image blur detection

by sourav mohanta

Submission date: 17-May-2020 02:13AM (UTC+0700)
Submission ID: 1325741786
File name: end.docx (1.31M)
Word count: 3855
Character count: 20600

1. INTRODUCTION

1.1 PROBLEM DEFINITION

Images in the world are subject to numerous forms of squalor during image capture, attainment and reproduction for several reasons. There are different types of attributes or properties of an image including blur, hue, sharpness, contrast and saturation, which are directly related to different type of degradation s. The task of an programmed image quality assessment system is to make a steadfast decision on image quality in near real time with least possible human manual participation.

As per the basic definition, a visual effect that makes the edges of images appear less visible or out of focus is referred to as a Blur**.** It is a general image degradation caused by low-quality lenses or intentional photographing of moving or distant objects or highlighting certain objects. Blur is a very common image deprivation problem when capturing the photos. The main explanations caused image blur are shaky camera, object movement, out-of-focus and low mega-pixel cameras. Thus, the levels of the images are vividly. On the contrary, everyday users are easier to blur the whole images because of the camera activities caused by pushing shutters. Apparently, this kind of blur is unintentional.

When image gets blurred, it basically makes the colour transition from one side of an edge in the image to another smooth rather than sudden. The effect generally occurs to average out fast changes in pixel intensity.

Many digital images contain unclear regions which are produced by motion or de-focus. Automatic detection and organization of blurred image areas are very important for various multimedia analysing tasks. Before attempting to process the image for any useful information, it is critical to be certain that the image being processed is of neat quality. Among several quality properties including blur, hue, sharpness, contrast and saturation, blurriness of images deteriorates high frequency contents in the image thereby making the image useless for any information retrieval that will be useful.

1.2 PROBLEM OVERVIEW

In order to develop dependable blur detection it is imperative to understand the image deprivation procedure. Deprivation utilities may be due to camera motion or defocus blur. Noise in the image is inevitable as the sources of noise mostly twig from the imaging process itself and may be due to the imaging process (quantization effects), dark current, and others. For all practical purposes image noise is usually approximated to be white Gaussian.

Blurring happens when each pixel in the image gets range over the neighbouring pixels. This spreading process is more often referred to as a smearing out around the neighbouring pixels. Thus, the blurred image now has pixels that are exaggerated due to this smearing process. An image blur is represented as a accurate convolution between the source image and the point spread function which is known as the blurring kernel. Motion blur is caused by the relative motion among the entity and the camera. The image so obtained contains an integration of pixel intensities that moved during the period of exposure governed by the camera's shutter speed. A motion blurred image is easily distinguishable to human eyes as such an image will be evidently blurred or smeared in the direction of motion. The PSF approximation of the motion blurred image will benefit categorise the number of pixels by which motion occurred and the direction of the blur.

Images that are captured due to a reduced convergence of light from an object on the image device plane results in what is identified as out-of-focus or defocus blur. In optics, focus is defined as the point where the light rays that might be coming from a point source will converge. The effect of a defocus blur on an image will cause the pixel intensities to be scattered around its neighbours in a circle.

The noise and degradation function have contradicting effects on the image spectrum. Noise often introduces additive broad band signals in the image data. Most degradation functions have the effect of averaging out the image data and act as a low pass filter.

1.3 HARDWARE SPECIFICATIONS:

- PC:
  - Multi core processor clocked higher than 2.0 GHz
  - 8 GB of ram
  - Intel GPU
- PHONE:
  - Working android OS with USB debugging allowed

1.4 SOFTWARE SPECIFICATIONS:

- Windows 7 or above
- Python libraries :
- NumPy 1.15.4
- SciPy
- Matplotlib 3.0.2
- OpenCV
- imutils
- Android studio 3.0 or higher with
- Open CV plugin for android

2. LITERATURE AND SURVEY

2.1 Existing System:

Processing blurred images is a key problem in most image applications. Fundamental approaches which are to acquire the blur invariants which are invariant with admiration to centrally symmetric blur are typically built on geometric moments or compound moments.

Can we combine one blurred image and one noisy or dark image in order to get a desirable image?

Answer - In our app for now this functionality is not available however various researches and results shows that “yes” we can get clear image from the meta-data of a blurred and noisy image.

- The approach is image de-blurring with the help of the noisy image. First of all, both images are used to evaluate and obtain a precise blur kernel, which otherwise is stimulating to acquire from a blurred image. Secondly, and again using both images, a residual de-convolution is anticipated to suggestively reduce resounding artefacts characteristic to image de-convolution.

