BODY SIZE RECOMMENDATION SYSTEM

Abdul Maajid Ansari[1], Arshad Khan[2], Arshan Shaikh[3], Dr Shabina Sayed[4]

[1-3]Student, Information Technology Engineering Dept. MHSSCOE, M.H Saboo Siddik College of Engineering, Maharashtra, India
[4]Assistant professor, Information Technology Engineering Dept. MHSSCOE, M.H Saboo Siddik College of Engineering, Maharashtra, India

---

ABSTRACT: *Recent years have seen an emerging trend of people surfing online shopping portals frequently to purchase clothes, shoes, and other apparel instead of visiting the retail stores in shopping malls. Although shopping-clothes through portals is time-saving, easily accessible at any time of the day, and involves viewing various options from different brands, it has shortcomings. One of the bothering issues is choosing the correct size, as several factors have to be taken into account before selecting the suitable size. Through our contribution to this paper, we implement an Android-based mobile application that can detect the approximate Men's T-shirt size based on a photo of the backside of the T-shirt and also display a list of brands with corresponding sizes. We evaluated against several T-shirts and achieved an accuracy of around 72%.*

Key words: **Canny Edge Detection, Corner Detection, Gaussian Blur, Microsoft Azure.**

1. INTRODUCTION

The expansion of the web and mobile-based shopping portals and many available brands have given rise to a paradox of choice while shopping for clothes online. A genuine and commonly faced problem is choosing a suitable cloth size. As there are various parameters on which the size can vary, namely physique, brands, gender, age-group, and different size charts, it sometimes becomes hard to know the exact size of a particular brand for an individual. At times this could lead to selecting clothes of incorrect size from onl+ine portals resulting in inconvenience to the buyer. According to research, 70 percent of apparel is returned due to incorrect fitting [1]. It is also unusual that the buyer would measure their size on their own using a measuring tape accurately. In this paper, we develop an application to overcome these problems by providing a simple and convenient platform for users to get their approximate T-shirt size, making it easy for them while shopping online. Although there are fitting tools available with online retailers in the market such as True Fit, Upload, and Virtusize, all of them require users to manually enter their details such as height, weight, age and based on the body dimensions entered in the tool provide the best fitting size [1]. Using our application, the user has to capture the image of the backside of the T-shirt, and based on the image; our T-shirt size detection algorithm will compute the measurements and provide the approximate T-shirt size and the list of brands where the user can get that T-shirt size.

- Our approach consists of the following steps:

- Using the measurements of the reference object, a one-dollar note in our case, we compute the pixel per metric ratio. Pixel per metric acts as a scaling factor for calculating the actual physical distance.

- Next, we identify the actual position of the T-shirt on the image and detect its edges and corners.

- Using the corners, we compute the width of the T-shirt and multiply it by pixel per metric to get the actual physical measurement.

- The calculated measurement is checked against a predefined size chart to get the approximate T-shirt size (Small, Medium, Large, X-Large, XXLarge, or XXXLarge), and based on the size, our application displays a list of brands with their corresponding T-shirt sizes for the required measurement.

- We tested our application against various sized t-shirts, and we were able to achieve an accuracy of 72%. The Android phone used for testing was the Samsung J7 Prime having a camera of 13 Megapixels and a picture size in the ratio of 4:3. Future research in this area would help us overcome the existing limitations and enhance the accuracy of our implementation.

- The contributions through this paper can be summarized as follows:

- We developed an Android application that takes an image of a men's T-shirt and predicts the approximate size.

- The detection algorithm uses pixel per metric ratio and key OpenCV's functions to detect edges, corners, and contours of the T-Shirt to compute the measurements.

The application is tested on Samsung J7 Prime against different T-shirt sizes, and an accuracy of around 72% is achieved. The paper organization is as follows: We discuss the background on the key concepts of image processing and computer vision relevant to our work in section 2. Next, we move on to the implementation and design methodology in section 3 and discuss our evaluation strategy and results in section 4. We discuss the limitations of the implementation in section 5. We explore the future scope of this project and conclude the paper in section 6.

2. BACKGROUND AND RELATED WORK

Over the years, there has been diverse and extensive research in the space of object detection. A renowned research work [2] involves

- implementing a real-time object detection and tracking system,
- using real-time video processing,
- Functioning in an unknown and changing background.

In [3], a unique approach different from the conventional methodology has been discussed wherein object detection is viewed as a regression problem. A single CNN is used to predict multiple bounding boxes of an image and the class probabilities for each box. Other notable works [4] explore object detection using advanced deep learning techniques such as the faster R-CNN.

