# Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Person |
| 23-06-2014 | 0.1 | Initial Document | Zane Bloom |
| 23-06-2014 | 0.1.1 | Added Pipeline Image Processing | Zane Bloom |
| 24-06-2014 | 0.1.2 | Added Image PreProcessing Filter | Zane Bloom |
| 24-06-2014 | 0.1.3 | Added Image FaceDetect Filter | Heelin Mistry |
| 25-06-2014 | 0.1.4 | Added Sampling Filter | Verushka Moodley |
| 29-06-2014 | 0.1.5 | Added Capture | Heelin Mistry |
| 01-07-2014 | 0.2 | Added Persister |  |
| 02-07-2014 | 0.2.1 | Added Database Persister | Verushka Moodley |
| 23-07-2014 | 0.2.2 | Added Domain Objects | Zane Bloom |
| 31-07-2014 | 0.2.3 | Updated UML model and added Facial Recognition | Zane Bloom |

# Functional Requirements

## Pipeline attachments

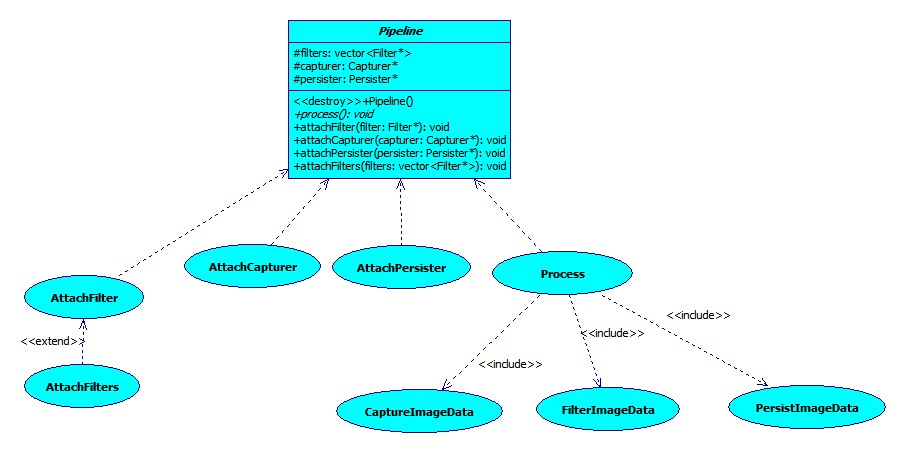
The user must be able to attach a Capturer, Persister and zero to many Filters to the pipeline.

## Pipeline Image Processing

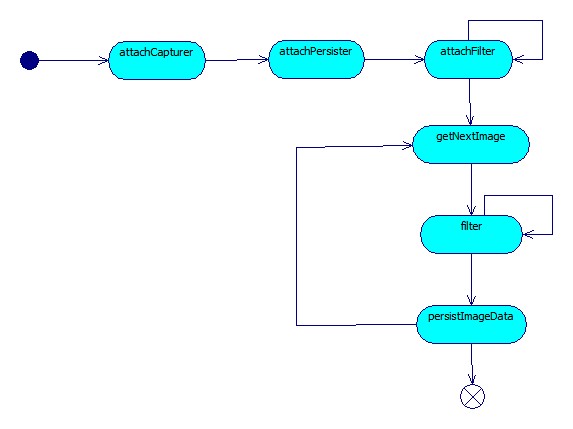
The pipeline must capture an image from a camera, push the image through the attached filters and persist the image and its data for reference later on.

