Project Documentation

Face recognition system

### Haim Shalelashvili & Zahi Kfir

# Introduction

A facial recognition system is a [computer application](https://en.wikipedia.org/wiki/Application_software) for automatically [identifying](https://en.wikipedia.org/wiki/Identification_of_human_individuals) or [verifying](https://en.wikipedia.org/wiki/Authentication) a [person](https://en.wikipedia.org/wiki/Person) from a [digital image](https://en.wikipedia.org/wiki/Digital_image) or a [video frame](https://en.wikipedia.org/wiki/Film_frame) from a [video](https://en.wikipedia.org/wiki/Video) source. One of the ways to do this is by comparing selected [facial features](https://en.wikipedia.org/wiki/Face) from the image and a facial [database](https://en.wikipedia.org/wiki/Database_management_system).

It is typically used in [security systems](https://en.wikipedia.org/wiki/Burglar_alarm) and can be compared to other [biometrics](https://en.wikipedia.org/wiki/Biometrics) such as [fingerprint](https://en.wikipedia.org/wiki/Fingerprint) or eye [iris recognition](https://en.wikipedia.org/wiki/Iris_recognition) systems.

Some facial recognition [algorithms](https://en.wikipedia.org/wiki/Algorithms) identify facial features by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face recognition. A probe image is then compared with the face data. One of the earliest successful systems is based on template matching techniques applied to a set of salient facial features, providing a sort of compressed face representation.

In [geometry](https://en.wikipedia.org/wiki/Geometry), an affine transformation  is a [transformation](https://en.wikipedia.org/wiki/Transformation_(function)) of an [affine space](https://en.wikipedia.org/wiki/Affine_space) which preserves [straight lines](https://en.wikipedia.org/wiki/Straight_line) (i.e., all points lying on a line initially still lie on a line after transformation) and ratios of distances between points lying on a straight line (e.g., the [midpoint](https://en.wikipedia.org/wiki/Midpoint) of a line segment remains the midpoint after transformation). It does not necessarily preserve angles or lengths, but does have the property that sets of [parallel](https://en.wikipedia.org/wiki/Parallel) lines will remain parallel to each other after an affine transformation.

# Goal

We will develop all modules of a face recognition system that uses representations compatible with privacy protecting mechanisms

# Project Structure:

This code is divided into couple of folders. Every folder implements a different section in the project. All the algorithms in the software can easily configure from the main configuration file that located in the project folder.

### Configuration File – ConfigurationFile.xml

The configuration file is divided to some nodes:

* Mandatory Folder Paths – Contains all the folders that the project needs. This folders is inserted to the Matlab include path when the function AddMandatoryFolders is called.
* General Configuration – Contain the all the general information and data.
* Sub Project Nodes – Contains the information related to each section in the project.

### Main Program .m:

Test runs all project modules.

### Add Mandatory Folders .m:

Add all the necessary folders and subfolders to project searching paths.

### Face and landmark Detection

Implements the face landmark Detection algorithm. This algorithm uses the "[X. Zhu](http://www.ics.uci.edu/~xzhu), [D. Ramanan](http://www.ics.uci.edu/~dramanan). "Face detection, pose estimation and landmark localization in the wild"" algorithm. We decorated the code to adapt it to our needs.

##### Section Default Configuration:

<FaceAndLandmarkDetection>

<!-- The maximum size of an image (Dimension 1 - Lines) which allowed to participate in the detection algorithm ( used for scaling) -->

<MaxImageSizeDim1>400</MaxImageSizeDim1>

##### </FaceAndLandmarkDetection>

##### Application Programing Interface:

* DetectLandmarks.m - Gets an image and try to detect facial landmarks using the face detector and landmarks finder engine. When landmarks successfully found this functions ouputs the best scored model describing the face in the image.

Affine Transform

Implements the Affine Transformation algorithm. This algorithm uses a predefined model (A structure which holds the detection information about an image) and transforms an input images to have the same landmark positions as the model.

The user can configure the algorithm to run the algorithm on specific landmarks. To change the enabled landmark, call the Choose Enabled Landmarks function and initialize the software using the initialization function located in the recognition section.

##### Section Default Configuration:

<AffineTransform>

<!-- The image from which the affine model built -->

<ModelInputImage>Resources/GT Affine Transformed Colored/s01/01.jpg</ModelInputImage>

<!-- The path to the affine model -->

<AffineModelPath>AffineTransform/AffineModel.mat</AffineModelPath>

<!-- The path to the enabled landmarks matrix - A matrix with zeros in the places where the landmarks disabled and ones in the rest-->

<EnabledLandmarksPath>AffineTransform/EnabledLandmarks.mat</EnabledLandmarksPath>

</AffineTransform>

##### Application Programing Interface:

* ApplyAffineTransform.m - Apply an affine transform on a 3 dimensional Image to match the supplied input AffinedModel. Returns the transformed image and model.
* ChooseEnabledLandmarks.m - Return a vector with 68 cells, every cell as a logical value :
  + - 0 - the landmark is disabled
    - 1 - The landmark is enabled.
* GetPositionsMatrix.m - Returns the position matrix (numberofLandmarks X 2) of a model.

