Point-and-Shoot Face Recognition Challenge Users Guide

Hao Zhang and J, Ross Beveridge
zhangh@cs.colostate.edu
ross@cs.colostate.edu

Computer Science Department Colorado State University

http://www.cs.colostate.edu/pasc/

Last Update - June 10, 2013

Abstract

This document is a companion to the paper "The Challenge of Face Recognition from Digital Point-and-Shoot Cameras " article currently under review for the IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS 2013) Conference. We recommend anyone interested in using the Point-and-Shoot Challenge (PaSC) data and associated code first read this article to gain an overview of the challenge problem. This Users Guide is designed to then provide details that should be helpful to those who are preparing to work with the data and software directly.

Overview

Any researcher interested in replicating and then extending results for the Point-and-Shoot Challenged (PaSC) will want to download both the images and the supporting meta-data and code. Because of human subject approval procedures governing the image data, it must be acquired from the University of Notre Dame. Meta-data is available through a website hosted by NIST. All other supporting documentation and code is available through a CSU hosted website. The meta-data may come either through the NIST or CSU sites. This document describes many of the most essential resources that come with this download and relate these to specific experiments and results presented in the paper "". It does not introduce in broad terms the challenge problem nor any results.

Two baseline face recognition algorithms are a major component of the CSU PaSC Software Package. These algorithms were first introduced as part of the Good, Bad and Ugly (GBU) Face Recognition Challenge. These algorithms are:

- Local Region Principal Components Analysis (LRPCA) - Released July 2010
- Cohort Linear Discriminant Analysis (CohortLDA) - released October 2011

The CSU PaSC Software is set up to run on Mac OS, however it should be possible to run the code on other platforms if the dependencies are installed correctly. In order to move to another OS it is helpful have experience with Python programing and unix scripting. The code is also available in two forms. First, as a pre-built virtual machine ready to run. Second, as a bundle of source files and associated meta-data. The VM includes all the files in the source file distribution and has the advantage that auxiliary libraries and packages are already installed and correctly configured.

Everything needed to replicate baseline algorithm experiments presented in "The Challenge of Face Recognition from Digital Point-and-Shoot Cameras" is included with the CSU Software Package with the exception of the data itself. Once the data, acquired from the University of Notre Dame is acquired, the software in this package is ready to go provided a simple linkage between this software and the data is established. What follows are the detailed instructions to guide one through making this happen.

Version Information and Installation Instructions

Here are details about dated software releases and instructions for doing an initial install of the software.

Release List

New releases of this software will be indexed by date. Here are some of the major changes in previous releases. This document is associated with the first release.

06/07/13 first release

Note that actual baseline algorithm code is essentially equivalent modulo a few bug fixes with the earlier releases associated with the GBU.

Installation of Dependencies from Source Files

The algorithms should work with Python 2.6 or later and require a few open source Python modules. Below are the dependencies required for the baseline algorithms with the versions with which the algorithm was tested:

Python 2.7.3 Numpy 1.7.1_1 Scipy 0.12.0 OpenCV 2.4 PIL 1.1.7 PyVision (Included with the current distribution)

Installation on MacOS

As of the most current release we recommend using MacPorts for installation on MacOS, which will keep the dependencies up-to-date. To get MacPorts follow the instructions here:

http://www.macports.org/install.php

The installation steps are as follows:

First install Python 2.7

```
$ sudo port install python27
```

To set the new python as the default run the command below consecutively.

```
$ sudo port install python_select $ sudo python_select python27
```

Install PIL, NumPy, and Scipy. (These depend on the numerical library ATLAS which may take many hours to compile.) Before you install them, please check whether your system already has them. If so, you do not have to install a new one.

```
$ sudo port install py27-pil py27-numpy py27-scipy
```

Install OpenCV with python bindings.

```
$ sudo port install opencv +python27
```

Download and unzip the CSU Baseline Release from here:

http://www.cs.colostate.edu/pasc/

Update the PYTHONPATH environment variable

```
export PYTHONPATH="$PYTHONPATH:<PATH_TO_BASELINE>/PaSC/src/

export PYTHONPATH="$PYTHONPATH:<PATH_TO_BASELINE>/PaSC/src/libraries/
```

Finally, you may want to install R to create ROC plots.