### Service ContractG:\University\2014\COS 301\Main Project\Documents\UML Pictures\PipelineClass.jpg

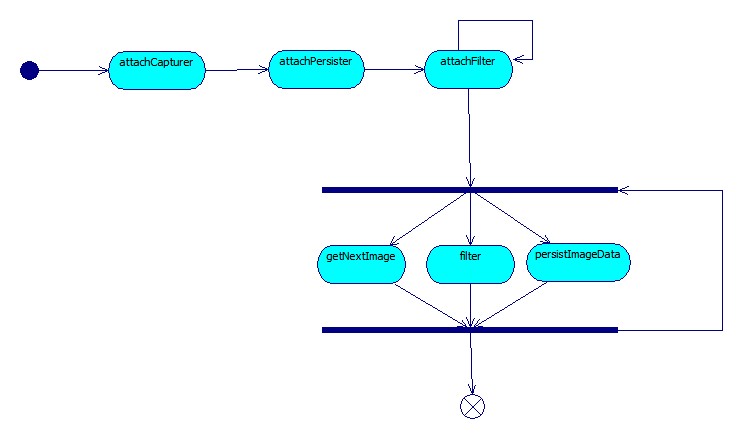
### Functional Use Case



### Sequential Pipeline Process Specification

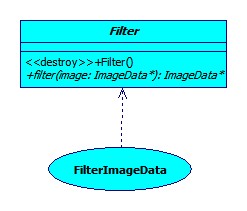


### Parallel Pipeline Process Specification

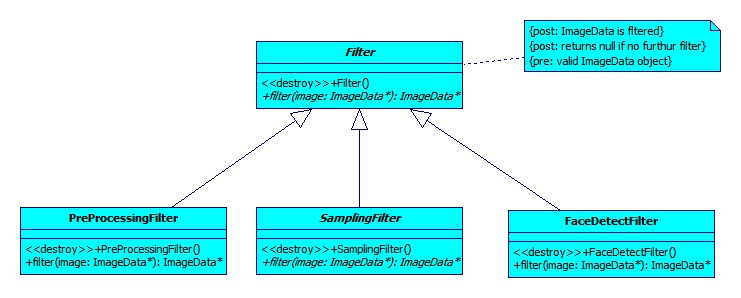


## Filtering an Image

### Functional Use Case



### Service Contract

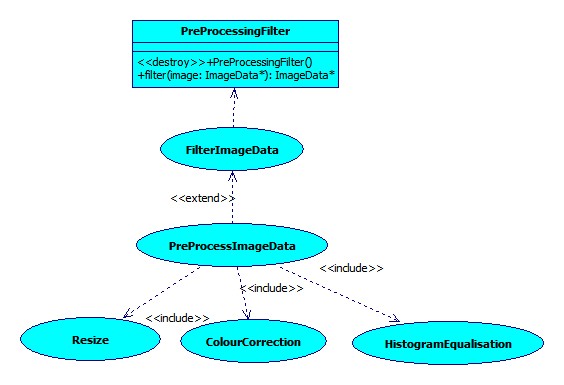


## PreProcessing an Image

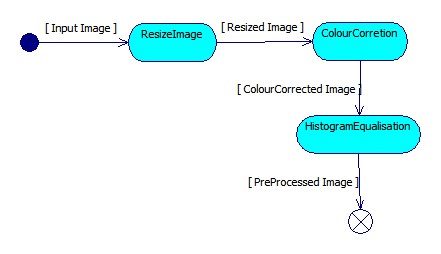
An image has to be pre-processed to ensure better facial recognition quality. Image pre-processing includes:

* Resizing the images to the same size. Images have to be the same dimensions in order to use them for facial recognition.
* Correcting the colour of the image. Images must be made greyscale as colour images are more susceptible to lighting conditions.
* Histogram equalisation of the image. This includes standardizing the brightness and contrast of the image.

### Functional Use Case



### Image PreProcessing Process Specification



## FaceDetectFilter

The FaceDetectFilter will be applied to a specific frame.Image FaceDetectFilter includes:

* Identify all faces within a frame.
* Crop facial images and add them to face vector for further processing. FaceDetectFilter attachments

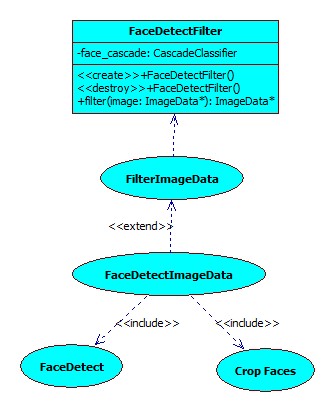
## FaceDetectFilter attachments

The filter must accept a valid location for the HaarCascading xml file.

## FaceDetectFilter processing

Thisfilter must capture faces from an image, push the facial images through the attached filters.

### Functional Use Case



### FaceDetect Filter Process Specification

### E:\Users\Mistry\Documents\GitHub\Andrologists2014\Documents\UML Pictures\FaceDetectActivity.jpgProcess

## Sampling Filter

The Sampling Filter will be applied to a specific frame. It includes two subclasses:

* NSamplingFilter: Accepts an image in a frame and returns every nth image, returns null otherwise
* DiffSamplingFilter: Accepts an image in a frame and compares it to the previous frame via histogram comparison. If the frames are similar, it returns the image. It returns null if the images in the frames are more than a certain percentage different.