There has also been prior and ongoing research in cloth detection and size prediction using Computer Vision. One of the interesting works [5] proposes virtual cloth fitting using Microsoft Kinect with a virtual garment being superimposed on the real-time visualization of the user. The user can freely move in front of a video screen, and the user's silhouette is found according to which the virtual garment can be synthesized and adjusted. [6] Talks about using dressed-human silhouettes and height of the individual as input to predict the 3D body sizes. The 2D images involving the front and side views of the silhouettes are converted to 3D information, and the size is predicted by looking up the corresponding cloth size with the 3D body sizes on the defined size chart.

A more relevant approach to ours has been implemented by a startup called StichFix, which provides personal styling service by designing clothes based on the preferences provided by the user. In their blog [7], key steps have been discussed regarding detecting cloth size from an image. They start by detecting the bounding box used as a reference for getting the location of the cloth in the image. The next edges of the cloth are detected, and the outermost contour is calculated using the matching squares algorithm. From the contours, corners are detected, which leads to identifying which key points match which part of the clothing by using a predefined clothing template. By calculating straight-line distances between the identified key points, the physical distances are obtained, giving the cloth measurements.

Image segmentation [8] plays a pivotal role in image processing. It is dividing an image into meaningful parts called segments upon which it is easy to perform any analysis. Its applications include Object Detection and Recognition, Content-Based Image Retrieval, Automatic Video Surveillance. Several prominent image segmentation techniques such as Thresholding this technique assumes that adjacent pixels below a certain range have the same intensity level and thus belong to the same class. It is primarily used over images with contrasting backgrounds, such as lighter objects on a dark background. Boundary-based the techniques used here to consider sudden rapid changes in the intensity level at the boundary between any two regions. The commonly used technique is the edge detection technique used to detect edges in an image using operators such as the Canny, Sobel, or the Roberts.

The region-based segmentation technique divides the image into regions having similar characteristics such as grey level, color value, or texture. Split and merge is a noted region-based segmentation method that iteratively splits the image into regions having similar features and merges it with the similar adjacent regions.

An *edge* is usually defined as a change in either the image intensity or the first derivate of the image intensity. Edge Detection [9, 10] is a process of identifying sharp discontinuities in the image brightness within adjacent regions in an image. These discontinuities can be a step or line discontinuities. A step discontinuity has intensity level changing abruptly, whereas inline discontinuities, the image intensity changes abruptly but returns to its initial value after a short distance. Generally, edge detection algorithms will have the following steps:

Filtering it is concerned with the removal of unwanted noise to improve the performance of the edge detector.

Enhancement it deals with emphasizing the pixels where the image intensity changes to be able to detect them. Detection this is the step where the gradient is found, and edges are detected based on a threshold criterion. Some of the known edge detection operators are Sobel, Prewitt, Roberts, and Canny. We are using a canny edge detector in our implementation. In edge detection algorithms, there is always a trade-off between noise suppression and localization. Canny edge detector being an optimal operator

is more popular than others in providing the best balance between noise, immunity, and localization.

Canny edge detector [11], developed by John F. Canny in 1986, uses Gaussian filtering to remove noise, finds out the gradient to detect edges, and maintains good localization by ensuring that the marked edges are as close to the original edges in the real image. Following stages are involved in this algorithm Noise Reduction. The first stage involved applying a 5x5 Gaussian filter to remove noises.

Finding the Intensity gradient Next, a Sobel kernel is applied on the smoothened image to get the first derivative in both the horizontal and vertical directions. The edge gradient is simply the sum of the squares of the horizontal and the vertical derivatives. Non-maximum suppression after finding the gradient, the following step is to suppress those pixels that are not a local maximum in their neighbourhood in the gradient direction. These pixels are unwanted and can be easily ignored. Thresholding this is the final stage of the algorithm where edges are checked against a threshold to evaluate whether those are edges or not. Edges above the maximum value of threshold are identified to be edges, and those below the minimum value of threshold are discarded.

Contours are defined as a curve joining continuous point's image with the same intensity on the boundary of an image. Finding contours is an essential task in object detection and recognition. Contours can only be found out once we have identified the edges using the edge detectors.