Third, the residual resounding artefacts are in even image regions which are auxiliary bottled-up by a gain-controlled de-convolution procedure.

Existing Problem of Blurred Image-

-Certainly technologies have grown vastly, so as the solution for every single problem. A decade before this problem has been addressed in the form of various algorithms.

-However the normal consumer won’t be able to pay a vast price to use the specific Algorithm.

- There are various app which may have functionality of detecting of blurred image but it’s not purely meant for blurred image detection. This is where our app comes into picture its specific and only functionality is to judge whether an image is Blurred or not.

Laplacian Method:

-The input image is anticipated to be despoiled by additive zero mean Gaussian noise. To exclude the chances of structures / details from contributing to the noise variance estimation, or a simple edge detection algorithm using first-order gradients is applied initially.

-Then a Laplacian operator followed by an averaging over the whole image will provide us with very accurate noise variance estimation. There is only one parameter which here is self-determined and adaptive to the image contents.

-The results show that this projected algorithm achieves well for dissimilar types of images over a large variety of noise alterations.

- The method mainly composed of three steps. Firstly, the Laplacian pyramids from each source image are deconstructed separately after which each level of new Laplacian pyramid is fused by adopting into different fusion rules.

- To the top level, it adopts the maximum region information rule while to the rest levels, it adopts the maximum region energy rule.

-Finally, the fused image is obtained by the inverse of Laplacian pyramid transform. Two sets of images are applied here to verify the fusion approach proposed and compared it with other fusion approaches.

-By examining the investigational results, it showed that this method has good enactment, and the quality of the fused image is improved than the outcomes of other methods.

- Distribution of low and high occurrences — if there are a little amount of high frequencies, then the image can be reflected blurry. However, defining what a low figure of high frequencies are besides what a high figure of high frequencies are can be pretty problematical, often prominent to sub-par results.

Use of deep learning-

-Advancement of computational power and big datasets brings us the opportunity of using deep learning to do image processing.

-We used profound convolutional procreative antagonistic networks (DCGAN) to do various image exemption errands like super-resolution, demonising and de-convolution.

-DCGAN allows the use of a single architecture to do different image processing tasks while achieving competitive PSNR scores. While the results of DCGAN are slightly lower PSNR compared to traditional methods, images produced by DCGAN appear more appealing when viewed by human.

-DCGAN can absorb from big datasets and inevitably add high-frequency details and features to images while out-dated methods can't.

-The generator discriminator design in DCGAN pushes it to produce more accurate and appealing images.

A lot of examiners had tried to do de-convolution expending convolutional neural network. One class of approaches is to try to use deep learning to envisage the stricture of the blur kernel.

-They projected a new CNN design that can do convolution with non-local kernel. On the other hand, in case of a specific dataset, traditional CNN had been proven to be useful. For example, Hardy et al. used CNN to do direct text de-blurring.

Also, Most image de-blurring methods assume knowledge of the point spread function (PSF) causing the blur. In this work we discourse the problem in recognizing the characterizing stricture present in the PSF, which resembles to motion or out-of-focus blur, from blurred and noisy images.

-The surveillance in the spectra of these blurring utilities is shown to have episodic (or almost episodic) zeros is the basis of an already known blur documentation method in the kestrel domain. However, this method is found to be highly noise sensitive.

2.2 PROPOSED SYSTEMS

Existing project works as an platform to identify blurred image using python ,and machine learning. The various theories used to identify the problem as well as solution are mentioned below.

Many researches have used many methods to compute this "blurriness metric", some of them simple and straightforward using just basic grey-scale pixel intensity statistics, others more advanced and feature-based, evaluating the Local Binary Patterns of an image.

-After a quick scan of the paper, we came to the implementation that I was looking for: variation of the Laplacian by Pech-Pacheco et al. in their 2000 ICPR paper named, Diatom autofocusing in bright field microscopy a comparative study.

The method is simple. Straightforward. Has sound reasoning. And can be executed in only a single line of code:

You simply take a single frequency of an image (seemingly grey-scale) and convolve it with the subsequent 3 x 3 kernel:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$ The Laplacian kernel.

And then take the variance (i.e. standard deviation squared) of the response.

And here If the variance falls below a pre-defined threshold, then the image is considered to be blurry; otherwise, the image is not blurry.