Linux Installation:

All of the software is available for linux and the CSU evaluation system has been run on both Ubuntu and Fedora operating systems successfully. However, due to the differences in Linux vendors and how software packages are installed, Linux is not supported. Binary installers for dependencies like OpenCV, PIL, Numpy and Scipy are available on most linux distributions. We would suggest using the MacOS installation instructions as a guide for getting things up and running on linux although the process will vary on different linux distributions.

Using the Pre-built Virtual Machine (VM)

For those that want to get the system up and running quickly, there is a Virtual Box (https://www.virtualbox.org/) VM available for download. This includes Ubuntu and a installation of PyVision and the full CSU Evaluation System. It also includes many other helpful packages such as Eclipse, iPython, etc.

Due to the consideration of space consumption on the web, we are not using the latest version of Ubuntu. Researchers can choose to upgrade it after downloading VM.

Since the image/video data are very large, it is not suitable to put them directly into VM. We recommend using shared folders where the data lie on the host machine and the VM can still have access to them via a symbolic link. To create a shared folder in VM, do the following:

Boot the Guest operating system in VirtualBox.

Select Devices -> Shared Folders

Click on the Add button and browse for the folder you want to share. Make sure that the name of the share doesn't contain any illegal characters like white spaces. Keep the name as simple as possible. For example, use the name "share".

Now that you have defined a shared folder, it's time to mount it. Create a folder where you will mount it on. For example, in your home folder, use "~/host" for the mount point. Create the folder using the following command:

```
mkdir ~/host
```

Now mount it with the following command:

```
sudo mount -t vboxsf share ~/host
```

The password for our virtual box is pyvision

Then you can see the shared content in "~/host"

Roadmap to the Files in the CSU PaSC Software Package

Here is complete guide to subfolders and files that come as part of the software distribution.

Metadata:

Under metadata/ mask_matrices, there are five mask matrices:

pasc_still_to_still_frontal_mask.mtx - Works with a similarity matrix produced comparing the still query set to the still target set. Includes only match pairs and nonmatch pairs where both images are frontal views of the face.

pasc_still_to_still_mask.mtx - Works with a similarity matrix produced comparing the still query set to the still target set. Includes all possible match and nonmatch combinations.

pasc_video_to_video_control_mask.mtx - Works with a similarity matrix produced comparing all the control videos to themselves. Cases where a video is compared to itself are ignored.

pasc_video_to_video_handheld_mask.mtx - Works with a similarity matrix produced comparing all the handheld videos to themselves. Cases where a video is compared to itself are ignored.

pasc_still_to_video_handheld_mask.mtx - Works with a similarity matrix produced comparing the handheld videos to the still image target set.

Under metadata/ sigsets, there are four signature sets:

pasc_still_query.xml - The query set for the still (single) image portion of the PaSC.

pasc_still_target.xml - The target set for the still (single) image portion of the PaSC.

pasc_video_control.xml - The complete set of 1,401 videos from the high quality control video cameras.

pasc_video_handheld.xml - The complete set of 1,401 videos from the handheld video cameras.

Under metadata/ support, there are four CSV files and a README file:

README.txt - Describes the positions of left and right eye, as well as the origin of the image.

pasc_still_pittpatt_eye_coor.csv - Contains the eye coordinate locations of the still query and target set.

pasc_still_to_still_images_meta_data.csv - Contains the date, location, distance, pose of the person and camera ID when the still query and still target images are taken.

pasc_training_video_pittpatt_detection.csv - Contains the face width, face height, face pose and eye coordinate for individual frames of the training videos on which face/eye detection is successful.

pasc_video_pittpatt_detections.csv - Contains the face width, face height, face pose and eye coordinate for individual frames of the 2,802 videos on which face/eye detection is successful.

