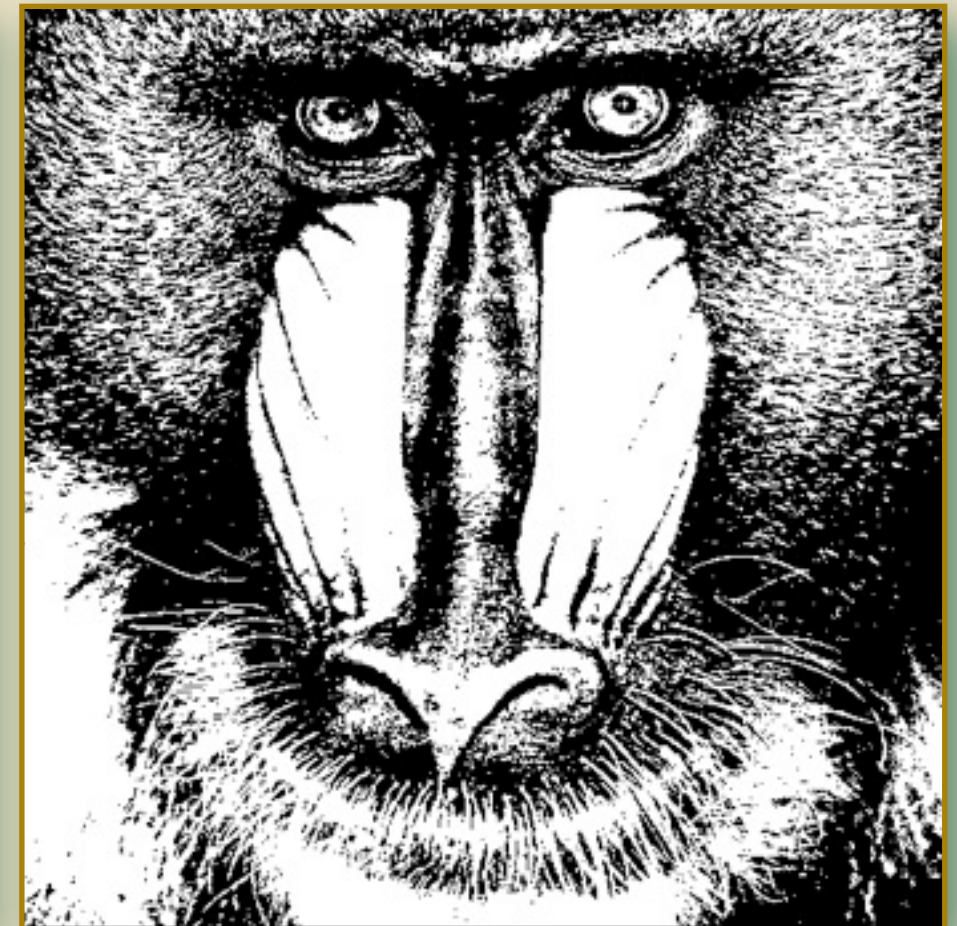
PIL



SciPy



OpenCV



System Architecture

Session 3

CSU Face Baseline
(LRPCA / LDA-IR)

Session 2

PyVision
(Face and Eye Detection)

Session 1

PIL

Scipy

OpenCV

Python

Session 3 Goals

- Introduction to the baseline system
 - LRPCA and LDA-IR
- Overview of the algorithms
- Code Walkthrough
- Test dataset - LFW test

Contributors

- David Bolme
- Yui Man Lui
- Ross Beveridge
- Bruce Draper
- Jonathon Phillips
- Hao Zhang
- Nayeem Teli
- Steve O'Hara

Face Eval Directory Structure

The Dataset: GBU Challenge Problem

The Good - 98%
Easiest Matches



The Bad - 75%
Average Matches



The Ugly - 15%
Hardest Matches



- Challenge problem created to encourage improvement on difficult to match cases (Bad & Ugly)
- All three partitions contain the same number of images from 437 people.
- Most images are “high quality frontal images” collected at University of Notre Dame
- No image pairs are collected on the same day.

