

Removista

Design and Implementation of Obstruction Removal Procedure

Final Year Project

Session 2015-2016

A 4th Year Student

A project submitted in partial fulfilment of the

Degree

of

BS in Computer Science

Department of Computer Science

SZABIST Karachi

13 May 2016

**Project Detail**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Type (Nature of project) | | | [ ] **Software** Based [ ] **Hardware** Based | | |
| Area of specialization | | |  | | |
| **Project Group Members** | | | | | |
| Sr.# | Reg. # | Student Name | | Email ID | \*Signature |
| (i) | Group Leader |  | |  |  |
| (ii) |  |  | |  |  |
| (iii) |  |  | |  |  |

\*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others

**Plagiarism Free Certificate**

This is to certify that, I am \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ S/D/o \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, group leader of FYP under registration no \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ at Computer Science Department, SZABIST. Karachi. I declare that my FYP proposal is checked by my supervisor. Code is attached herewith as Appendix A.

Date: \_\_\_\_\_\_\_\_\_\_\_\_ Name of Group Leader: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Supervisor: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Co-Supervisor (if any):\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Designation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Designation: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HoD: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgement**

We would like to grasp this opportunity to express our thankfulness to ALLAH. Indeed, there is no such thing that we can do without the help of the most merciful.

This project would not have been possible without the support and leadership of our project advisor Dr. Faraz Junejo, whose support and guidance lead us to cross every hurdle in our path to complete this project in the given deadline. We place on record, our sincere gratitude to Dr. Imran Amin (Head of Department BS Computing), Sir Asim Riaz (Program Coordinator BS Computing), Dr. Husnain (Program Manager BS Computing) and Madam Faria Jameel (FYP supervisor) for the continuous encouragement and prompt support throughout the course of this project.

We also would like to express our gratitude to the faculty of Computer Science Department SZABIST who provided us the healthy and helpful environment in developing this project as the group work involved in the project was mostly completed in the computer labs at campus.

These four years have been remarkable when it comes to cooperation from our colleagues. We would like to thank our class members and colloquies who supported and helped us whenever it was required and maintain a healthy competitive environment which proved to be a catalyst in developing this project.

We would also like to thank our mentors who express and showed great support regarding any queries related to the project and provide us way out whenever there was a breakout. The mentors did not hurry upon us at any stage and gave enough room to work independently on various tasks

Table of Contents

[1 Introduction 7](#_Toc450882034)

[1.1 Objective 7](#_Toc450882035)

[1.2 Problem Statement 7](#_Toc450882036)

[1.3 Assumption and Constraint 7](#_Toc450882037)

[1.4 Project Scope 7](#_Toc450882038)

[1.4.1 Definition of Obstruction 8](#_Toc450882039)

[1.4.2 Usage of OpenCV APIs 8](#_Toc450882040)

[1.4.3 Camera Calibration 8](#_Toc450882041)

[2 Requirements Analysis 9](#_Toc450882042)

[2.1 Existing system study/ State of Art 9](#_Toc450882043)

[2.2 Stakeholders list (Actors) 9](#_Toc450882044)

[2.3 Requirements elicitation 9](#_Toc450882045)

[2.3.1 Functional requirements 9](#_Toc450882046)

[2.3.2 Non-functional requirement 10](#_Toc450882047)

[2.4 Use case descriptions 10](#_Toc450882048)

[2.5 Use case design 12](#_Toc450882049)

[2.6 Software development life cycle model 13](#_Toc450882050)

[3 System Design & Implementation 14](#_Toc450882051)

[3.1 Context Diagram (Level 0) 14](#_Toc450882052)

[3.2 Activity diagram 15](#_Toc450882053)

[3.3 Software Architecture 16](#_Toc450882054)

[3.3.1 Input 17](#_Toc450882055)

[3.3.2 Pre-Processing 17](#_Toc450882056)

[3.3.3 Processing 17](#_Toc450882057)

[3.3.4 Output 18](#_Toc450882058)

[3.4 Sequence Diagram 19](#_Toc450882059)

[3.5 Class diagram 20](#_Toc450882060)

[4 System Testing 22](#_Toc450882061)

[4.1 Test cases 22](#_Toc450882062)

[5 Conclusion 23](#_Toc450882063)

[5.1 Problems faced and lessons learned 23](#_Toc450882064)

[5.2 Project summary 23](#_Toc450882065)

[5.3 Future work 23](#_Toc450882066)

[6 References 24](#_Toc450882067)

[7 Appendix 25](#_Toc450882068)

[7.1 Android Implementation 25](#_Toc450882069)

[7.1.1 MainActivity.java 25](#_Toc450882070)

[7.1.2 CameraViewActivity.java 29](#_Toc450882071)

[7.1.3 SplashActivity.java 31](#_Toc450882072)

[7.1.4 WalkthroughActivvity.java 32](#_Toc450882073)

[7.1.5 Activity\_main.xml 34](#_Toc450882074)

[7.1.6 Content\_main.xml 35](#_Toc450882075)

[7.1.7 Splash.xml 36](#_Toc450882076)

[7.1.8 Fade.xml (animation) 37](#_Toc450882077)

[7.1.9 AndroidManifest.xml 38](#_Toc450882078)

[7.1.10 Build.gradle 40](#_Toc450882079)

[7.2 C++ Implementation 41](#_Toc450882080)

[7.2.1 Source.cpp 41](#_Toc450882081)

**List of Figures**

1. Figure I Use case description 9
2. Figure II Use case diagram 12
3. Figure III Context diagram 13
4. Figure IV Activity diagram 14
5. Figure V Software Architecture 15
6. Figure VI Sequence Diagram 19
7. Figure VII Class diagram 20

# Introduction

Capturing photographs when surrounding conditions are not ideal can be difficult enough at times. Say if you are in middle of a crowded scene and crowd is blocking partially your view and affecting the actual composition of the shot, this is probably the last thing that you would want if you were a tourist. Overcoming these limitation poses a huge challenge.

This challenge falls under the domain of *Digital Image Processing(DIP)*, which is a technique to enhance raw images received from cameras/sensors. Image processing is any form of signal processing for which the input is an image, and the output is an enhanced image that is processed as per requirements.

We have successfully designed an obstruction removal procedure which proposes a smart solution for above mentioned limitation by employing digital image processing (DIP) and implemented it onto an android application called “Removista” [1].

Note: This application has been made available on Google Playstore. For download link kindly refer to Chapter 6.

## Objective

To design and implement an occlusion removal procedure that would;

* remove any undesired occlusions (including objects and people) from a given image by smartly identifying object of interest and occluded region.
* be implementable on smartphones.

## Problem Statement

Smartphone photography has constantly evolved since the very inception and realization of integrated mobile-cameras. Throughout this evolution digital photography industry has addressed number of photography challenges however there is a still a lot of room for improvement.

Instant photography in a crowded scene without getting a tainted photograph is an example of such place of improvement which needs to be addressed.