Under metadata/ software_test /mask_matrices/, there are three mask matrices:

mask_still_to_still_software_test.mtx - For software test. Works with a similarity matrix produced comparing a small sample of still query set to a small sample of still target set

mask_video_to_video_software_test.mtx - For software test. Works with a similarity matrix produced comparing a small sample of videos to themselves.

mask_still_to_video_software_test.mtx - For software test. Works with a similarity matrix produced comparing a small sample of videos to a small sample of still target set.

Under metadata/software_test/sigsets/, there are three signature sets:

stills_query_software_test.xml - For software test. A sample set of still query images

stills_target_software_test.xml - For software test. A sample set of still target images

videos_software_test.xml - For software test. A sample set of videos.

Output:

Under output/, there are four folders:

csv_files - To place all the CSV files that are used to generate ROC for PaSC .

similarity_matrices - To place all the similarity matrices for PaSC.

roc_curves - To place all the ROC curves for PaSC.

software_test – Contains three folder csv_files, similarity_matrices and roc_curves. When conducting software test, the CSV files that are used to generate ROC, the similarity matrices and the ROC curves will be place in them respectively.

Under output_archive/csv_files, there are seven folders:

pasc_cohortlda_still_to_still.csv – Contains data to plot ROC curves when comparing the still query set to the still target set using CohortLDA algorithm. All match pairs and all nonmatch pairs are included.

pasc_cohortlda_still_to_still_frontal.csv - Contains data to plot ROC curves when comparing the still query set to the still target set using CohortLDA algorithm. Only includes only match pairs and nonmatch pairs where both images are frontal views of the face.

pasc_lrpca_still_to_still.csv - Contains data to plot ROC curves when comparing the still query set to the still target set using LRPCA algorithm. All match pairs and all nonmatch pairs are included.

pasc_lrpca_still_to_still_frontal.csv - Contains data to plot ROC curves when comparing the still query set to the still target set using LRPCA algorithm. Includes only match pairs and nonmatch pairs where both images are frontal views of the face.

pasc_lrpcamax_still_to_video_handheld.csv - Contains data to plot ROC curves when comparing the still target set to 1,401 handheld videos using LRPCA algorithm.

pasc_lrpcamax_video_to_video_control.csv - Contains data to plot ROC curves when comparing 1,401 control videos to themselves using LRPCA algorithm.

pasc_lrpcamax_video_to_video_handheld.csv - Contains data to plot ROC curves when comparing 1,401 handheld videos to themselves using LRPCA algorithm.

Under output_archive/roc_curves/, there are five ROC plots:

pasc_cohortlda_still_to_still_roc.pdf - Contains two ROC curves. One is for comparing the still query set to the still target set using CohortLDA algorithm, where all match pairs and all nonmatch pairs are included. The other is for comparing the still query set to the still target set using CohortLDA algorithm, where only frontal-view match pairs and nonmatch pairs are included.

pasc_lrpca_still_to_still_roc.pdf - Contains two ROC curves. One is for comparing the still query set to the still target set using LRPCA algorithm, where all match pairs and all nonmatch pairs are included. The other is for comparing the still query set to the still target set using LRPCA algorithm, where only frontal-view match pairs and nonmatch pairs are included.

pasc_lrpcamax_still_to_video_handheld_roc.pdf - Contains an ROC curve for comparing the still target set to 1,401 handheld videos using LRPCA algorithm.

pasc_lrpcamax_video_to_video_control_roc.pdf - Contains an ROC curve for comparing 1,401 control videos to themselves using LRPCA algorithm.

pasc_lrpcamax_video_to_video_handheld_roc.pdf - Contains an ROC curve for comparing 1,401 handheld videos to themselves using LRPCA algorithm.