Algorithm Interface

- Currently ignores face detection rect.
- getFaceRecord, similarity, and similarity matrix are required for testing.
- addTraining, addCohort, train are only needed if you are using the CSU training framework.
- `__init__`: setup/initialization
- preprocess: alignment, norm

```
class FaceRecognitionAlgorithm:
    def addTraining(self, label, im, rect, l, r):
        raise NotImplementedError()
    def addCohort(self, im, rect, leye, reye):
        pass # NO OP
    def train(self, iolog=None):
        raise NotImplementedError()
    def getFaceRecord(self, im, rect, leye, reye):
        raise NotImplementedError()
    def similarity(self, face_rec1, face_rec2):
        raise NotImplementedError()
    def similarityMatrix(self, probes, targets):
        raise NotImplementedError()

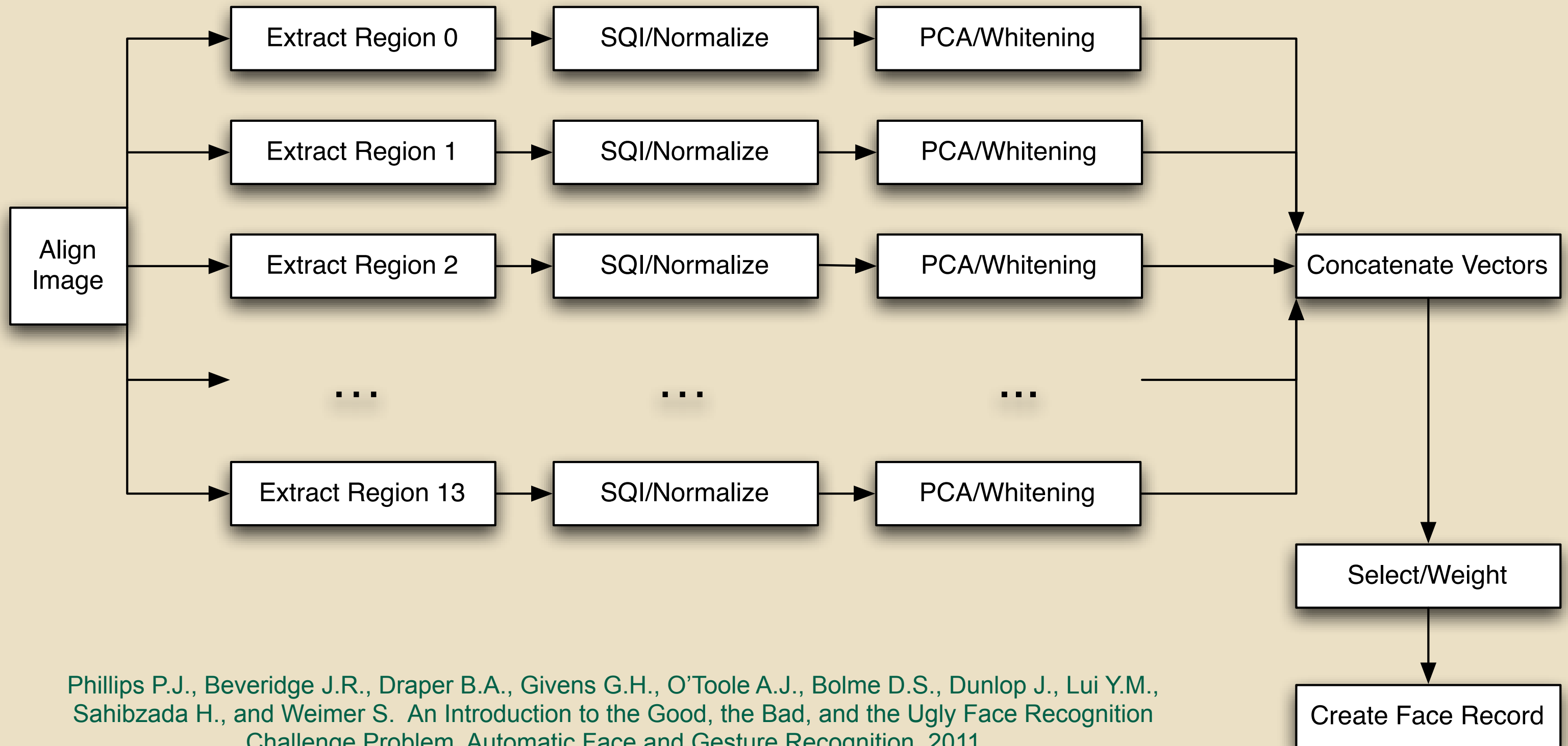
class FaceRecord:
    def __init__(self, rect, leye, reye):
        ...
```