## Assumption and Constraint

Removista operates on a basic assumption that the camera will be held still by user. The performance of this application is subject to several factors which includes;

* the degree to which camera is held still
* luminosity conditions and scene contrast
* hardware processing capabilities

## Project Scope

This project aims to address limited occlusion removal issues.

### Definition of Obstruction

Most important point to grasp is the very definition of obstruction. Our definition of obstruction/occlusion for the course of this project will be any object that is in state of motion. Rest of the scene will be considered as the desired object of interest. This definition will serve as the foundation of our designed obstruction removal procedure and determine its functionality.

### Usage of OpenCV APIs

At this point, we lack the technical as well as theoretical knowledge of few methodologies of image processing. Hence for that purpose, instead of implementing these methods on our own we have delegated few responsibilities cum tasks of image processing to an industry-standard open source library for computer vision and image processing known as “Open CV (Open Source Computer Vision)”.

### Camera Calibration

*Camera calibration* is process of estimation of parameters of a given imaging device’s lens. These parameters are later used to correct lens distortion and to determine the location of the camera in the scene.

Our designed procedure for obstruction removal does not cater camera calibration. It has been intentionally left as a possible area for future work.

# Requirements Analysis

Our study helped us identify several requirements, which were kept in consideration while designing the functionality of Removista.

## Existing system study/ State of Art

Till date the only existing solution for occlusion and obstructions removal is by manually editing those images using image editing tools e.g. Photoshop. Which again requires human efforts and man power. Hence, a better automated solution is required.

Following are initiatives that motivated us and made us choose this problem for our project.

1. **Groopic [3]**

Groopic is a local image processing and enhancement initiative by Groopic Eyedus Lab’s. Groopic is an application that automatically merges two incomplete photos into a complete group photo which has both the photographers, and everybody else.

1. **Scaldo Remove [4]**

A self-proclaimed image processing industry leader named Scaldo issued a press release back in 2012 stating that they have found the solution for removing obstructions from images. But their product never made it to market.

1. **Google and MIT CSAIL De-fencing Algorithm [5]**

In a recent development in digital image processing industry, originally carried out by Google and MIT CSAIL. Researchers claim that they have found an algorithm that would revolutionized smartphone photography. They published a research paper that proposed a method for de-fencing images.

## Stakeholders list (Actors)

Our designed obstruction removal procedure requires two actors;

1. User
2. Processing capable device (PCD)

In case of the android implementation “Removista”, that PCD will be the android device.

## Requirements elicitation

### Functional requirements

#### Successful object of Interest Detection

This application has to be intelligent enough to distinguish object of interest and occlusions. This decision is made on the basis of the state of motion of objects in a given scene.

#### Resource Consumption

Our design procedure is supposed to operate with minimal resources. However, on a more honest note our designed procedure works fine when given processing resources of a full fledge computer, with intel based processer and a 8GB RAM but fails to operate optimally on android based devices with a 1 or 2 GB RAM.

### Non-functional requirement

#### HCI and Usability

In case of Removista, optimal usability has been our foremost preference. we have designed the user interface of application keeping in view all the fundamental principles of HCI.

An example of such usability is the walkthrough of application that user gets when the application is launched.

## Use case descriptions

Please note that Removista specifically does one thing i.e. removal of obstructions which makes use case very simplistic.

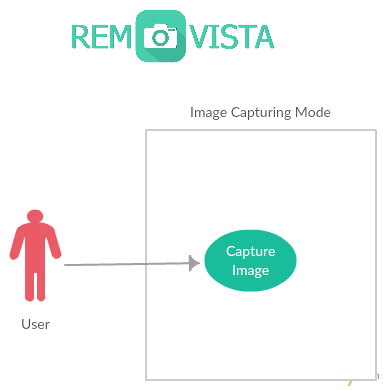
For the use case description kindly refer to next page.

|  |  |  |
| --- | --- | --- |
| Actor: | User | |
| Description: | In this use-case once the application is launched the user will tap on screen. | |
| Pre-Condition: | None. | |
| Trigger: | This use case triggers when user launches application | |
| Typical Course of Events: | **Actor Action** | **System Response** |
| **Step-1:** The user will capture image by tapping.  **Step-5:** The user will preview results. | **Step-2:** The system will capture stream of images while button is pressed  **Step-3:** The system will enter in algorithm execution mode and will process the stream of images.  **Step-4:** The system will return processed image |
| Alternate Course: | None | |
| Conclusion: | The use-case will conclude when the user will preview result | |

Figure I - Use Case Description

## Use case design

Following is the use case design of above mentioned use case scenario;

Figure II -Use Case Diagram

**Description**

Removista simply has an image capturing capability. Once the application is launched, a preview of processed stream of images is made visible to user. A live feed of camera preview, but a processed one where all moving objects are invisible, and static objects are visible.

## Software development life cycle model

We adopted Agile SDLC model throughout the course of FYP I and FYP II. Agile is much robust and is more persistence in terms of on-going development. Also, Agile provides some chance in reverse engineering as well as run time space to get it more useful for a project having a scope of FYP. This approach carries far less risk than Waterfall approaches. We focus on delivering fully-tested, independent, valuable, small features. As such, we diversify our risk – if one feature goes wrong, it should not impact another feature.

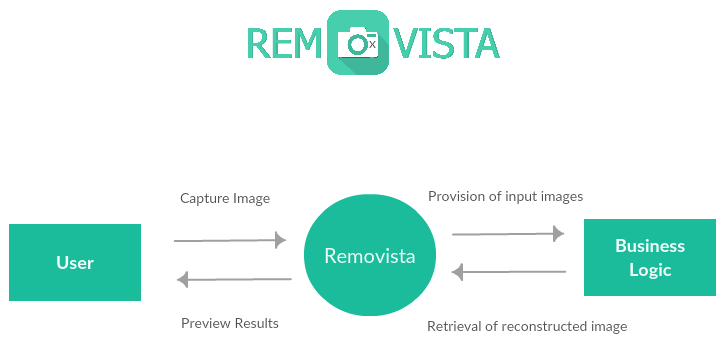
# System Design & Implementation

Following is the system design of Removista and a detailed overview of the employed procedure. Kindly note that few system diagrams were intentionally left that are usually part of System Design Document. Listed are those diagrams:

1. Database diagram (as there are no databases involved)
2. Network Diagram
3. Entity relationship diagram (there are no such entities in this project);

## Context Diagram (Level 0)

Figure III – Context Diagram



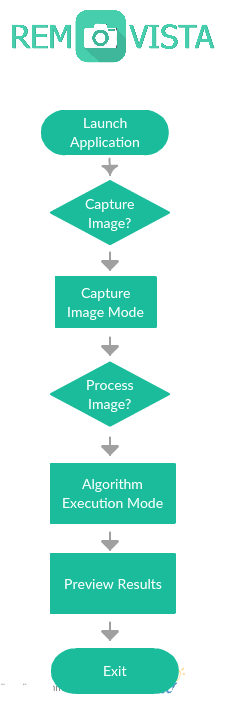
**Description**

Once application is launched, Removista enters into image capturing mode. A stream of captured frames is constantly loaded and sent to main business logic for further processing. Business logic is performed over received frames and processed stream of frames is returned.