-The reason this method works is due to the definition of the Laplacian operator itself i.e. used to measure the second derivative of a given image. The Laplacian operator highlights regions of an image containing rapid intensity changes, much like the Sobel

and Scharr operators. And, just like these operatives, the Laplacian is regularly used for edge detection.

-We are pretentious here is that if an image contains high alteration then there is a wide spread of retorts, both edge-like and non-edge like, illustrative of a normal, in-focus image. But if there is very low amendment in the image, then there is a very tiny spread of responses, representing there are very little edges in the image. And we all know, the more an image is blurred, the less edges there are.

-Perceptibly the trick here is setting the precise threshold which can be quite province reliant on depending what you do. Set Too low of a threshold and you'll incorrectly mark images as blurry when they are not.

-Set Too high of a threshold then images that are actually blurry will not be marked as blurry. So, This method tends to work best in environments where you can compute an acceptable focus measure range and then detect outliers.

2.3 FEASIBILITY STUDY

2.3.1 Financial Feasibility: Being a single platform application it has a few financial capabilities. Since the scheme doesn't consist of any combination data transfer, bandwidth required for the process of this request is very low. The system will simply follow the freeware software standards as default. No cost will be indicted from the prospective customers. Bug fixes and sustaining errands will have an accompanying cost. At the preliminary stage the prospective market space will be the local academes and higher edifying institutes. Beside the associated cost, there will be many benefits for the included applications to use the technique with their own features. From these it's clear that the project is financially feasible.

2.3.2 Technical Feasibility: The project image blur detection is a machine learning based application. The main technologies and tools that are associated with it are python libraries like OpenCV ,NumPy, Android Studio, MathPlot Lib ,imgutils . Each of the technologies are liberally accessible and the methodical skills required are adaptable. Time margins of the product improvement and the ease of instigating using these technologies are co-ordinated. Initially it will be done in a stand-alone addition to photo storing and display apps as gallery but can be then added to camera applications directly to stabilize or re-shoot. From these it's clear that the project Image Blur Detection is technically feasible.

2.3.3 Resource and Time Feasibility: Properties those are essential for the project includes,

- Programming device (Desktop or Laptop)
- Hosting space (Available for free)
- Programming tools (Available for free)
- Programming individuals

So it's clear that the project has the obligatory source feasibility.

2.3.4 Risk Feasibility: It can be discussed under several contexts.

Risk associated with size:

Estimated size of the product in line of codes: Being an android application with much number of partitions, this project will contain significant amount of code lines. As the system doesn't contain any multimedia aspect, the file sizes and the complete project size will not exceed 30MB.

2.3.5 Social/Legal Feasibility this project uses freely available development tools, and provides the system as an open source system. Only the preservation cost will be indicted from prospective customers. Training software libraries that are used in this system are free open source libraries.

SYSTEM ANALYSIS AND DESIGN

3.1 Requirement Specifications:

3.1.1External Interface Requirements

User Interfaces

The user interface will run on android 9 or higher.

Hardware Interfaces

The application requires a camera. The camera is assumed to have necessary driver installed within the OS. Also, it requires 1 USB port on the PC.

Software Interfaces

There are no external software interface requirements on android os.

Communication Interfaces

There are no external communications interface requirements

3.1.2 Functional Requirements:

This section describes major functional requirements of the system:-

- The application should take an image from the user.
- The application should be able to process any image, and then produce output. The whole processing of the image is to be done using OpenCV .
- Application should handle multiple stream request and user inputs efficiently.

3.1.2 Non Functional Requirements:

Security

The computer that runs the package will have its identifiable security. Only the System Admin will log in to the system with his/her username and password. The person will access to view the output.

Maintainability

As a tool to obtain, for the ease of maintainability UML diagrams will be used in the development process.

Portability

To ensure portability, the application will be developed in JAVA android language.

Efficiency in Computation

This software shall take minimum use of Central Processing Unit (CPU) and memory resources on the operating system. When it is executing, the software shall utilize less than 80% of the system's CPU resource and less than 500 megabytes of system memory.

Performance

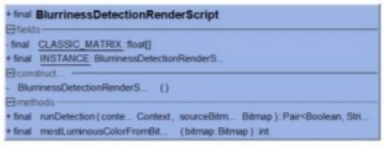
This software shall minimize the number of calculations which are needed to perform any image processing. Each captured image shall be processed within 1 second to achieve performance.