File: `--Example.py--`

Source Code

The Idea: Update and Tune Well-Known Algorithms

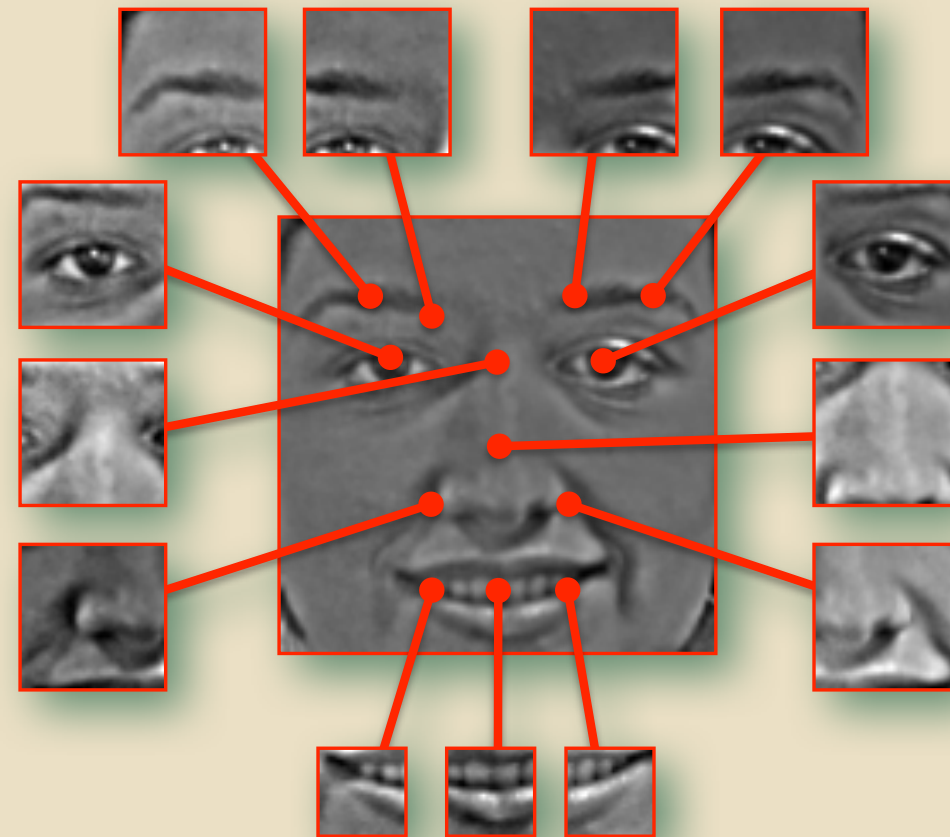
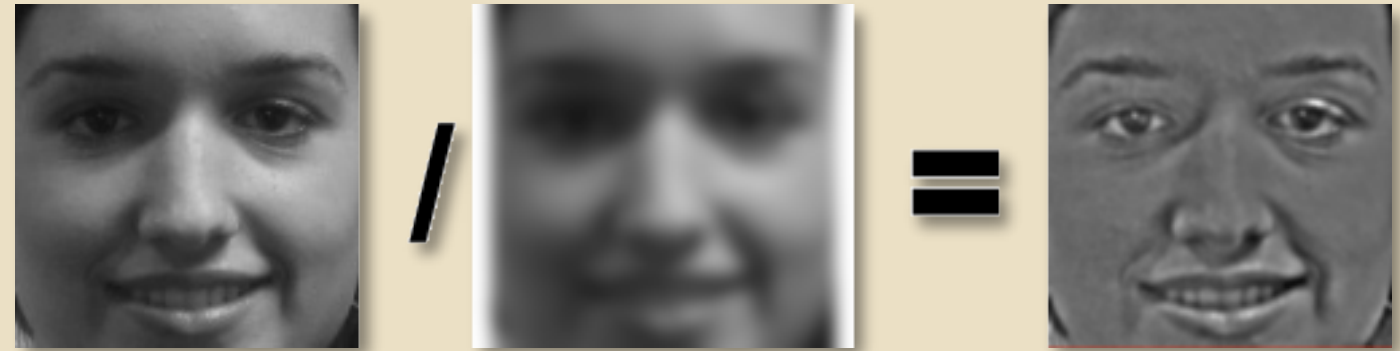
LRPCA Workflow



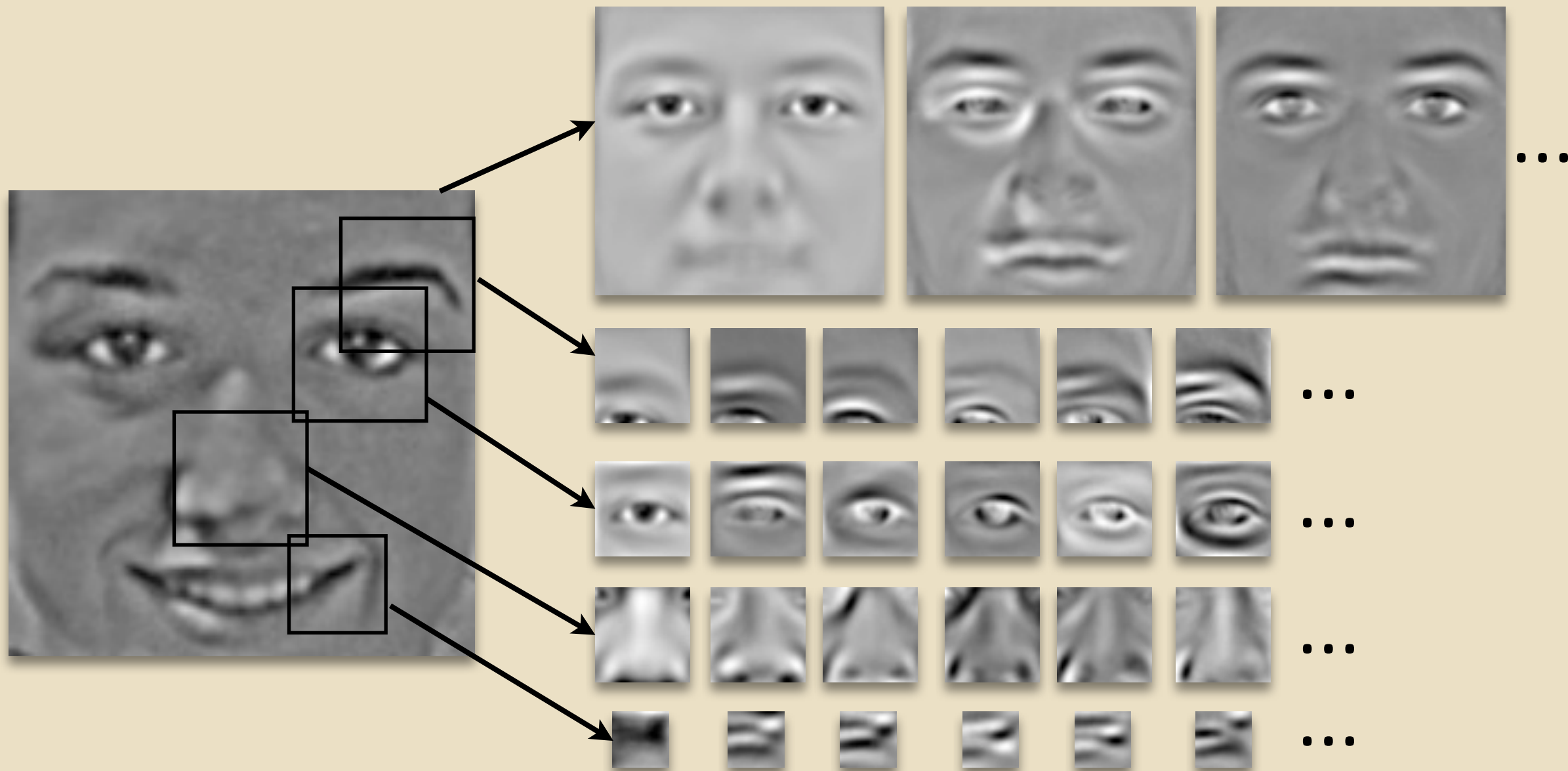
Phillips P.J., Beveridge J.R., Draper B.A., Givens G.H., O'Toole A.J., Bolme D.S., Dunlop J., Lui Y.M., Sahibzada H., and Weimer S. An Introduction to the Good, the Bad, and the Ugly Face Recognition Challenge Problem. Automatic Face and Gesture Recognition. 2011