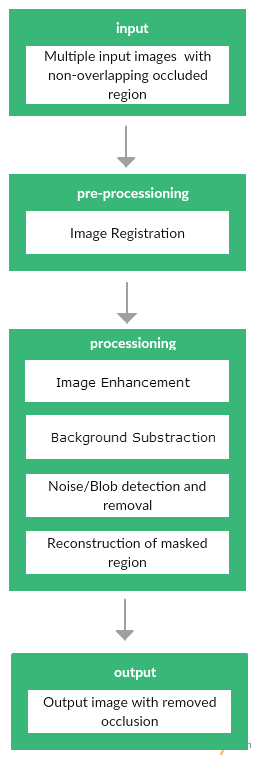
Given an intel based processer with 1 GB ram, Removista will operate on 10 fps (10 frames per second), i.e. a stream of 10 frames will be sent to business logic in every second. Better the computing power will be of device, more optimal and fast will be results. All these activities are performed on the main UI thread of android.

## Activity diagram

Figure IV – Activity Diagram



## Software Architecture

Figure V – Software Architecture

Intentionally excluded from this project. Possible area for future work.

**Description**

This procedure consists of four fundamental stages.

### Input

This algorithm will take images as input. Images where occluded region does not overlap.

Image is a two-dimensional function f(x,y) (here x and y are plane coordinates). The amplitude of image at any point say f is called intensity of image. It is also called the grey level of image at that point. We need to convert these x and y values to finite discrete values to form a digital image. We need to convert analog image to digital image to process it through digital computer. Each digital image composed of a finite elements called pixel.

### Pre-Processing

In pre-processing this algorithm shall realign images using image registration technique so that scene could be reconstructed. Image registration is a technique that re-align images that are taken from two different perspectives with minor displacement. This would accommodate and normalize varying conditions at the time of image capturing. Primarily it will address issues regarding camera displacement.

In our algorithm we would use "Intensity-Based Automatic Image Registration".

### Processing

Both foreground detection and foreground masking will be performed during the process of background subtraction.

**Image Enhancement**

Prior to any processing we will require stream of images with high contrast and moderate brightness. Gain (contrast) is adjusted by multiplication of frames and bias (brightness) is adjusted by addition of frames.

**Image Segmentation**

**Background Segmentation**

We will be using OpenCV’s BackgroundSubtractionMOG2 for image segmentation purposes. BackgroundSubtractorMOG2 is an implementation of Mixed Gaussian Model for background subtraction, that was presented in a research paper titled "Improved adaptive Gaussian mixture model for background subtraction" by Z.Zivkovic. In order to function this implementation requires parameters which are listed as follow;

1. **Learning rate**

The rate with which background model will adapt to stream of live feed. that we found to be working in almost all cases was 0.2

1. **Threshold**

The initial threshold value which makes a pixel either part of background model or foreground model. This is number is actually determined dynamically per pixel as background subtraction progresses.

1. **Threshold Detection**

Boolean value, decides either to make exceptions of shadows that are in scene.

1. **History Size**

The number of frames that will be used to calculate foreground model.

**Foreground Detection**

Processing will initiate by identification of background and foreground. Any object that is in state of motion will fall in domain of foreground. Rest of the region will be our background model. Note that foreground will be occlusion and background will be the region of interest.

For foreground detection we would use either "Frame Differencing" or Gaussian Mixture Model" algorithm.

**Foreground Masking**

Identified foreground will get a mask for further processing. This mask will represent the occluded region.

**c. Blob Detection and Removal**

Obtained mask will be further cleansed by carrying out noise/blob detection and removal technique. We’ll first run histogram equalization which will help us obtain uniformly distributed grey scale levels. Lastly, Salt and Pepper were removed using median blurring method.

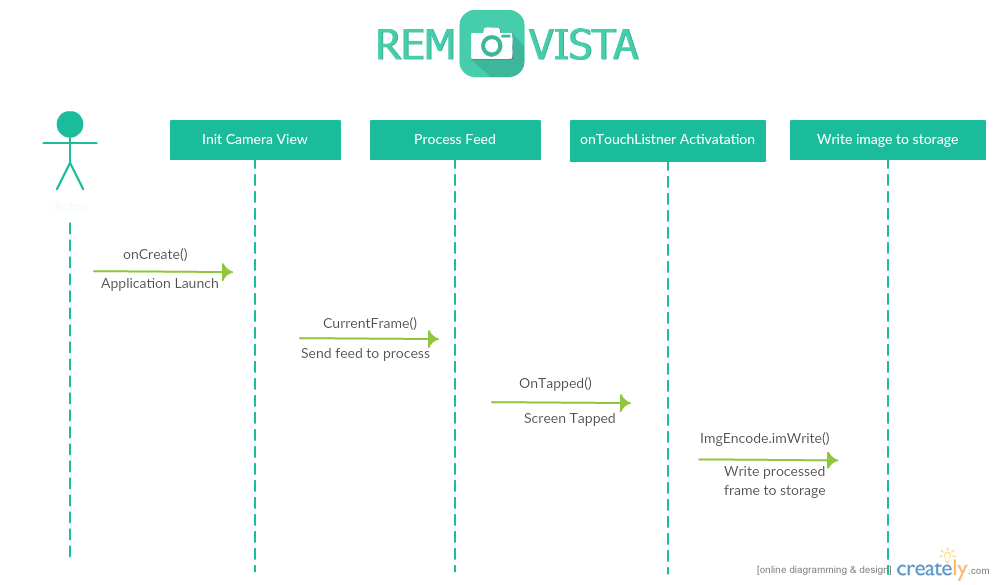
**d. Reconstruction of masked image**

For reconstruction of occluded region this algorithm will use spatial-temporal information. Masked region that was occluded in frame from time t will be replaced by pixels of frame from time t+x of same region. This is the very reason why we used image with non-overlapping occluded regions in input.

### Output

Final resulting output will be an image with no occluded region. The only thing that will be in the scene of resulting image will be object of interest.