Under output_archive/similarity_matrices/, there are five similarity matrices:

pasc_cohortlda_still_to_still_simi.mtx - A similarity matrix containing similarity scores for comparing the still query set to the still target set using CohortLDA algorithm, where all match pairs and all nonmatch pairs are included.

pasc_lrpca_still_to_still_simi.mtx - A similarity matrix containing similarity scores for comparing the still query set to the still target set using LRPCA algorithm, where all match pairs and all nonmatch pairs are included.

pasc_lrpcamax_still_to_video_handheld_simi.mtx - A similarity matrix containing similarity scores for comparing the still target set to 1,401 handheld videos using LRPCA algorithm

pasc_lrpcamax_video_to_video_control_simi.mtx - A similarity matrix containing similarity scores for comparing 1,401 control videos to themselves using LRPCA algorithm.

pasc_lrpcamax_video_to_video_handheld_simi.mtx - A similarity matrix containing similarity scores for comparing 1,401 handheld videos to themselves using LRPCA algorithm.

Unix Scripts and Source Code:

scripts/ - Contains scripts to run the software test and the whole PaSC. Detail will be described in the section on running scripts.

src/ - Contains the trained LRPCA and CohortLDA algorithms, and also the source code to run the scripts.

A detailed guide to the src code for the baseline algorithms is beyond the scope of this document. The following section provided details necessary to run these algorithms on the PaSC data.

Running Baseline Algorithms and Generating Performance Summary Output

In this section we briefly discuss scripts in general then discuss specific examples of how to run the three major components of the PaSC: still image, video, and still image to video.

About the Executable Scripts

In this Point and Shoot Challenge (PaSC), there are 3 different problems: face recognition comparing still images (still-to-still challenge), face recognition comparing still images against videos (still-to-video challenge), face recognition comparing videos (video-to-video challenge).

For the sill-to-still challenge, we provide 3 different ways to run the algorithm, namely reduced version, manual version and automatic version. Besides these, two baselines algorithms are available: LRPCA and CohortLDA.

- Reduced version: Uses small images that are geometrically normalized to be 512x512 pixels with the eyes located at (192,236) and (320,236) and the filename corresponds to the Recording ID of the image.
- Manual version: Reads manually selected eye coordinates from a CSV file.
- Does automatic face and eye detection on the source images.

For the still-to-video challenge, only LRPCA is used. The still images in this challenge are assumed to be normalized to those from the reduced version.

For the video-to-video challenge, only LRPCA is used.

Running the Algorithms

The first step is to obtain a copy of the PaSC image/video data

The second step is to add the two libraries to PYTHONPATH:

export PYTHONPATH="$PYTHONPATH:PaSC/src/

export PYTHONPATH="$PYTHONPATH:PaSC/src/libraries

The rest of this tutorial assumes that you are running from the command line in the PaSC directory.

$ cd PaSC

In order to run this test you will need to edit the script and set the IMAGE_DIR variable to point at the top level of folder containing all the images. The new interface requires images to be in a directory structure consistent with the names in the sigsets.

Still challenge:

Software test of manual version:

./scripts/software_test/stills/run_stills_manual_software_test.sh src/algorithms/lrpca_manual.py Original_Still_Image_Directory

Instead of using the whole set of images in PaSC, it copies a few still images to output/software_test/original_still_images and conducts face recognition in them using a provided eye coordinate file. Output files include

similarity matrix:

output/software_test/similarity_matrices/lrpca_manual_still_to_still_simi_software_test.mtx

ROC curve:

output/software_test/roc_curves/lrpca_manual_still_to_still_roc_software_test.pdf

output/software_test/csv_files/lrpca_manual_still_to_still_software_test.csv contains data to compute the ROC curve.

output/software_test/temp:

A temp folder stores the intermediate results. If the program crashed in the middle of the process, it can accelerate the next execution given the temp folder. **Please make sure the temp file is deleted or moved to somewhere else before you run another algorithm.**

Software test of automatic version:

./scripts/software_test/stills/run_stills_automatic_software_test.sh src/algorithms/lrpca_automatic.py Original_Still_Image_Directory

Instead of using the whole set of images in PaSC, it copies a few still images to output/software_test/original_still_images and conducts face recognition in them using automatic eye detection. Output files include

similarity matrix:

output/software_test/similarity_matrices/lrpca_automatic_still_to_still_simi_software_test.mtx