[12] Image Smoothing using Gaussian Blur Image Smoothing or Blurring is the process of applying a low pass filter kernel on an image to remove noise and high-frequency content from the images. The value of each pixel in image is replaced by the average of the gray levels in the neighborhood defined by the filter mask resulting in an image with reduced sharp transitions in the gray levels. This is how smoothing achieves noise reduction. One

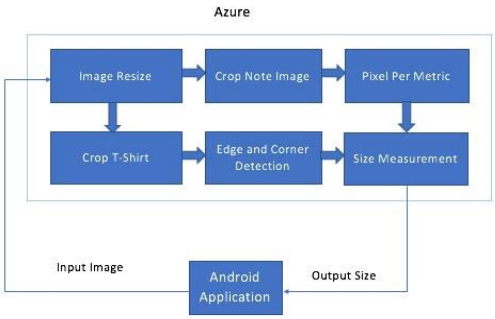

Figure 1: Application Workflow

Figure 2: Resized Image

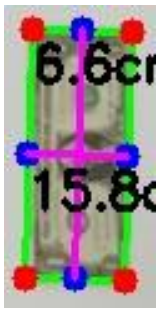

Figure 3: Reference Object Dimensions

of the popular smoothing technique to remove Gaussian noise from the image is the Gaussian blur which is achieved by convolving the image with a Gaussian kernel. [13] A Gaussian kernel is a square array of pixels where the pixel values correspond to the values of a Gaussian curve. Each pixel in the image gets multiplied by the Gaussian kernel. This is done by placing the centre pixel of the kernel on the image pixel and multiplying the values in the original image with the pixels in the kernel that overlap. The values resulting from these multiplications are added up and that result is used for the value at the destination pixel[14]

3. IMPLEMENTATION

3.1 Backend Processing

To detect the edges and corners in an image, we first resize the input image keeping the same aspect ratio. Keeping aspect ratio is very important as changing it can change the height or width of the t-shirt and can cause an error in measurements. We have tried different heights and widths while resizing and found that keeping the height of 500 and calculating width according to the aspect ratio gives the most suitable image to detect edges and corners.

After resizing, as shown in Figure 2, we apply Gaussian Blur with a 5*5 filter to reduce any noise like wrinkles on the t-shirt or background noise. Then we apply the canny detector [11] with a threshold of 100, which gives an image with edges detected which is further dilated and eroded to

enhance the edges. Then this eroded image is used to detect currency notes and the t-shirt in the image using contours. *Contour* is a curve that joins all the connected points in an image. We have used the OpenCV method to find contours with the approximation method as

CHAIN_APPROX_SIMPLE, which gives only the simple corner points of the contour region.

To detect currency note in the image, we have detected contours in the eroded image then filtered contours with an area in the range of 1100 to 8000, which depends on the camera megapixel and the height from which the input image is captured. We tried different photos to settle for the above-specified range, which satisfied most of the scenarios. However, images are captured with a more high-resolution camera or long distance, which can cause this assumption to fail. After finding the noted contour, we bound the contour by a rectangle that fits the contour and has the smallest area. Then we know the height, width, and coordinates of the top left corner of the enclosing rectangular box. Depending on the height and width of the note, we decide its orientation as if the width is more significant than height then it is horizontally aligned else vertically aligned. This note orientation helps find the overall orientation of the image, whether it is 0, 90, 180, or 270 degrees rotated. We also crop the note image to find the Pixel per metric ratio, which gives the scaling factor between the distance between the image and actual physical distance.

Using the detected currency note image, we find the Pixel per metric ratio. To find the ratio, we again find the contour and enclose the most prominent contour in a rectangular box. We calculate the Euclidean distance between the four corners to give the height and width of the currency regarding the pixels. As we know the actual dimensions for the 1 dollar currency note, we divide the pixel distances with physical distances in centimeters. This gives us the scaling factor for pixel distance to centimeters. After considering the Pixel per metric ratio, the actual distance in the currency is shown in Figure 3, which denotes 6.6 cm width of the currency and 15.8 cm is the height for a vertically oriented one dollar currency note.

After finding the Pixel per Metric ratio, we find the t-shirt in the image similarly to find the currency note by finding contour and then filtering on the area. For the t-shirt, we have observed that the contour pixel area is above 50000. As we enclose the contour in the bounding box, we know the height and width of the enclosing box. We then crop the image to select only the t-shirt with a buffer of 10 pixels on each side. Cropping the t-shirt from the main image helps us mark edges and detect corners in the t-shirt more efficiently and accurately. Cropped Image of the T-Shirt is shown in Figure 4.

For our size detection algorithm to work perfectly, we have to know the orientation of the t-shirt. As we have limited the user to keep the currency note just above the t-shirt neck portion, we can find t-shirt orientation using the reference currency note. Depending on the top left corner of the currency note and its orientation, we decide whether the t-shirt is 0, 90, 180, or 270 degrees oriented.