## Sequence Diagram

Figure VI – Sequence Diagram

**Description**

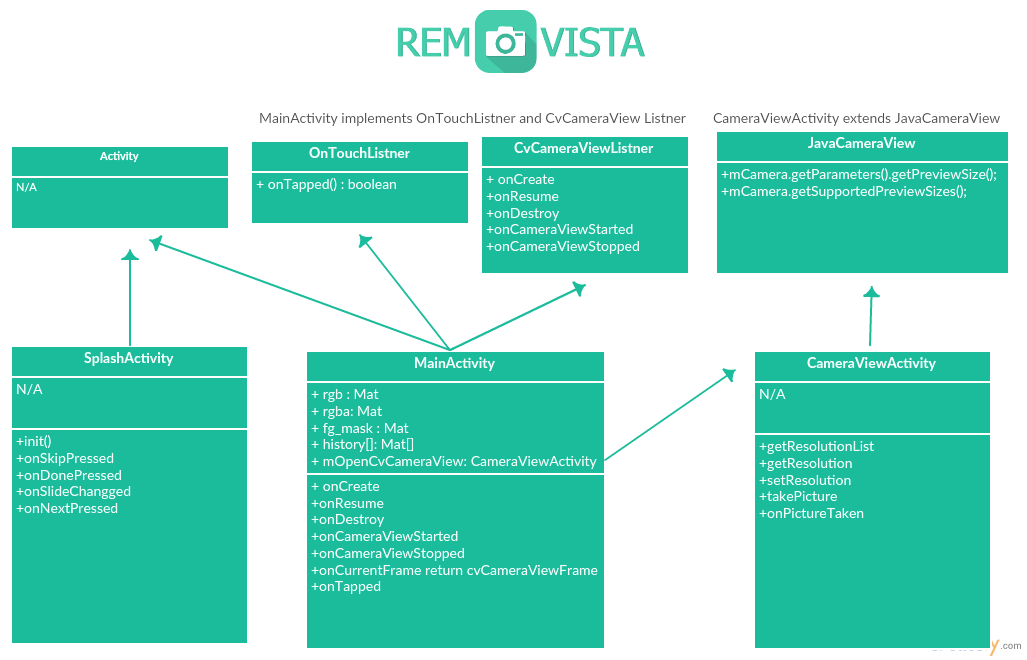
Removista is composed of a set of four sequence control participants; “Init Camera View”, “Process Feed”, “onTouchListner Activation” and “Write image to storage”.

With launch of application by actor, the MainActivity,java calls onCreate() trigger and displays Layout\_main.xml on screen. Layout\_main.xml contains a camera view control which is initiated as soon as Layout\_main.xml is displayed.

OnCreate() triggers several other methods with its call. Amongst them, most important of all is CurrentFrame() where all core image processing logic is implemented. Application retrieves feed from camera, and sends all raw frames to CurrentFrame() which in return sends back processed frames. The processed individual frames are attached to main layout, which is the preview.

OnCreate() also initiates an onTouchListner. An OnTapped() method is initiated when a user taps screen. And the current processed frame is then written to storage of device.

## Class diagram

Figure VII – Class Diagram

**Description**

MainAcitivty.java is the heart and soul of Removista where all business logic is implemented. It extends Activity.java class (which is mandatory for all android activity classes) and implements two abstract class; OnTouchListner and CvCameraViewListner.java (provided by OpenCV library). CvCameraViewListner.java is responsible for listening feed of camera and processes them. CameraViewActivity.java implements JavaCameraView.java which provides methods for dealing with camera operations. Considerable components are discussed as follow;

**MainActivity.java**

Frames of images are basically a 2 dimensional array of pixels having different grayscale level values ranging from 0-255. We use Mat object to represent the matrices of frames. Object rgb will hold 3 channel frame of red, green and blue while object rgba will hold four channel frame of red, green, blue and alpha. Object fg\_mask will contain 1 channel mask which is returned after the process of background subtraction. mOpenCvCameraView is used for previewing processed image which is represented by CameraViewActivity.class.

Since MainActivity.java has implemented OnTouchListner.java it will now have to make use of OnTapped() event trigger. Most importantly CvCameraViewListner implementation will require usage of onCurrentFrame().

Logic of when a camera view is initiated or suspended is written in OnCameraViewStarted() and OnCameraViewStopped(). The logic when application’s process/thread is initiated is defined OnCreate(), its pause logic and resume logic is defined in OnPause() and OnResume() accordingly. In these methods we enable or disable camera as per our requirement. OnTapped() is the method which is trigger when an on-screen tap is recorded.

**SplashAcivity.java**

The introductory walkthrough is defined is SplashAcitivty.java and walkthrough.java.

# System Testing

## Test cases

Removista was tested on iterative bases during development. Note that instead of quantitative metrics, we relied on naked eyes for results comparison. This project was tested in outdoor and indoor conditions, furthermore the results of each test cases were different. Test cases helped us realize that with good lighting, and proper camera handling the results can be made consistent, and the extra-terrestrial object was accurately removed from frames, on contrary, in case of poor camera handling and low lighting made the results blurry, and some parts of undesired object were left inside the image.

Our findings were as followed;

1. This procedure works well in a scene with optimal contrast and brightness. Better contrast makes interpretability of objects easier and which in return makes background Subtraction more effective.
2. This process works optimally on devices/systems with higher processing power.
3. Low resolution requires less processing power, tuning down camera to lower resolution enhances results.

# Conclusion

Our solution would prove to be significant for everyone who owns a Smartphone with a camera. Pakistan alone has 72% Smartphone using population. Tourists will find this development very helpful. Tourist and other point and shoot photographers would never want a photo-bombed/spoiled memory.

## Problems faced and lessons learned

During the course of this project we learned fundamentals of digital image processing, implemented different key stages that are involved in digital image processing problems and got an insight of several image processing techniques. On a technical stand point we found the phase of background subtraction extremely difficult, though we found a work-around with the help of our respected advisor. On a non-technical note we have improved our team working skills.

## Project summary

Removista is an implementation of our very own obstruction removal procedure cum algorithm, answering a long-due question of computer vision. With some assumptions and constraints, we have successfully found a solution for obstruction free photography which can be further improved. By employing a sliding-window logic we can remove occlusion without disturbing the comparison of scene and protect our object of interest.

## Future work

As identified earlier a possible area of future work can be to make this application constraint-free of stiffed position using image registration or calibration techniques. Moreover, there is allot room for improvement when it comes to efficiency and results.

# References

1. play.google.com/store/apps/details?id=unversity.shawaiz.projectopencv3test&hl=en
2. Zivkovic, Zoran. "Improved adaptive Gaussian mixture model for background subtraction." Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. Vol. 2. IEEE, 2004.
3. Groopic.com
4. scalado.com/display/en/Remove
5. sites.google.com/site/obstructionfreephotography
6. docs.opencv.org/3.1.0/d1/dc5/tutorial\_background\_subtraction.htmlwww.slideshare.net/ravi5raj\_88/background-subtraction
7. www.tutorialspoint.com/dip/
8. www.mathworks.com/discovery/digital-image-processing.html
9. eeweb.poly.edu/~onur/lectures/lectures.html
10. nptel.ac.in/courses/117105079/
11. www.cs.utexas.edu/~grauman/courses/fall2009/slides/lecture9\_background.pdf