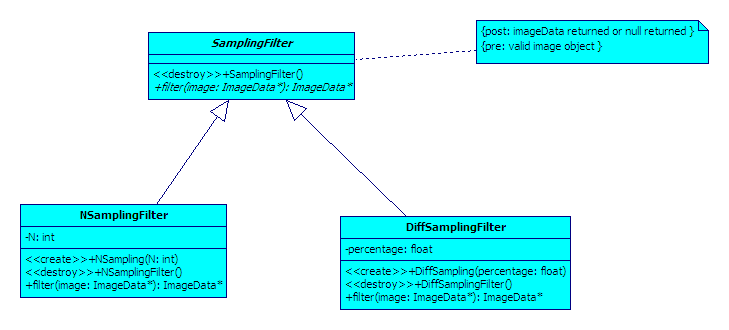
## Sampling Filter Attachments

This filter must accept valid images.

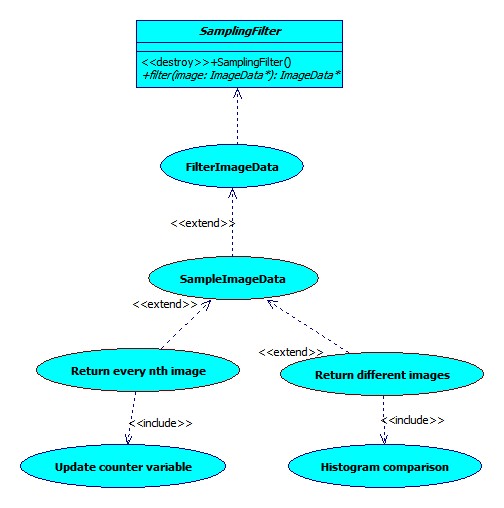
## Sampling Filter Processing

This filter must reduce the number of images that are returned. It must do comparisons in order to determine which images are returned.

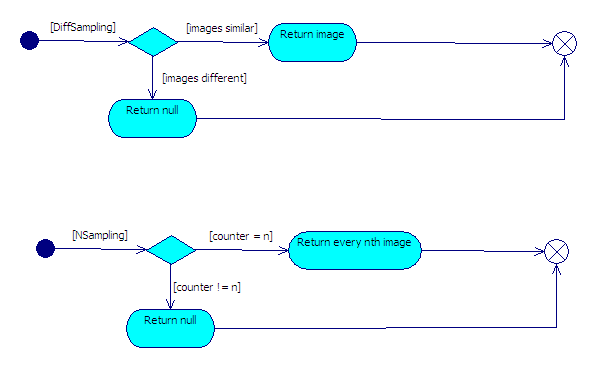
### Service Contract



### Functional Use Case



### Sampling Filter Process Specification



## Capturer

The Capturer will be used to get frames from a camera. It includes two subclasses:

* StreamCapturer: Connects to the camera directly using a connection string to specify which camera to connect to.
* DirectoryCapturer: Gets images stored in a directory.

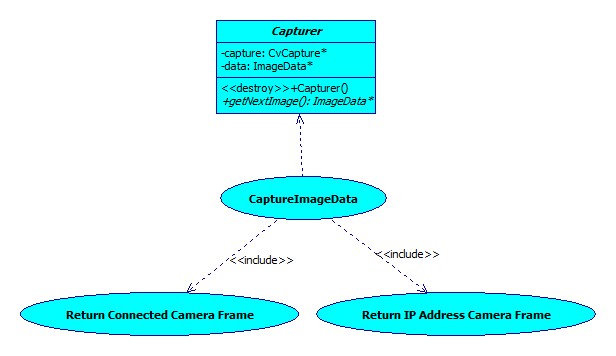
## Camera Attachments

Requires the IP Address of that camera, cameras connected to the computer do not require to be specified.

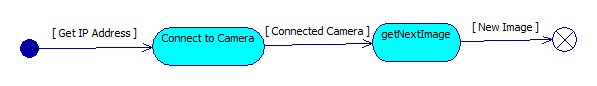
## Camera Processing

Once connected to the specified camera, a connection will be maintained and will return frames from the camera as being called upon.