ROC curve:

output/software_test/roc_curves/lrpca_automatic_still_to_still_roc_software_test.pdf

csv file containing data to compute the ROC curve:

output/software_test/csv_files/lrpca_automatic_still_to_still_software_test.csv

output/software_test/temp:

A temp folder stores the intermediate results. If the program crashed in the middle of the process, it can accelerate the next execution given the temp folder. **Please make sure the temp file is deleted or moved to somewhere else before you run another algorithm.**

Software test of reduced version:

./scripts/software_test/stills/run_stills_reduced_software_test.sh src/algorithms/lrpca_reduced.py Original_Still_Image_Directory

Instead of using the whole set of images in PaSC, it copies a few still images to output/software_test/original_still_images and generate reduced images using a provided eye coordinate file. The locations of eyes are fixed at (192,236) and (320,236). The reduced images are stored in output/software_test/reduced_still_images. It then conducts face recognition using these reduced images. Output files include

similarity matrix:

/output/software_test/similarity_matrices/lrpca_reduced_still_to_still_simi_software_test.mtx

ROC curve:

output/software_test/roc_curves/lrpca_reduced_still_to_still_roc_software_test.pdf

csv file containing data to compute the ROC curve:

output/software_test/csv_files/lrpca_reduced_still_to_still_software_test.csv

Run the whole still to still challenge:

Run LRPCA using reduced version:

Before running the reduced version, reduced still images should be generated. It can be done using this command:

```
python src/common/generate_reduced_images.py Original_Still_Image_Directory Reduced_Still_Image_Directory
Eye_Coordinate_File Image_Sigset_File
```

It generates images in the Image_Sigset_File to reduced images and put them in Reduced_Still_Image_Directory. Reduced images for both query and target set should be generated.

Then run the following script:

```
./scripts/stills/run_stills_reduced.sh src/algorithms/lrpca_reduced.py Reduced_Still_Image_Directory
```

Output files include

similarity matrix:

output/similarity_matrices/pasc_lrpca_reduced_still_to_still_simi.mtx

ROC curve:

output/roc_curves/pasc_lrpca_reduced_still_to_still.pdf

csv files containing data to compute the ROC curve:

output/csv_files/pasc_lrpca_reduced_still_to_still_frontal.csv

output/csv_files/pasc_lrpca_reduced_still_to_still.csv

Run CohortLDA using reduced version:

As aforementioned, reduced images for query and target set should be generated before running the script. See details in "Run LRPCA using reduced version"

Then run the script:

```
./scripts/stills/run_stills_reduced.sh src/algorithms/cohortlda_reduced.py Reduced_Still_Image_Directory
```

Output files include

similarity matrix:

output/similarity_matrices/pasc_cohortlda_reduced_still_to_still_simi.mtx

ROC curve:

output/roc_curves/pasc_cohortlda_reduced_still_to_still.pdf csv files containing data to compute the ROC curve:

output/csv_files/pasc_cohortlda_reduced_still_to_still_frontal.csv

output/csv_files/pasc_cohortlda_reduced_still_to_still.csv

Run LRPCA using manual version:

./scripts/stills/run_stills_manual.sh src/algorithms/lrpca_manual.py Original_Still_Image_Directory

Output files include

similarity matrix:

output/similarity_matrices/pasc_lrpca_manual_still_to_still_simi.mtx

ROC curve:

output/roc_curves/pasc_lrpca_manual_still_to_still.pdf

csv files containing data to compute the ROC curve:

output/csv_files/pasc_lrpca_manual_still_to_still_frontal.csv

output/csv_files/pasc_lrpca_manual_still_to_still.csv

output/temp:

A temp folder stores the intermediate results. If the program crashed in the middle of the process, it can accelerate the next execution given the temp folder. **Please make sure the temp file is deleted or moved to somewhere else before you run another algorithm.**

Run CohortLDA using manual version:

./scripts/stills/run_stills_manual.sh src/algorithms/cohortlda_manual.py Original_Still_Image_Directory