The System: Local Region PCA

- Modernized version of the PCA algorithm.
- Establishes open-source baseline performance for GBU.
- Includes advances in preprocessing: Self Quotient Image
- Analysis of multiple face regions.

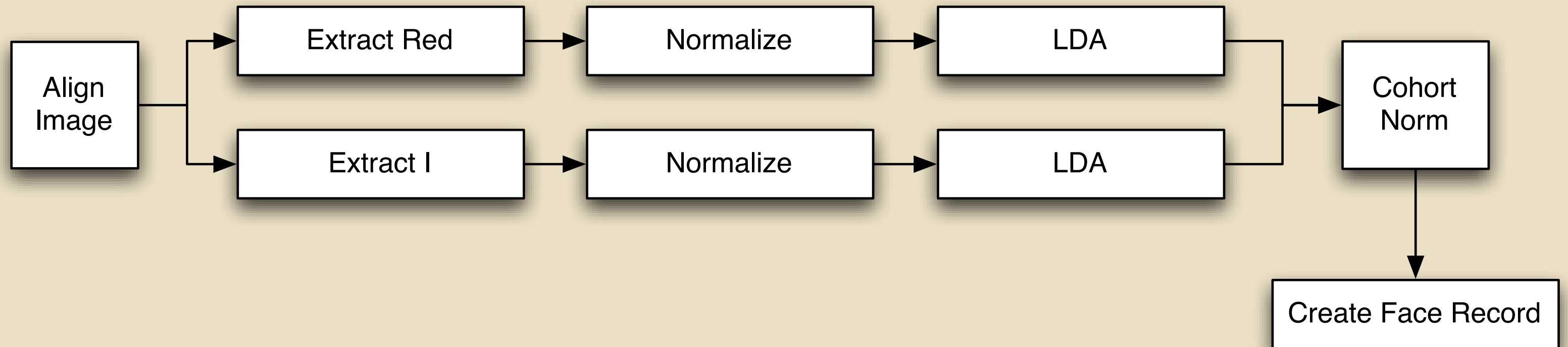


LRPCA Eigenbasis



Source Code

LDA-IR Flow



LDAIR Preprocessing

- 65x75 Image Tiles - 32 Pixels between eyes
- $I = 0.596 * R - 0.275 * G - 0.321 * B$ (from YIQ)
- $R = 1.000 * R + 0.000 * G + 0.000 * B$. (from RGB)
- Log Preprocessing on R channel.
- Z-Norm both channels.

LDA-IR

Red Channel



I Channel



Source Code

Running the Algorithms

- Both algorithm include a trained algorithm stored in a python pickle (.pkl) file.
- Three testing scripts for each algorithm in the “bin” directory. They conform to the BEST interface.
- Setup Image Directories and Eye Coordinates.
- “./scripts/gbu_test_reduced.sh” or “./scripts/gbu_test_manual.sh” runs it all including R plots.
- Test Script: “./scripts/lfw_tests.sh”