# Appendix

## Android Implementation

### MainActivity.java

package szabist.fyp.removista;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.hardware.Camera;

import android.os.Bundle;

import android.os.Environment;

import android.util.Log;

import android.view.MotionEvent;

import android.view.SurfaceView;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.view.WindowManager;

import android.view.View.OnTouchListener;

import android.widget.Toast;

import org.opencv.android.BaseLoaderCallback;

import org.opencv.android.CameraBridgeViewBase;

import org.opencv.android.CameraBridgeViewBase.CvCameraViewListener2;

import org.opencv.android.LoaderCallbackInterface;

import org.opencv.android.OpenCVLoader;

import org.opencv.core.CvType;

import org.opencv.core.Mat;

import org.opencv.imgcodecs.Imgcodecs;

import org.opencv.imgproc.Imgproc;

import org.opencv.video.BackgroundSubtractor;

import org.opencv.video.Video;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

//public class MainActivity extends Activity implements CvCameraViewListener2{

public class MainActivity extends Activity implements CvCameraViewListener2, OnTouchListener{

private static final String TAG = "Removista::MainActivity";

static{

//helper method, check if opencv loaded

if(!OpenCVLoader.initDebug()){

Log.i("Removsita", "Initilzization failed");

}else{

Log.i("Removsita", "Initilzization succeded");

}

}

public MainActivity() {

Log.i("Removsita", "Instantiated new " + this.getClass());

}

//opencv dedicated camera

protected CameraViewActivity mOpenCvCameraView;

//protected CameraBridgeViewBase mOpenCvCameraView;

// variable initializatio

//frames

private Mat rgba; //mat with alpha

private Mat rgb; //mat without alpha

private Mat fg\_mask;

private Mat history[];

private int index = 0;

private List<Camera.Size> mResolutionList;

//backgground segmentation

private BackgroundSubtractor sub; //background subtractor

private final int history\_size = 50;

private final float thresold = 0;

private final boolean detect\_shadows = true;

//opencv manager call

private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {

@Override

public void onManagerConnected(int status) {

switch (status) {

case LoaderCallbackInterface.SUCCESS:{

Log.i("opencv", "OpenCV loaded successfully");

mOpenCvCameraView.enableView();

mOpenCvCameraView.setOnTouchListener(MainActivity.this);

} break;

default:{

super.onManagerConnected(status);

} break;

}

}

};

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// keep cmamera open

getWindow().addFlags(WindowManager.LayoutParams.FLAG\_KEEP\_SCREEN\_ON);

//bin content layout to following class

setContentView(R.layout.activity\_main);

// get hangle of camera

mOpenCvCameraView = (CameraViewActivity) findViewById(R.id.camera\_canvas);

//mOpenCvCameraView = (CameraBridgeViewBase) findViewById(R.id.camera\_canvas);

//set visibility

mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);

mOpenCvCameraView.setCvCameraViewListener(this);

// enable camera

mOpenCvCameraView.enableView();

mOpenCvCameraView.setOnTouchListener(MainActivity.this);

Toast.makeText(this, "Tap to capture!", Toast.LENGTH\_LONG).show();

}

@Override

public void onResume()

{

super.onResume();

//enable camera view thread application is resumed

mOpenCvCameraView.enableView();

}

@Override

public void onPause()

{

super.onPause();

//disable camera when application thread is paused

if (mOpenCvCameraView != null)

mOpenCvCameraView.disableView();

}

public void onDestroy() {

super.onDestroy();

if (mOpenCvCameraView != null)

mOpenCvCameraView.disableView();

}

public void onCameraViewStarted(int width, int height) {

sub= Video.createBackgroundSubtractorMOG2(history\_size, thresold, detect\_shadows);

rgba = new Mat(height, width, CvType.CV\_8UC4);

rgb = new Mat(height, width, CvType.CV\_8UC4);

fg\_mask = new Mat(height, width, CvType.CV\_8UC1);

history= new Mat[1000];

}

public void onCameraViewStopped() {

rgba.release();

rgb.release();

fg\_mask.release();

}

public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {

//current frame's rgba

rgba = inputFrame.rgba();

history[index] = new Mat(3, 3, CvType.CV\_64FC1 );

// convert rgba to rgb

Imgproc.cvtColor(rgba, rgb, Imgproc.COLOR\_RGBA2RGB);

rgba.copyTo(history[index]);

if(index==999){

index=10;

}

if(index > 10) {

// apply background subtraction

sub.apply(rgb, fg\_mask);

Imgproc.equalizeHist(fg\_mask, fg\_mask);

Imgproc.medianBlur(fg\_mask, fg\_mask, 5);

history[index-10].copyTo(rgba, fg\_mask);

return rgba;

}

index++;

return null;

}

@SuppressLint("SimpleDateFormat")

@Override

public boolean onTouch(View v, MotionEvent event) {

Toast.makeText(this, "Tapped!", Toast.LENGTH\_SHORT).show();

Log.i(TAG, "onTouch event");

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd\_HH-mm-ss");

String currentDateandTime = sdf.format(new Date());

String fileName = Environment.getExternalStorageDirectory().getPath() +

"/sample\_picture\_" + currentDateandTime + ".jpg";

Imgcodecs.imwrite(fileName, rgba);

//mOpenCvCameraView.takePicture(fileName);

Toast.makeText(this, fileName + " saved", Toast.LENGTH\_SHORT).show();

return false;

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.menu\_main, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

int id = item.getItemId();

//noinspection SimplifiableIfStatement

if (id == R.id.action\_settings) {

return true;

}

return super.onOptionsItemSelected(item);

}

}

### CameraViewActivity.java

package szabist.fyp.removista;

import java.io.FileOutputStream;

import java.util.List;

import org.opencv.android.JavaCameraView;

import android.content.Context;

import android.hardware.Camera;

import android.hardware.Camera.PictureCallback;

import android.hardware.Camera.Size;

import android.util.AttributeSet;

import android.util.Log;

public class CameraViewActivity extends JavaCameraView implements PictureCallback {

private static final String TAG = "Removista::debugging";

private String mPictureFileName;

public CameraViewActivity(Context context, AttributeSet attrs) {

super(context, attrs);

}

public List<Size> getResolutionList() {

Camera.Parameters parameters = mCamera.getParameters();

return parameters.getSupportedPreviewSizes();

}

public void setResolution(Size resolution) {

disconnectCamera();

mMaxHeight = resolution.height;

mMaxWidth = resolution.width;

connectCamera(getWidth(), getHeight());

}

public Size getResolution() {

return mCamera.getParameters().getPreviewSize();

}

public void takePicture(final String fileName) {

Log.i(TAG, "Taking picture");

this.mPictureFileName = fileName;

// Postview and jpeg are sent in the same buffers if the queue is not empty when performing a capture.

// Clear up buffers to avoid mCamera.takePicture to be stuck because of a memory issue

mCamera.setPreviewCallback(null);

// PictureCallback is implemented by the current class

mCamera.takePicture(null, null, this);

}

@Override

public void onPictureTaken(byte[] data, Camera camera) {

Log.i(TAG, "Saving a bitmap to file");

// The camera preview was automatically stopped. Start it again.

mCamera.startPreview();

mCamera.setPreviewCallback(this);

// Write the image in a file (in jpeg format)

try {

FileOutputStream fos = new FileOutputStream(mPictureFileName);

fos.write(data);

fos.close();

} catch (java.io.IOException e) {

Log.e("PictureDemo", "Exception in photoCallback", e);

}

}

}

### SplashActivity.java

package szabist.fyp.removista;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.animation.Animation;

import android.view.animation.AnimationUtils;

import android.widget.ImageView;

/\*\*

\* Created by user212 on 5/10/2016.