Output files include

similarity matrix:

output/similarity_matrices/pasc_cohortlda_manual_still_to_still_simi.mtx

ROC curve:

output/roc_curves/pasc_cohortlda_manual_still_to_still.pdf

csv files containing data to compute the ROC curve:

output/csv_files/pasc_cohortlda_manual_still_to_still_frontal.csv

output/csv_files/pasc_cohortlda_manual_still_to_still.csv

output/temp:

A temp folder stores the intermediate results. If the program crashed in the middle of the process, it can accelerate the next execution given the temp folder. **Please make sure the temp file is deleted or moved to somewhere else before you run another algorithm.**

Run LRPCA using automatic version:

```
./scripts/stills/run_stills_automatic.sh src/algorithms/lrpca_automatic.py Original_Still_Image_Directory
```

Output files include

similarity matrix:

output/similarity_matrices/pasc_lrpca_automatic_still_to_still_simi.mtx

ROC curve:

output/roc_curves/pasc_lrpca_automatic_still_to_still.pdf

csv files containing data to compute the ROC curve:

output/csv_files/pasc_lrpca_automatic_still_to_still_frontal.csv

output/csv_files/pasc_lrpca_automatic_still_to_still.csv

output/temp:

A temp folder stores the intermediate results. If the program crashed in the middle of the process, it can accelerate the next execution given the temp folder. **Please make sure the temp file is deleted or moved to somewhere else before you run another algorithm.**

Run CohortLDA using automatic version:

```
./scripts/stills/run_stills_automatic.sh src/algorithms/cohortlda_automatic.py Original_Still_Image_Directory
```

Output files include

similarity matrix:

output/similarity_matrices/pasc_cohortlda_automatic_still_to_still_simi.mtx

ROC curve:

output/roc_curves/pasc_cohortlda_automatic_still_to_still.pdf

csv files containing data to compute the ROC curve:

output/csv_files/pasc_cohortlda_automatic_still_to_still_frontal.csv

output/csv_files/pasc_cohortlda_automatic_still_to_still.csv

output/temp:

A temp folder stores the intermediate results. If the program crashed in the middle of the process, it can accelerate the next execution given the temp folder. **Please make sure the temp file is deleted or moved to somewhere else before you run another algorithm.**

Video challenge:

Software test:

./scripts/software_test/video/run_video_software_test.sh Video_Directory

Instead of using the whole set of videos in PaSC, it copies a few videos to output/software_test/example_videos and computes the similarity of every pair of them.

Output files include

Extracted frames stored in:

output/software_test/extracted_frames

Reduced images of these frames stored in:

output/software_test/reduced_frames

One signature set created for each video using all the available frames in it:

output/software_test/frame_sigset

LRPCA processed frames for each video stored in:

output/software_test/processed_frames

all frame-pair comparisons for every pair of videos stored in:

output/software_test/block_matrices

similarity matrix:

output/software_test/similarity_matrices/lrpca_max_video_to_video_simi_software_test.mtx

ROC curve:

output/software_test/roc_curves/lrpca_max_video_to_video_roc_software_test.pdf

csv file containing data to compute the ROC curve:

output/software_test/csv_files/lrpca_max_video_to_video_roc_software_test.csv

Run the whole video to video challenge:

Warning: This execution is very time consuming. It may take more than a week if running on only one machine. It is recommend that researchers distribute the task to multiple machines.

Run handheld video to handheld video challenge:

./scripts/video/run_video_handheld.sh Video_Directory

Output files include

Extracted frames stored in:

output/extracted_frames

Reduced images of these frames stored in:

output/reduced_frames

One signature set created for each video using all the available frames in it:

output/frame_sigset

LRPCA processed frames for each video stored in:

output/processed_frames

all frame-pair comparisons for every pair of videos stored in:

output/block_matrices

similarity matrix:

output/similarity_matrices/pasc_lrpcamax_video_to_video_handheld_simi.mtx

ROC curve:

output/roc_curves/pasc_lrpcamax_video_to_video_handheld_roc.pdf

csv file containing data to compute the ROC curve:

output/csv_files/pasc_lrpcamax_video_to_video_handheld.csv

Run control video to control video challenge:

./scripts/video/run_video_control.sh Video_Directory

Output files include

Extracted frames stored in:

output/extracted_frames

Reduced images of these frames stored in:

output/reduced_frames

One signature set created for each video using all the available frames in it:

output/frame_sigset

LRPCA processed frames for each video stored in:

output/processed_frames

all frame-pair comparisons for every pair of videos stored in:

output/block_matrices

similarity matrix:

output/similarity_matrices/pasc_lrpcamax_video_to_video_control_simi.mtx

ROC curve:

output/roc_curves/pasc_lrpcamax_video_to_video_control_roc.pdf

csv file containing data to compute the ROC curve:

output/csv_files/pasc_lrpcamax_video_to_video_control.csv

Still to Video challenge:

Software test:

./scripts/software_test/still2video/run_still2video_software_test.sh Video_Directory **Original_Still_Image_Directory**

Instead of using the whole set of videos and still images in PaSC, it copies a few videos to output/software_test/example_videos and a few still imaegs to output/software_test/example_still. Then it computes the similarity of every pair of a still image and a video.

Output files include

Reduced still images stored in:

output/software_test/reduced_still_images

Extracted frames stored in:

output/software_test/extracted_frames

Reduced images of these frames stored in:

output/software_test/reduced_frames

One signature set created for each video using all the available frames in it:

output/software_test/frame_sigset

LRPCA processed frames for each video stored in:

output/software_test/processed_frames

LRPCA processed still images stored in:

output/software_test/processed_stills

similarity matrix:

output/software_test/similarity_matrices/lrpca_max_still_to_video_simi_software_test.mtx

ROC curves:

output/software_test/roc_curves/lrpca_max_still_to_video_roc_software_test.pdf

csv file containing data to compute the ROC curve:

output/software_test/csv_files/lrpca_max_still_to_video_roc_software_test.csv

Run the whole still to video challenge:

Run still to handheld video challenge:

./scripts/still2video/run_video_still2handheld.sh Video_Directory **Original_Still_Image_Directory**

Note that the still image directory should contain still images which are already reduced.

Output files include

Extracted frames stored in:

output/extracted_frames

Reduced images of these frames stored in:

output/reduced_frames

One signature set created for each video using all the available frames in it:

output/frame_sigset

LRPCA processed frames for each video stored in:

output/processed_frames

LRPCA processed still images stored in:

output/processed_stills

similarity matrix:

output/similarity_matrices/pasc_lrpcamax_still_to_video_handheld_simi.mtx

ROC curves:

output/roc_curves/pasc_lrpcamax_still_to_video_handheld_roc.pdf

csv file containing data to compute the ROC curve:

output/csv_files/pasc_lrpcamax_still_to_video_handheld.csv

Preferred Citations:

The best citation when mentioning the LRPCA algorithm is:

P.J. Phillips, J.R. Beveridge, B. Draper, G. Givens, A. O'Toole, D. Bolme, J. Dunlop, Y.M. Lui, H. Sahibzada, and S. Weimer, "An introduction to the good, the bad, & the ugly face recognition challenge problem", Image and Vision Computing, vol. 30, no. 3, pp. 206-216, 2012.

The best citation when mentioning the Cohort LDA algorithm is:

Y.M. Lui, D. Bolme, J.P. Phillips, J.R. Beveridge, and B. Draper, "Preliminary Studies on on the Good, the Bad, and the Ugly Face Recognition Challenge Problem", CVPR Biometrics Workshop, Rhode Island, 2012.

Acknowledgments:

This work was supported by the Technical Support Working Group (TSWG)

Much of the software in this distribution was developed by Hao Zhang and David Bolme with additional support by Yui Man Lui, Mohammad Nayeem Teli, Steve O'Hara and James R. Matey.

Special thanks to the open source community for contributing high quality software without which these algorithms would not be possible. In particular, those who have contributed to the following projects:

OpenCV Python Numpy Scipy Python Image Library