### Functional Use Case



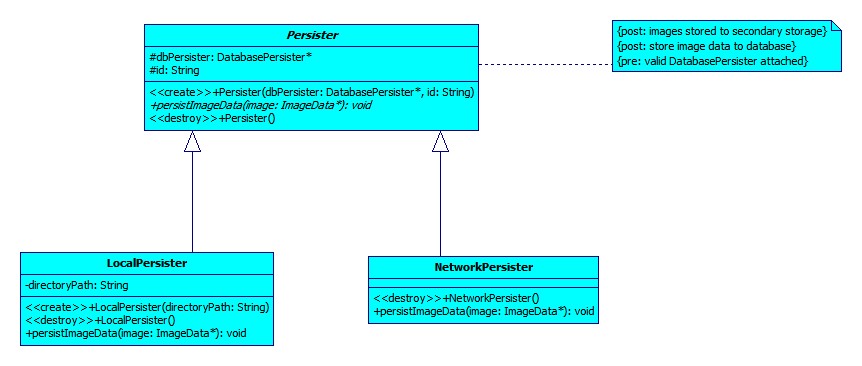
### Capturer Process Specification



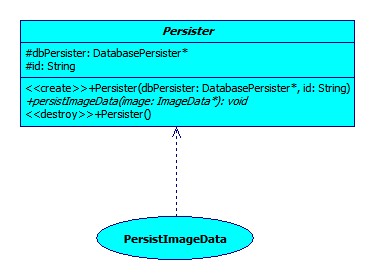
## Persister

A Persister is responsible for storing the image and its relative data to secondary storage.

### Service Contract



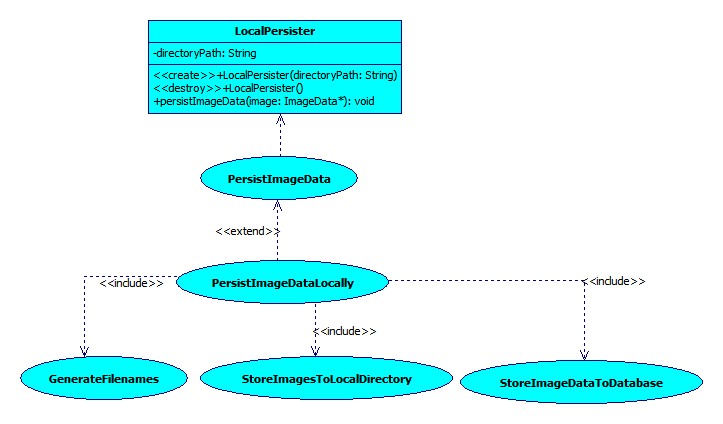
### Functional Use Case



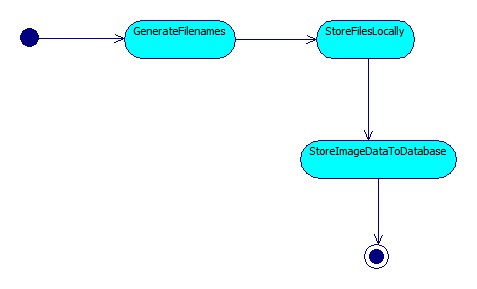
## LocalPersister

The LocalPersister is responsible for persisting the images locally (on the same machine) and delegating the responsibility of persisting the image data to a DatabasePersister.

### Functional Use Case



### LocalPersister Process Specification



## Database Persister

The Database Persister will be used to store the image filename and timestamp in the database. It has one subclasses:

* PostgreDatabasePersister: Implements the function to store the variables in the database.

It uses the DatabasePersistRequest class to gain access to the variables (filename and timestamp) of each image.

## Database Persister Attachments

This persister must accept valid image pointers.

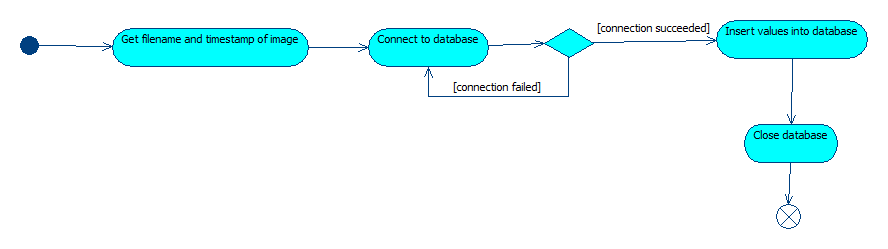
## Database Persister Processing

This persister must store each image’s values in the database.

## Functional Use Case

## 

## Database Persister Process Specification



# G:\University\2014\COS 301\Main Project\Documents\UML Pictures\DomainObjects.jpgDomain Objects