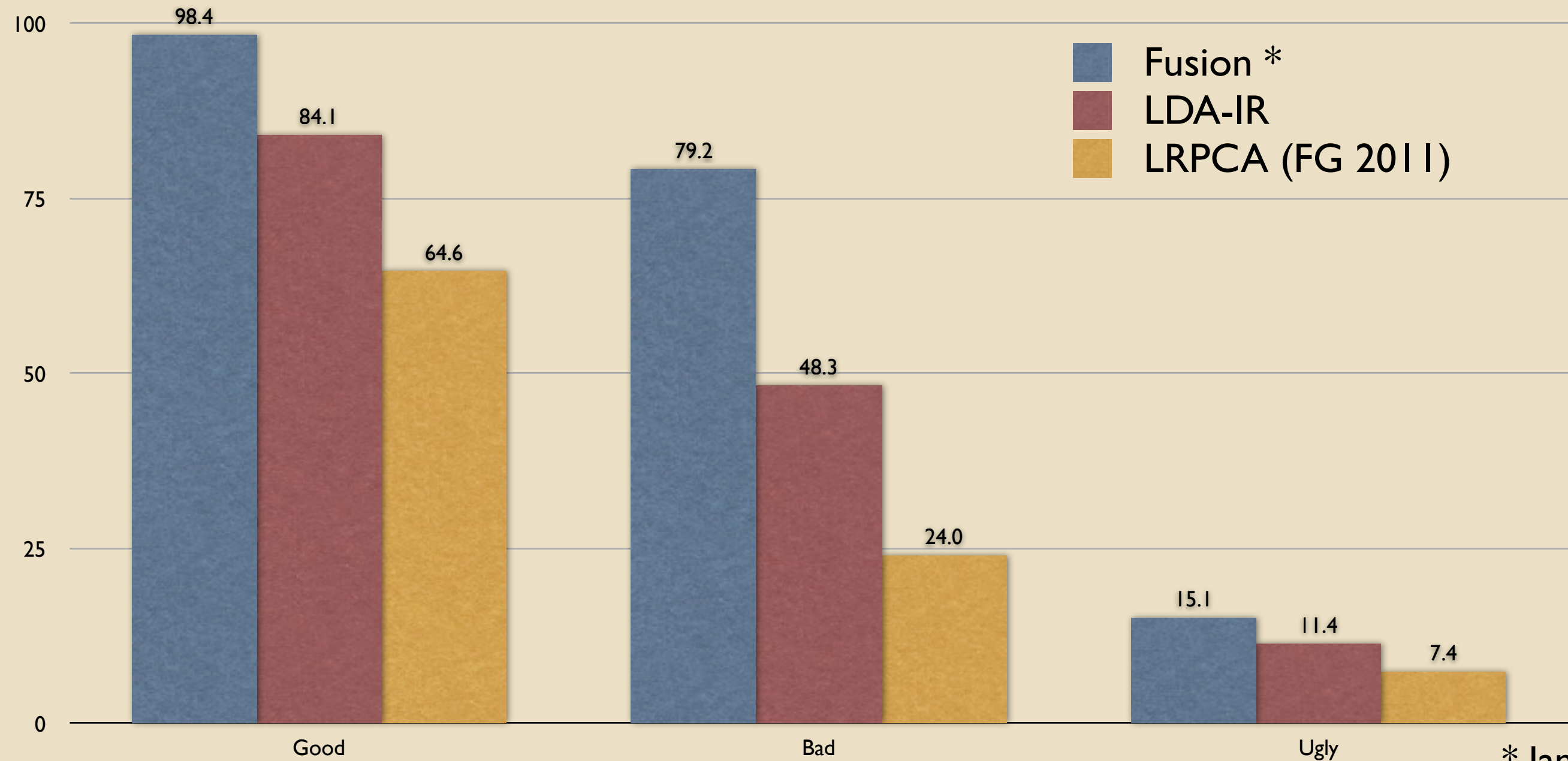
Source Code

Training the Algorithms

- Training is optional.
- LRPCA: “scripts/train_lr pca.sh”
- LDA-IR: “src/facerec2010/train_lda.py”