\*/

public class SplashActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.splash);

final ImageView iv = (ImageView) findViewById(R.id.imageView);

final Animation an = AnimationUtils.loadAnimation(getBaseContext(), R.anim.fade);

final Animation an2 = AnimationUtils.loadAnimation(getBaseContext(), R.anim.fade\_0);

iv.startAnimation(an);

an.setAnimationListener(new Animation.AnimationListener(){

@Override

public void onAnimationStart(Animation animation) {

}

@Override

public void onAnimationEnd(Animation animation) {

iv.startAnimation(an2);

finish();

Intent i = new Intent(getBaseContext(),WalkthroughActivity.class);

startActivity(i);

}

@Override

public void onAnimationRepeat(Animation animation) {

}

});

}

}

### WalkthroughActivvity.java

package szabist.fyp.removista;

import android.content.Intent;

import android.graphics.Color;

import android.os.Bundle;

import com.github.paolorotolo.appintro.AppIntro;

import com.github.paolorotolo.appintro.AppIntroFragment;

public class WalkthroughActivity extends AppIntro {

// Please DO NOT override onCreate. Use init.

@Override

public void init(Bundle savedInstanceState) {

// Add your slide's fragments here.

// AppIntro will automatically generate the dots indicator and buttons.

// addSlide(first\_fragment);

// Instead of fragments, you can also use our default slide

// Just set a title, description, background and image. AppIntro will do the rest.

addSlide(AppIntroFragment.newInstance("REMOVISTA", "The ultimate obstruction removal tool for crowded scene photography.", R.drawable.intro, Color.parseColor("#2196F3"), Color.parseColor("#ffffff"),Color.parseColor("#ffffff")));

addSlide(AppIntroFragment.newInstance("AIM", "Aim your target, make sure that camera doesn't move", R.drawable.aim, Color.parseColor("#E91E63"), Color.parseColor("#ffffff"),Color.parseColor("#ffffff")));

addSlide(AppIntroFragment.newInstance("WAIT", "Wait for the right moment", R.drawable.wait, Color.parseColor("#009688"), Color.parseColor("#ffffff"),Color.parseColor("#ffffff")));

addSlide(AppIntroFragment.newInstance("TAP", "and Capture your moment by tapping on screen", R.drawable.tap, Color.parseColor("#B71C1C"), Color.parseColor("#ffffff"),Color.parseColor("#ffffff")));

addSlide(AppIntroFragment.newInstance("HAVE FUN!", "Find captured image in files directory of your android device.", R.drawable.location, Color.parseColor("#880E4F"), Color.parseColor("#ffffff"),Color.parseColor("#ffffff")));

// OPTIONAL METHODS

// Override bar/separator color.

setBarColor(Color.parseColor("#000000"));

setSeparatorColor(Color.parseColor("#2196F3"));

// Hide Skip/Done button.

showSkipButton(true);

setProgressButtonEnabled(true);

setFadeAnimation();

// Turn vibration on and set intensity.

// NOTE: you will probably need to ask VIBRATE permisssion in Manifest.

}

@Override

public void onSkipPressed() {

// Do something when users tap on Skip button.

Intent i = new Intent(getBaseContext(),MainActivity.class);

startActivity(i);

}

@Override

public void onDonePressed() {

// Do something when users tap on Done button.

Intent i = new Intent(getBaseContext(),MainActivity.class);

startActivity(i);

}

@Override

public void onSlideChanged() {

// Do something when the slide changes.

}

@Override

public void onNextPressed() {

// Do something when users tap on Next button.

}

}

### Activity\_main.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal"

>

<szabist.fyp.removista.CameraViewActivity

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:visibility="gone"

android:id="@+id/camera\_canvas"

/>

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge"

android:text="Large Text"

android:id="@+id/lblTest"

android:textAlignment="center"

android:layout\_gravity="center\_vertical|center\_horizontal"

android:hint="test" />

</LinearLayout>

### Content\_main.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"

tools:context="szabist.fyp.removista.MainActivity"

tools:showIn="@layout/activity\_main">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Hello World!" />

</RelativeLayout>

### Splash.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:paddingBottom="@dimen/activity\_vertical\_margin"

tools:context=".splash"

android:background="#e6e6e6"

android:alpha="1"

>

<ImageView

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:layout\_gravity="center\_horizontal|center\_vertical"

android:layout\_centerHorizontal="true"

android:id="@+id/imageView"

android:src="@mipmap/logo"

android:alpha="1"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:background="#ffffff"

/>

</LinearLayout>

### Fade.xml (animation)

<?xml version="1.0" encoding="utf-8"?>

<set xmlns:android="http://schemas.android.com/apk/res/android">

<alpha xmlns:android="http://schemas.android.com/apk/res/android"

android:interpolator="@android:anim/accelerate\_interpolator"

android:fromAlpha="0.0" android:toAlpha="1.0"

android:duration="3000" />

</set>

### AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="szabist.fyp.removista"

android:versionCode="301"

android:versionName="3.01">

<uses-sdk android:minSdkVersion="8" />

<supports-screens

android:anyDensity="true"

android:largeScreens="true"

android:normalScreens="true"

android:resizeable="true"

android:smallScreens="true" />

<uses-permission android:name="android.permission.CAMERA" />

<uses-feature

android:name="android.hardware.camera"

android:required="false" />

<uses-feature

android:name="android.hardware.camera.autofocus"

android:required="false" />

<uses-feature

android:name="android.hardware.camera.front"

android:required="false" />

<uses-feature

android:name="android.hardware.camera.front.autofocus"

android:required="false" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />

<uses-permission android:name="android.permission.VIBRATE" />

<application

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/Theme.AppCompat.Light.NoActionBar"

android:largeHeap="true"

>

<activity

android:name="szabist.fyp.removista.MainActivity"

android:configChanges="keyboardHidden|orientation"

android:label="@string/app\_name"

android:screenOrientation="landscape"

android:theme="@android:style/Theme.Light.NoTitleBar">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</activity>

<activity

android:name="szabist.fyp.removista.SplashActivity"

android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity android:name="szabist.fyp.removista.WalkthroughActivity"

android:label="Walkthrough"

android:theme="@style/Theme.AppCompat.Light.NoActionBar"/>

</application>

</manifest>

### Build.gradle

apply plugin: 'com.android.application'

android {

compileSdkVersion 23

buildToolsVersion "23.0.2"

defaultConfig {

applicationId "unversity.shawaiz.projectopencv3test"

minSdkVersion 17

targetSdkVersion 23

versionCode 1

versionName "1.0"