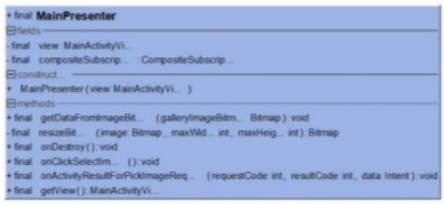
Usability

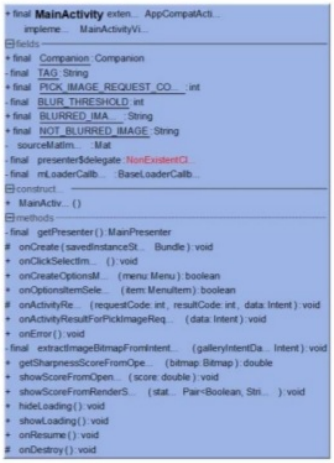
This application shall be easy to use for all the users with minimal instructions. And 100% of the languages on the graphical user interface (GUI) shall be intuitive and understandable even by non-technical users.

3.2 UML DIAGRAM:

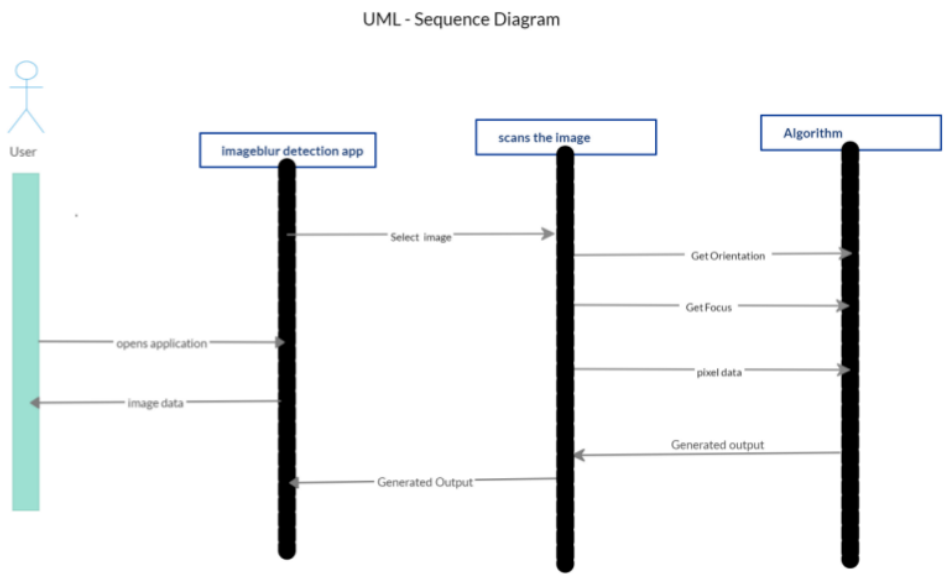
Class diagram:-



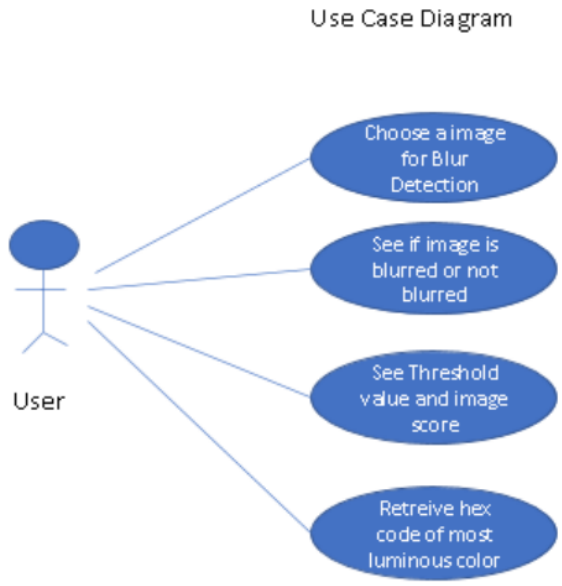




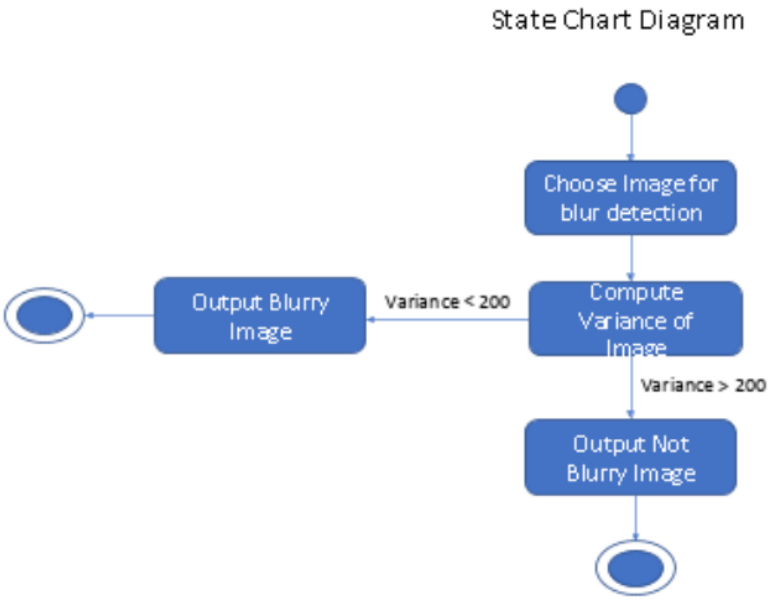
Sequence Diagram;-



Use Case Diagram:-



State Chart Diagram



3.3 DESIGN AND TEST STEPS:

It is designed to be used with least steps and proper accuracy ,such that the application is not only a standalone process to detect ,but it can later be also included into camera modules and other imaging devices .

ASSUMPTIONS:

This is an attempt to amalgamate an algorithm for automatic ranking of cardinal photographs, based on the following basic conventions about quality:

- Decent photos have the subject matter in focus;
- The focal mass will usually lie close to the points definite by the rule of thirds;
- The focus matter will also be detached from the background by brightness and saturation.

We note that the sharpness of an area is a rational interpreter of focus, as most projecting subject matter comprises detailed edge structures that are damaged by out-of-focus/motion blur.

OpenCV – Open Source Computer Vision

Open Source Computer Vision, also known as OpenCV, is a real time computer vision library with many image processing functions which are developed by Intel for the C++ or Java programming platform . This API provides many functions to perform calculation on captured image processing on each frame to support software. The blur detection components used in this project are heavily supported by some of the functions built into the OpenCV library .

Software Constraints

Although the software utilizes various methods of filtering and blur detections, the system still varies in output under different environmental light. This system cannot guarantee to perform correct under very bright light source such as direct sunlight or having bright lights in the background in the case of windows. These factors affect the algorithm due to the fact that the light removes the edges needed for detection, thus rendering the blur detector ineffective to detect accurately.

Unit Testing