For predicting the size, we have measured chest distance and distance between the last two endpoints of the t-shirt. We have applied Gaussian Blur 3 times to remove noise or any shadow effect and then have detected edges using the Canny edge detector. The blurred image and Edge detected image are shown in Figure 4 and Figure 5. We then detect strong corners in the edge image using goodFeaturesToTrack [16], as shown in Figure 6. To calculate the distance, we then classify the corners in 4 quadrants depending on the t-shirt orientation. There are two distances to find first: the chest distance, and second is the distance between the last two points.

To calculate the chest distance, we have sorted the corners detected in the I and II quadrants such that we have corners representing the edges marked by the sleeve and the t-shirt. We then take the Euclidean distance between those two points and then divide it by Pixel per Metric to give the actual physical distance. Similarly, to calculate the physical distance between the last two points representing waist, we take the Euclidean distance between the points in the III and IV quadrant and then divide by the Pixel per Metric.

We have experienced from the user trials that the chest's points are not always appropriately detected, but the waist points are detected each time accurately. To accommodate this, we have put a particular condition that if the calculated chest distance is greater than the last two points distance, we select the last two points distance as the final unit for predicting size. Here we have assumed that a Men's t-shirt is straight from the chest until the end.

After getting this distance, we have then compared it with standard measurements for Small, Medium, large, X-Large, XXLarge, and XXXLarge sizes and output the size that has the most negligible difference. We then return the size to the Android application.

3.2 Android Implementation

We have designed and implemented Photo based T-shirt size detector application for Android Phones. We used Android Studio for our application development and tested our application on Samsung J7 Prime and Moto X Style. The application consists of three screens as shown in figures 8, 9 and 10.The first screen as shown in figure 8 allows the user to capture an image or upload an existing image from his phone for the t-shirt size prediction. We have integrated the

Figure 4: Cropped image of T-shirt

Figure 5: Blurred Image

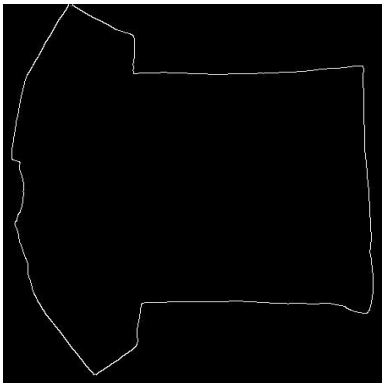

Figure 6: Edge Detection

camera module into our application for capturing the image. The user would see the preview of the image he captured or uploaded from the phone. The second screen as shown in figure 9 would have an option of retaking the image if the user is not satisfied with the preview image. We have built a RESTful API which gets invoked if the user hits the upload image button on the second screen. The basic idea behind building an API and deploying our code on cloud

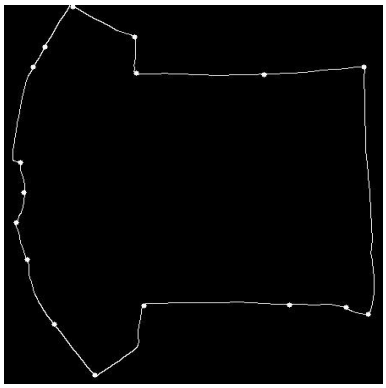

Figure 7: Corner Detection

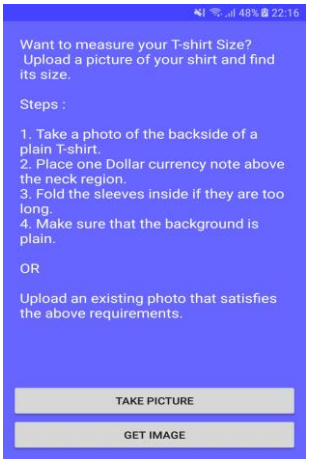

Figure 8: Capture an Image or Upload an existing one

Figure 9: View The Preview Image and Upload image to cloud

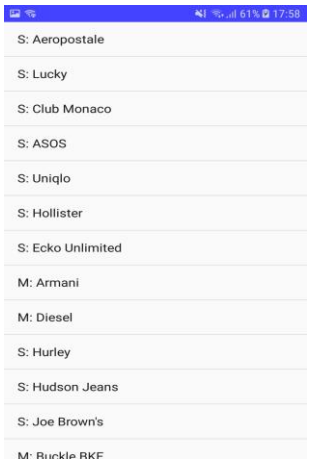

Figure 10: View the results

was to keep the application lightweight and computationally inexpensive. We tried integrating OpenCV on Android side and skipped the computation on the cloud side but the size of the android application increased way beyond 40 Mb. Hence, we scraped this idea of OpenCV integration on Android side. The current implementation includes transferring an image from the android to Microsoft Azure cloud where we have deployed our application code for t-shirt size detection. The algorithm takes an input image from Android and computes the size of the T-shirt. This result is returned back to Android where we output the associated brands and their sizes on third screen of our application as shown in figure 10. We have also handled any erroneous return from the cloud side and our android application would give a dialogue message which would help the user to take the necessary actions. This includes error messages due to poor network connectivity and non Tshirt images being uploaded to the cloud.