}

buildTypes {

debug {

debuggable true

}

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

}

}

}

repositories {

mavenCentral()

}

dependencies {

compile fileTree(dir: 'libs', include: ['\*.jar'])

testCompile 'junit:junit:4.12'

compile 'com.android.support:appcompat-v7:23.1.1'

compile 'com.android.support:design:23.1.1'

compile project(':libs:opencv')

compile 'com.github.paolorotolo:appintro:3.4.0'

compile 'com.android.support:appcompat-v7:23.1.1'

}

## C++ Implementation

### Source.cpp

**//**video writing specific

#include "opencv\highgui.h"

#include "opencv\cv.h"

//image segmentation (bg subtraction) specific

#include "opencv2\opencv.hpp"

#include "opencv2\core\core.hpp"

#include "opencv2\highgui\highgui.hpp"

#include "opencv2\video\background\_segm.hpp"

//c++ specific

#include "iostream"

#include "stdio.h"

using namespace cv;

using namespace std;

int main(int argc, char\* argv[])

{

//-------------------------------------------------------------------image enhacment tests

//Mat orignal, contrast, histogram, grey;

//orignal = imread("C:\\Users\\user212\\Documents\\Visual Studio 2015\\Projects\\projectTestOpenCV\\projectTestOpenCV\\test\\test\_image\_enhacement\_opreationsimg.jpg", CV\_LOAD\_IMAGE\_COLOR);

//if (orignal.empty())

//{

// std::cout << "!!! Failed imread(): image not found" << std::endl;

// // don't let the execution continue, else imshow() will crash.

//}

//

////contrast adjustment

// orignal.convertTo(contrast,-1,2.2,50);

// if (contrast.empty())

// {

// std::cout << "!!! Failed to convertto(): image not found" << std::endl;

// // don't let the execution continue, else imshow() will crash.

// }

////histogram adjustment

// /// Convert to grayscale

// cvtColor(orignal, grey, CV\_BGR2GRAY);

// /// Apply Histogram Equalization

// equalizeHist(grey, histogram);

//namedWindow("Original Image", 0);

//namedWindow("Contrast adjustments", 0);

//namedWindow("Histogram adjustments", 0);

//while (1) {

//

// imshow("Original Image", orignal);

// imshow("Contrast adjustments", contrast);

// imshow("Histogram adjustments", histogram);

// switch (waitKey(10))

// {

// case 32:

// //capture image at space bar

// //imwrite("test\\/orignal1.jpg", orignal);

// //imwrite("test\\/contrast1.jpg", contrast);

// imwrite("test\\/histogram1.jpg", histogram);

// break;

// case 27:

// //exit at escape

// return 0;

// }

//}

//------------------------------------------------------------------- image segmentation + init camera

//

// //global variables

// Mat frame /\*camera feed\*/, res /\*processed image\*/, mask /\*mask\*/;

//

// Ptr<BackgroundSubtractor> pMOG; //MOG Background subtractor

// Ptr<BackgroundSubtractor> pMOG2; //MOG2 Background subtractor

// Ptr<BackgroundSubtractorGMG> pGMG; //MOG2 Background subtractor

//

// //pMOG = new BackgroundSubtractorMOG(100,5,0,1);

// pMOG2 = new BackgroundSubtractorMOG2(1000,0,1);

// //pGMG = new BackgroundSubtractorGMG();

//

// //for reading saved file

// //char fileName[100] = "C:\\opencv\\/myVideo.avi"; //Gate1\_175\_p1.avi"; //mm2.avi"; //";//\_p1.avi";

// //VideoCapture stream1(fileName);

//

// //camera initializer

// VideoCapture cap;

// cap.open(0);

//

// //unconditional loop

// while (true) {

//

// cap >> frame;

// //contrast adjustment+brightness

// //frame.convertTo(frame, -1, 0.5,0);

//

//

//

// res = frame;

//

// /////////////////////////////////////////using MOG -> constructor(set\_input\_frame, get\_output\_mask\_here, learning\_rate)

// //pMOG->operator()(frame, mask, 0.8);

// ////enhancment of mask

// //Mat histogram, grey;

// ///// Convert to grayscale

// ////cvtColor(frame, grey, CV\_BGR2GRAY);

//

// ///// Apply Histogram Equalization

// //equalizeHist(mask, histogram);

//

// ////test for channels in image matrix

// ////std::cout << mask.channels() << endl;

// ////std::cout << histogram.channels() << endl;

//

// /////////////////////////////////////////using MOG2 -> constructor(set\_input\_frame, get\_output\_mask\_here, learning\_rate)

// pMOG2->operator()(frame,mask,0.6);

//

// //enhancment of mask

// Mat mask\_histogram,grey,mask\_salt,res\_histogram, res\_salt;

// //Convert to grayscale

// //cvtColor(frame, grey, CV\_BGR2GRAY);

//

// //Apply Histogram Equalization

// equalizeHist(mask, mask\_histogram);

//

// //salt pepper removal using median blurring

// medianBlur(mask, mask\_salt, 5);

//

// //test for channels in image matrix

// //std::cout << mask.channels() << endl;

// //std::cout << histogram.channels() << endl;

//

// /////////////////////////////////////////using GMG/////////////////////////////////////////

// //pGMG->operator()(frame, mask);

//

// //imshow("frame", frame);

// //imshow("mask", mask);

// //imshow("histogram enhanced mask", mask\_histogram);

//

// imshow("salt", mask\_salt);

//

// //????????????????? here frame from previous 10th last frame needs to be copied using max to current frame??????????????????

// frame.copyTo(res, mask);

//

// frame.copyTo(res\_histogram, mask\_histogram);

// frame.copyTo(res\_salt, mask\_salt);

//

// /\*imshow("No image enhancment", res);

// imshow("Histogram Equalized", res\_histogram);

// imshow("Salt fixed", res\_salt);

//\*/

// //listen for 10ms for a key to be pressed

// switch (waitKey(10))

// {

// case 32:

//

// //capture image at space bar

// imwrite("captured\_image\\/unprocessed.jpg", frame);

// imwrite("captured\_image\\/mask with histogram enhanced.jpg", mask\_histogram);

// imwrite("captured\_image\\/mask with salt removed.jpg", mask\_salt);

// imwrite("captured\_image\\/processed.jpg", res);

//

// imshow("captured", res);

// imshow("captured", mask\_salt);

//

// break;

//

// case 27:

// //exit at esc

// return 0;

// }

//

// }

//

/////////////////////////////////////////////////////////////////background model calculation

//global variables

Mat frame /\* camera feed \*/,

processed\_image /\* processed image \*/,

initial\_mask /\* initial mask returned by MOG backgroud subtractor \*/,

histogram\_mask /\* mask after histogram equalziation \*/,

noise\_free\_mask /\* mask after noise removal of salt and peper \*/,

history[100],

res

;

int counter\_history = 0; /\*sliding window history counter\*/

//MOG Background subtractor initialziation

//constructor paramters : history size, number of gaussain mixtures, background ratio, noise strength)