Unit testing was done after the coding phase. The purpose of the unit testing was to locate errors present in the current module, independent of the other modules. Some changes in the coding was done during the testing phase. Finally, all the modules were tested individually following bottom to top approach, starting testing with the smallest and lowest modules and then one at a time.

Integration Testing

Once the unit was over, all the modules were then integrated for integration testing. External and internal boundaries are executed and work as per design, the presentation of the module is not degraded.

Validation Testing

At the beginning of the integration testing, the application is said to be completely assembled into a package; interfacing and then the errors have been uncovered and corrected. After which a final series of software test, validation tests were carried out.

Acceptance Testing

This is the last stage in the whole testing process before the system is accepted for operational use. Any requirement or definition problem revealed from acceptance testing are considered and then they are made error free.

3.3 ALGORITHMS AND PSEUDO CODE

Step 1- import the required classes and libraries .

Step 2- Select the method to display the correct user interface to enter the image.

Step 3- Take the in the luminosity data from the image.

Step 4- Set an activity to Take in the image set to a matrix.

Step 5- Use the image to generate a sharpness score using the laplacian operator.

Step 5.1- get sharpness score and pair it with the given threshold.

Step 5.2- Repeat process from step 3 at multiple points.

Step 6- If the average Sharpness score from the operator is

Step 6.1- less than the given threshold Set the Output variable to Blurry.

Or

Step 6.2- If more than the given threshold Set the Output variable to not blurry.

Step 7- Display the final result along with the selected image .

3.4 TESTING PROCESS:

The testing was done on a number of images, to obtain the correct threshold for the application so as to detect the blur successful . The dataset consists of undistorted, naturally-blurred and artificially-blurred images which are used for image quality with changes related to particular tests and assessment purposes. Download the dataset from here: http://mklab.iti.gr/files/imageblur/CERTH_ImageBlurDataset.zip

Threshold is an important parameter to verify the blurriness of image.