Source Code

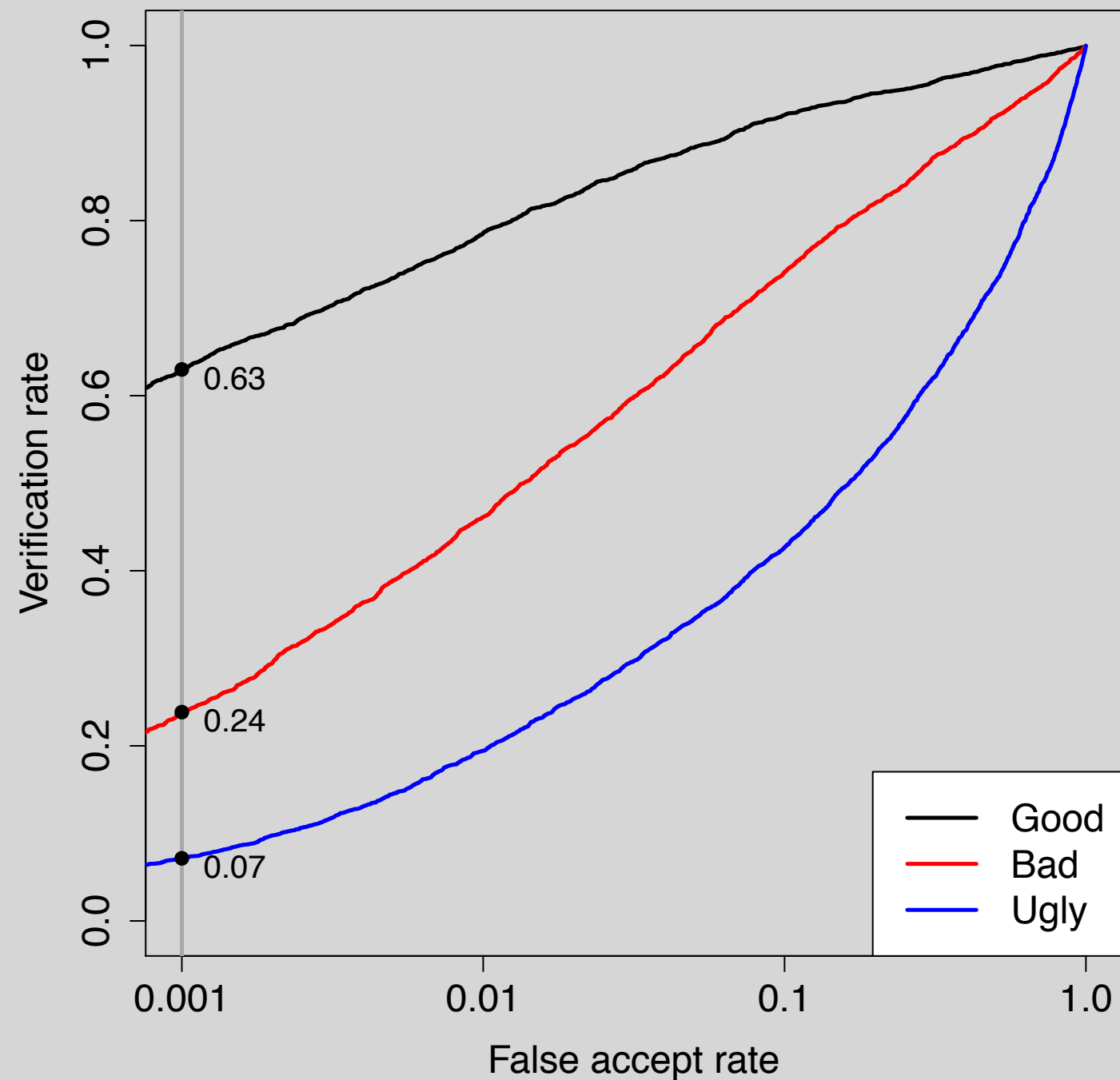
CSU Baseline Performance



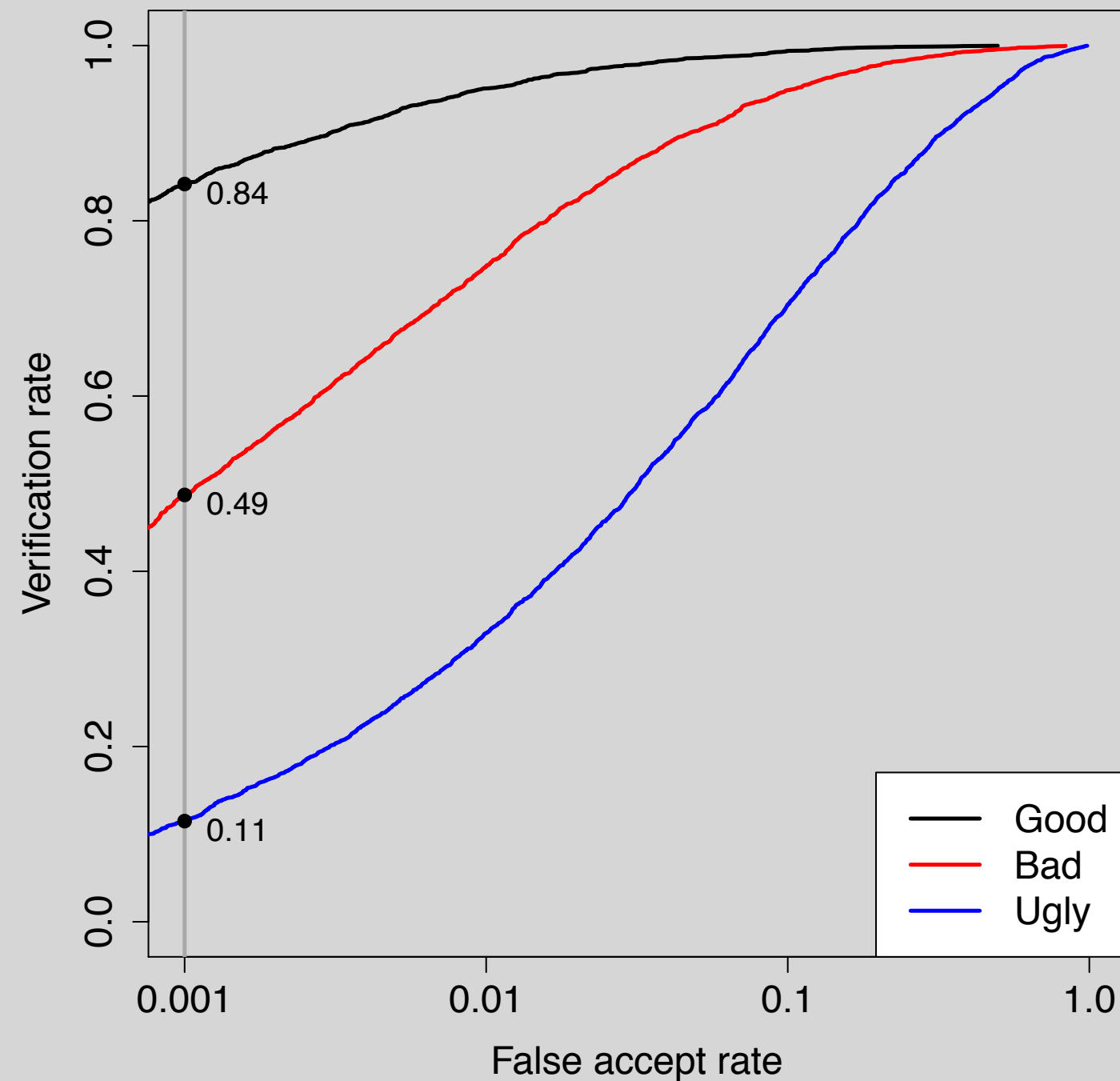
* Jan 2010 Version

CSU Baseline ROCs

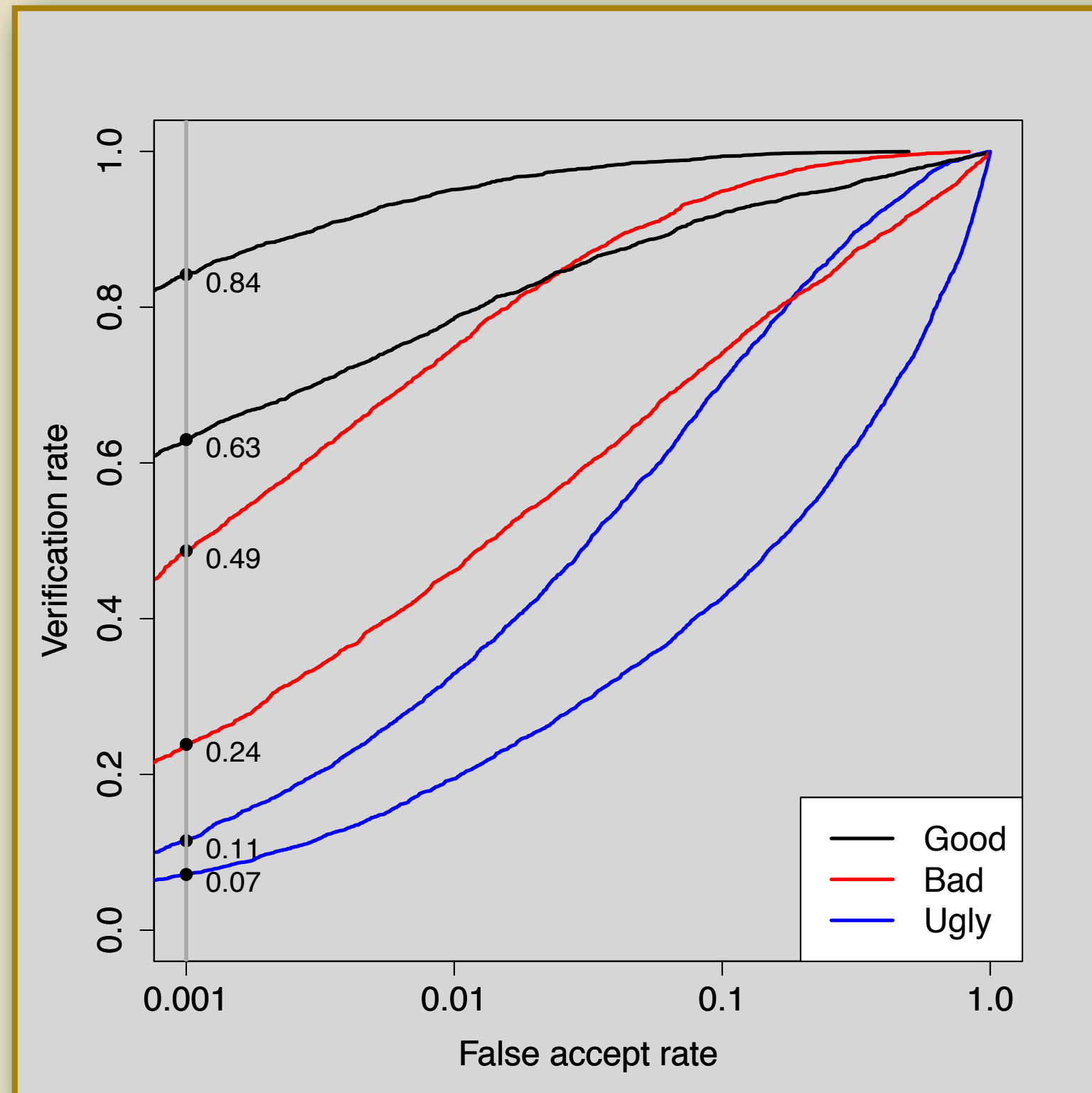
LRPCA



LDA-IR



CSU Baseline ROCs



Automatic Face Detection

- Face Detection - OpenCV Viola and Jones
- Eye Detection - Average of Synthetic Exact Filters (ASEF)