### Train DB

Holds all the code related to creation and initialization of the Training DB

##### Section Default Configuration:

<TrainDB>

<!-- The folder from which the Train DB is built -->

<InputDBFolder>Resources\train\_im</InputDBFolder>

<!-- The path to the trainDb images structure -->

<ImagesFilePath>TrainDB/imagesDB.mat</ImagesFilePath>

<!-- The Path to the Train DB positions file -->

<PositionsTrainDBFilePath>TrainDB/positionsDB.mat</PositionsTrainDBFilePath>

<!-- The path to the trainDb affined images structure -->

<AffinedImagesFilePath>TrainDB/affinedimagesDB.mat</AffinedImagesFilePath>

<!-- The path to the trainDB affined positions structure -->

<AffinedPositionsTrainDBFilePath>TrainDB/affinedpositionsDB.mat</AffinedPositionsTrainDBFilePath>

##### </TrainDB>

##### Application Programing Interface:

* InitializeTrainDB.m - Initialize the train DB files with an up to date train DB:
  + - Call to the Generate Train DB function
    - Update and save the train DB structures

### Appearance Vocabulary

Implements the [Appearance Face Vocabularies](http://www.cs.haifa.ac.il/~rita/vision_lab_course/lab_course.htm#appearance_voc) representation and training algorithm. The user can easily turn on or off the use of this algorithm from the configuration file. When using the code of this algorithm the user need to add the code to the searching path of Matlab ( This section and the Spatial Vocabulary section shares the same interface , thus the software always changes the current algorithm folder) using the UseAppearanceVocabulary function in the Recognition algorithm section.

##### Section Default Configuration:

<AppearanceVocabulary>

<!-- The code folder to include when using the appearance vocabulary -->

<folder>Appearance Vocabulary</folder>

<!-- Enable or disable the use of this vocabulary : true/false-->

<enabled>true</enabled>

<!-- The optimal treshold between the output representation vectors -->

<AppearanceVocabularyTreshold>108</AppearanceVocabularyTreshold>

<!-- The path to the appearance information struct -->

<AppearanceDBPath>TrainDB/appearances.mat</AppearanceDBPath>

<!-- Use Affine transform before using the appearance vocabulary functions : true/false -->

<UseAffineTransform>true</UseAffineTransform>

</AppearanceVocabulary>

##### Application Programing Interface:

* GenerateAppearanceRepresentation1.m - Generate and return appearance vector representation using method 1.
* TrainAppearanceVocabulary.m - Wrapper function to all the appearance vocabulary train functions.

### Spatial Vocabulary

Implements the [Spatial Face Vocabularies](http://www.cs.haifa.ac.il/~rita/vision_lab_course/lab_course.htm#appearance_voc) representation and training algorithms. The user can easily turn on or off the use of this algorithm from the configuration file. When using the code of this algorithm the user need to add the code to the searching path of Matlab ( This section and the Appearance Vocabulary section shares the same interface , thus the software always changes the current algorithm folder) using the UseSpatialVocabulary function in the Recognition algorithm section.

##### Section Default Configuration:

<SpatialVocabulary>

<!-- The Code folder to include when using the spatial vocabulary -->

<folder>Spatial Vocabulary</folder>

<!-- Enable or disable the use of this vocabulary : true/false-->

<enabled>false</enabled>

<!-- Spatial Representation method - On every change the recognition threshold should change also -->

<Method>2</Method>

<!-- The optimal threshold between the output representation vectors, For Method 1 - 50 , For Method 2 - 180-->

<SpatialVocabularyTreshold>180</SpatialVocabularyTreshold>

<!-- The path to the spatial information histogram -->

<HistogramsPath>TrainDB/histograms.mat</HistogramsPath>

##### </SpatialVocabulary>

##### Application Programing Interface:

* GenerateSpatialRepresentation.m - Wrapper function to the GenerateSpatialRepresentations functions.
* TrainSpatialVocabulary.m - Wrapper function to all the spatial vocabulary train functions.

### Recognition Algorithm

Implements the Recognition Algorithm.

##### Section Default Configuration:

<RecognitionAlgorithm>

<!-- The path to the Registered subjects folder -->

<RegisterSubjectsFolder>Resources\Images 4 Registration</RegisterSubjectsFolder>

<!-- The path to the Recognize subjects folder -->

<RecognizeSubjectsFolder>Resources\Images 4 Recognition</RecognizeSubjectsFolder>

##### </RecognitionAlgorithm>

##### Application Programing Interface:

* FindGeneralConfiguration.m - Get a node name and returns the data from the generalConfiguration node in the configuration file.
* FindSubProjectConfiguration.m - Get a subproject name and a node name and returns the data from the configuration file.
* InitializeRecognizer.m - Initalize all the data needed for the algorithm:
  + - Train DB
    - Affine Transform Model
    - Appearance vocabulary train structure.
    - Spatial vocabulary train structure(histograms)
* LoadAndRecognize.m-Load a face image and try to recognize its identity from the all the persons in the vector DB.
* LoadAndReg.m -Load a face image and register it to the vector DB.
* RecognizeInputFolder.m - Try to recognize all the pictures in the input folder.
* RegisterInputFolder.m -Register all the pictures in the input folder.
* ScaledImRead.m - Read an image and returns a scaled version. One must use this function when reading an image related to this project.

### Vectors DB

Contain all the code related to creation and using the Vectors DB (The main DB of vectors which represent each person)

The project was built and tested only on the "MATLAB R2013a" Ide

Results:

The algorithm found to work well on the given train DB when used the shown default configuration:

* + - Using appearance vocabulary (Affined images and train DB)
    - No spatial vocabulary usage

The algorithm could be tested on each configuration simply by editing the configuration file.

Resources:

[1] [X. Zhu](http://www.ics.uci.edu/~xzhu), [D. Ramanan](http://www.ics.uci.edu/~dramanan). "Face detection, pose estimation and landmark localization in the wild" Computer Vision and Pattern Recognition (CVPR) Providence, Rhode Island, June 2012.