- If you choose Too small of a threshold value, and you'll accidentally mark images as blurry which they are not.
- While with a overly large of a threshold, you'll be marking images as non-blurry even when they are.

We have also used two methods for testing the operator:

Variation of the Laplacian

Using OpenCV2

This method yielded an accuracy of **87.29%**

```
cv2.Laplacian(img, cv2.CV_64F).var()
```

The Laplacian Operator is applied on each of the images. The variation of the result is calculated. If the variation is below the threshold, 400 in this case, the image is categorised as blurry. Otherwise, it is classified as non-blurry.

Maximum of Laplacian

Using OpenCV2

An accuracy of **63.58%** is achieved using this method.

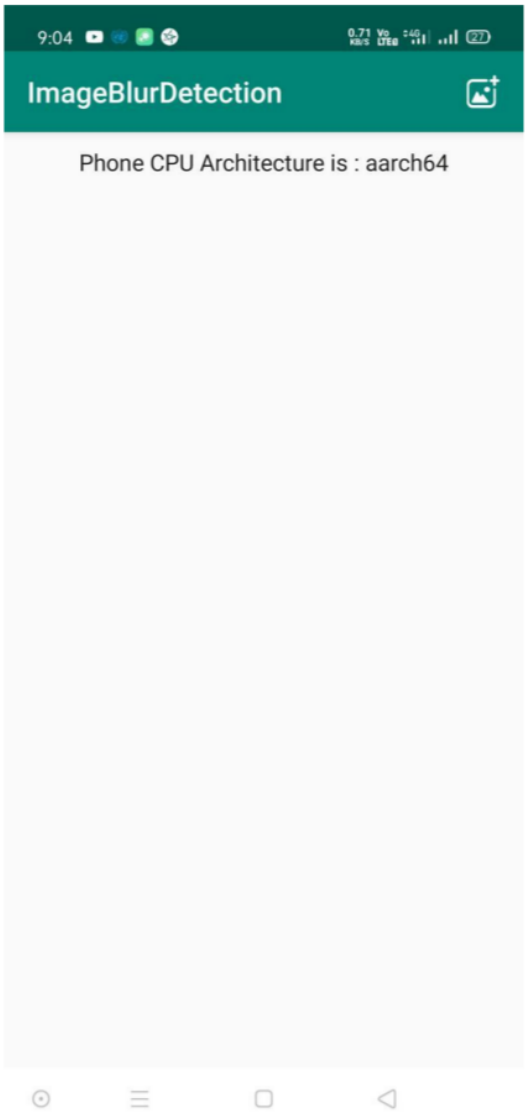
```
gray=cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

numpy.max(cv2.convertScaleAbs(cv2.Laplacian(gray,3)))
```

The outcome of the Laplacian Operator is transformed to an absolute scale (0-255). The max of the standards is taken for each image. The threshold is set at 216. Values lower than 216, classify the image as out-of-focus or blurry. Greater than 216 is classified as non-blurred.

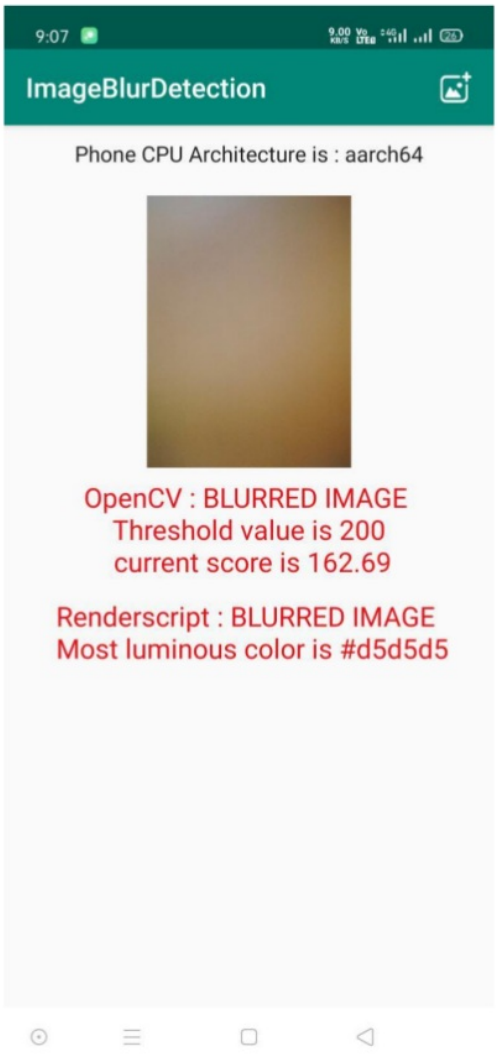
4. OUTPUTS:

After successfully building the application, we got the following output:

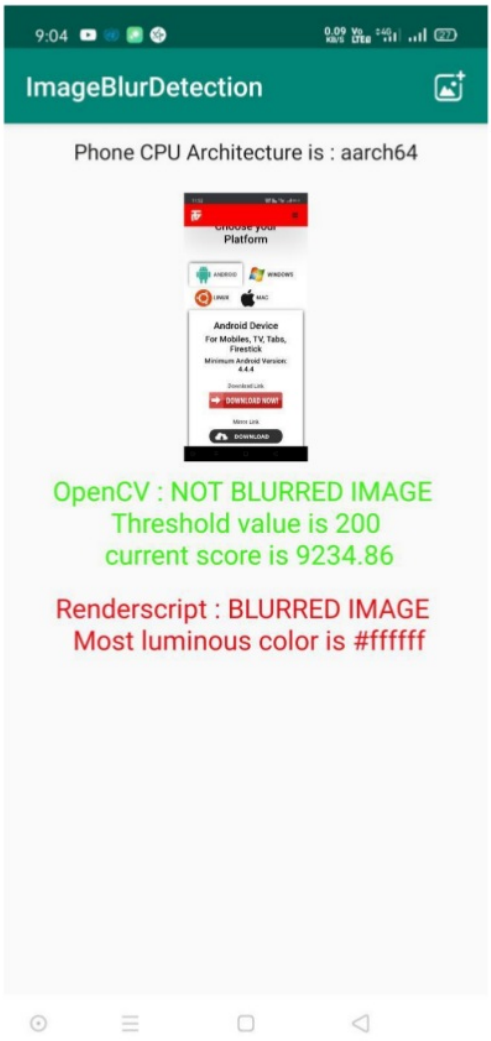
At idle:



With a blurry image as input :



With a non-blurred image as input:



5. CONCLUSIONS:

The problem of automatic blur detection finds application in modern digital cameras that can detect for bad images and automatically discard them, time critical image inspection, biomedical applications in detecting images that even any experienced cardiologist will not be able to say whether an image is good or not.The problem of blur detection has been pursued by computer vision researchers for a very long time. The ability of a computer to detect between a good and bad image data is a challenging task especially in a time critical real time applications.

Also, edges in the image get severely affected on the event of an image blur and lose their sharpness and sometimes edges tend to completely disappear. Based on, this study has focused on identifying the different edge types and the effect of blur on these different edge types viz. Dirac-structure, Roof-structure and Step-structure edges. Our effort has been to use whatever information is available on the image and then finding a definitive and consistent result.

This report investigated the solution to using integrated hardware components with the personal computer or mobile phone. The project presented a program that allowed user to perform easy differentiation through various algorithms and detection techniques, the program achieved simple blur detection in an image executed by the user.

6. REFERENCES

https://medium.com/@sukritipaul005/a-beginners-guide-to-installing-opencv-android-in-android-studio-ea46a7b4f2d3

Pertuz, S., Puig, D., & García, M.Á. (2013). Analysis of focus measure operators for shape-from-focus. Pattern Recognit., 46, 1415-1432.

Ahmed, Kareem & El-henawy, Ibrahim & Amin, A.E & adel, hadeer. (2014). A Comparative Study On Image Deblurring Techniques. International Journal of Advances in Computer Science and Technology (IJACST), Vol.3 , No.12, Pages : 01-08. 3. 1-8.

Huang, Rui & Fan, Mingyuan & Xing, Yan & Zou, Yaobin. (2019). Image Blur Classification and Unintentional Blur Removal. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2932124.

Fan, Mingyuan & Huang, Rui & Feng, Wei & Sun, Jizhou. (2017). Image Blur Classification and Blur Usefulness Assessment. 10.1109/ICMEW.2017.8026291.

7. Similarity Report:

image blur detection

ORIGINALITY REPORT

10%	10%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.pyimagesearch.com Internet Source	10%

Exclude quotes	On	Exclude matches	< 20 words
Exclude bibliography	On		