* P. Viola and M.J.Jones. Robust real-time face detection. Int. J. Computer Vision, 57(2):137–154, 2004

**Bolme D.S., Draper B.A., and Beveridge J.R. "Average of Synthetic Exact Filters", Computer Vision and Pattern Recognition, 2009

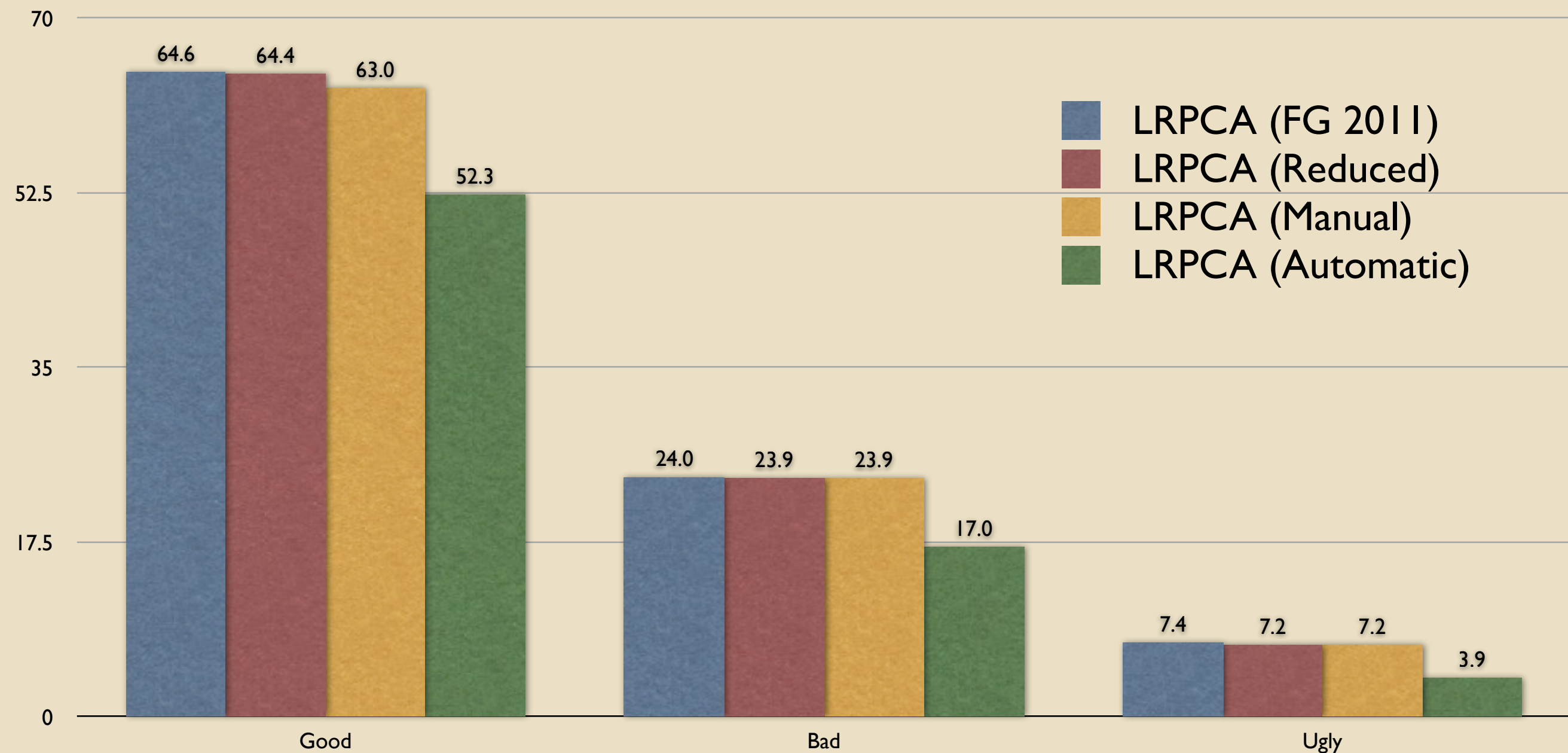
Algorithm Versions

Version	Eye Coordinates	Image Size	Time
Reduced	Pre-normalized	512x512 (75kb)	2 min
Manual	Manual Selection	3008x2000 (6.2 Mb)	10 min
Automatic	Viola&Jones*, ASEF**	3008x2000 (6.2 Mb)	60 min

* P. Viola and M.J.Jones. Robust real-time face detection. Int. J. Computer Vision, 57(2):137–154, 2004

**Bolme D.S., Draper B.A., and Beveridge J.R. "Average of Synthetic Exact Filters", Computer Vision and Pattern Recognition, 2009

Reduced, Manual, and Automatic



Question / Answer