/\*Ptr<BackgroundSubtractor> pMOG;

pMOG = new BackgroundSubtractorMOG(100,5,0,1);\*/

//MOG2 Background subtractor initialziation

//constructor paramters : history size, threshold value, shadow detection

Ptr<BackgroundSubtractor> pMOG2;

pMOG2 = new BackgroundSubtractorMOG2(300,0, 0);

//camera initializer

VideoCapture cap;

cap.open(0);

//sliding window history logic

while (1) {

//transfer feed to pre-intialized matrix

cap >> frame;

//image enhancement of gain (contrast) and bias (brightness) adustments

//constructor : resulting (enhanced) image, ??, alpha/gain, beta/bias

//frame.convertTo(frame, -1, 0.5, 0);

//make a copy of frame for later comparision

frame.copyTo(res);

//exit if buffer exceeds 100

if (counter\_history == 100) {

std::cout << "Applciation timed out. buffer exceeded 100 frames." << counter\_history << endl;

return 0;

}

//make a copy of frame for sliding window history

waitKey(1); //buffer for not crashing system

frame.copyTo(history[counter\_history]);

//wait untill 10th frame is obtained. to initialize history array

if (counter\_history > 10) {

//output frame number for debugging purposes

std::cout << "Frame feed No. " << counter\_history << endl;

////MOG implenetation

////constructor : input image, resulting mask, learning\_rate

// pMOG->operator()(frame, initial\_mask, 0.8);

//

// //enhancment of mask using histogram equlzation - constructor : input image (1 channel), output image (1 channel)

// //Apply Histogram Equalization

// equalizeHist(initial\_mask, histogram\_mask);

// //noise reduction of salt and pepper using median blurring

// //constructor - input image (1 channel), output image (1 channel), k size (?)

// medianBlur(initial\_mask, noise\_free\_mask, 5); //using input without histogram initialization

// //medianBlur(histogram\_mask, noise\_free\_mask, 5); //using input with histogram initialization

// //test for checking channels in image matrix

// ////std::cout << initial\_mask.channels() << endl;

// ////std::cout << histogram\_mask.channels() << endl;

//MOG2 implementation (image segmentation)

//constructor : input image, resulting mask, learning\_rate

pMOG2->operator()(frame, initial\_mask, 0.2);

//enhancment of mask using histogram equlzation

//constructor : input image (1 channel), output image (1 channel)

equalizeHist(initial\_mask, histogram\_mask);

//noise reduction of salt and pepper using median blurring

//constructor - input image (1 channel), output image (1 channel), k size (?)

medianBlur(initial\_mask, noise\_free\_mask, 5); //using input without histogram initialization

//medianBlur(histogram\_mask, noise\_free\_mask, 5); //using input with histogram initialization

//test for channels in image matrix

//std::cout << mask.channels() << endl;

//std::cout << histogram.channels() << endl;

//Morphological opreations

//dilation

//constructor : input image (1 channel), dilated image (1 image)

dilate(noise\_free\_mask, noise\_free\_mask, MORPH\_CROSS);

//erosion

//constructor : input image (1 channel), dilated image (1 image)

erode(noise\_free\_mask, noise\_free\_mask, MORPH\_CROSS);

//Alpha Channel usage for traspareny addition of mask. smoothening

//alpha is the transperny value. value of stimulated alpha channel

//reconsider this?????

/\*double alpha=0.05, beta;

beta = (0.5 - alpha);

addWeighted(frame, alpha, history[counter\_history - 10], beta, 0.0, history[counter\_history - 10]);\*/

//Overwrite last frame from sliding window using generated mask of segmentation

history[counter\_history - 10].copyTo(frame ,noise\_free\_mask);

//output results

imshow("Orignal Image", res);

imshow("Final processed image", frame);

imshow("Final mask", noise\_free\_mask);

}

//event handling on preview

switch (waitKey(10))

{

//capture image at space bar

case 32:

//freeze output window from feed

imshow("Orignal Image", res);

imshow("Final processed image", frame);

imshow("Final mask", noise\_free\_mask);

std::cout << "Image written from feed. frame no. " << counter\_history << endl;

imwrite("test\\captured\_image\\/orignal iamge.jpg", res);

imwrite("test\\captured\_image\\/final processed image.jpg", frame);

imwrite("test\\captured\_image\\/final mask.jpg", noise\_free\_mask);

break;

//exit at esc

case 27:

return 0;

}

//increment frame count

counter\_history++;

}

//unconditional loop for uninterupted webcam feed

while (false) {

cap >> frame;

//image enhancement of gain (contrast) and bias (brightness) adustments

//constructor : resulting (enhanced) image, ??, alpha/gain, beta/bias

frame.convertTo(frame, -1, 0.5,0);

////MOG implenetation

////constructor : input image, resulting mask, learning\_rate

// pMOG->operator()(frame, initial\_mask, 0.8);

//

// //enhancment of mask using histogram equlzation - constructor : input image (1 channel), output image (1 channel)

// //Apply Histogram Equalization

// equalizeHist(initial\_mask, histogram\_mask);

// //noise reduction of salt and pepper using median blurring

// //constructor - input image (1 channel), output image (1 channel), k size (?)

// medianBlur(initial\_mask, noise\_free\_mask, 5); //using input without histogram initialization

// //medianBlur(histogram\_mask, noise\_free\_mask, 5); //using input with histogram initialization

// //test for checking channels in image matrix

// ////std::cout << initial\_mask.channels() << endl;

// ////std::cout << histogram\_mask.channels() << endl;

//MOG2 implementation

//constructor : input image, resulting mask, learning\_rate

pMOG2->operator()(frame, initial\_mask,0.6);

//enhancment of mask using histogram equlzation

//constructor : input image (1 channel), output image (1 channel)

equalizeHist(initial\_mask, histogram\_mask);

//noise reduction of salt and pepper using median blurring

//constructor - input image (1 channel), output image (1 channel), k size (?)

medianBlur(initial\_mask, noise\_free\_mask, 5); //using input without histogram initialization

//medianBlur(histogram\_mask, noise\_free\_mask, 5); //using input with histogram initialization

//test for channels in image matrix

//std::cout << mask.channels() << endl;

//std::cout << histogram.channels() << endl;

//preview tempororary results

//imshow("salt", noise\_free\_mask);

//????????????????? here frame from previous 10th last frame needs to be copied using max to current frame??????????????????

//frame.copyTo(processed\_image, noise\_free\_mask);

//imshow("captured", noise\_free\_mask);

//event handling on preview

switch (waitKey(10))

{

//capture image at space bar

case 32:

//imwrite("test\\captured\_image\\/processed\_image.jpg", noise\_free\_mask); //save image as jepg

imshow("captured", processed\_image); // show image in seprate windo

break;

//exit at esc

case 27:

return 0;

}

}

}

static public void main(String[] args) {

try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());

}

catch(