3.3 Cloud Implementation

As discussed in the prior section, the key objective of running our algorithm for t-shirt size detection on cloud was to make the android application lightweight and computationally inexpensive. Currently we are sending the input image to cloud using RESTful

API and the algorithm for evaluating t-shirt size is being deployed on cloud. The server has a 64-bit processor and is configured to run a maximum of four cores. The operating system running on the server is Microsoft Windows NT 10.0.14393.0.

4. EVALUATION AND RESULTS

We developed an application in Android for t-shirt size detection from an input image. We measured the accuracy of our implementation by taking several t-shirts ranging from size small(S) to Double Extra Large (XXL). For compatibility, the prototype was tested to see whether it works between different versions of Android.

The primary device used for testing was Samsung Prime J7 with Android open source version 7.0.1. We also tested our application

Figure 11: Cropped Design Image

Figure 12: Incorrect corner identification due to design

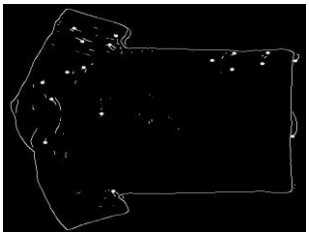

Figure 13: Incorrect corner identification due to lot of wrinkles

on Moto X Style and both devices were uploading image to cloud using same Wi-Fi access point. The Moto X Style had Android open source version 5.0.1. Both the smartphones had 13 Megapixels rear camera but Motox X Style had Qualcomm SnapDragon HexaCore Processor while Samsung J7 Prime had Qualcomm SnapDragon Octacore Processor. The server has a 64-bit processor and is configured to run a maximum of four cores. The operating system running on the server is Microsoft Windows NT 10.0.14393.0

The testing was done on many t-shirts and the results obtained are shown in Table 1.

Table-1: Result

Size	Correctly Classified	Incorrectly Classified
S	2	0
M	4	3
L	5	2
XL	1	0
XXL	1	0

4.1 Analysis of wrong prediction

Our implementation predicts wrong sizes for light coloured t-shirts as the edge and corner detection algorithms give less accurate or incomplete results on light coloured objects. Further there were some striped t-shirts in the testing dataset which were wrongly predicted by our algorithm. There can be several other factors associated with the wrong prediction. If the user is not proficient enough in taking a photo or he has not placed the one dollar currency note above the neck portion of a t-shirt then the prediction may go wrong. The key reason behind this is improper identification of the one dollar currency whose dimensions serve a great deal in prediction of our final output. It predicts wrong sizes for printed t-shirts or t-shirts with a lot of wrinkles as unwanted corners are detected in an image.

5. LIMITATIONS

The primary limitation of this project is that user would need to take image of backside of the t-shirt. The existing implementation requires user to own a one dollar currency. The user should be proficient enough to take a good image of the T-shirt with one dollar currency clearly placed above the neck region of the T-shirt for the algorithm to give accurate results. These limitations are justified as the margin for error in T-shirt size prediction is very high as the size of T-shirt changes by a margin of 4cms difference in chest size. The current implementation is limited to only men's T-shirt

It does not give accurate results for printed T-shirts as shown in Figure 12, as it detects corners within the T-Shirt boundaries. In pursuit of making the application lightweight and computationally inexpensive, we have deployed our application on the Microsoft Azure Cloud. This required invoking an API from the android side. Hence the user needs to have an active internet connection to use our application. The benefits of doing the computation on cloud clearly outweighs the overhead of sending image data over cloud. The current implementation is platform dependent. It only benefits Android users. Further, the hardware requirements for the project requires the user to have an Android phone with at least 5 MegaPixels camera.

6. CONCLUSION AND FUTURE WORK

We developed an android application to predict the correct size for the T-shirt. We achieved an accuracy of approximately 72%. This opens up the possibility of future research in a various directions.

For example, we could train a Neural network and build a model which would help us to find the perfect size for a T-shirt. This would be helpful in greatly enhancing the accuracy of the current implementation and we could avoid using the cloud and still keep the application lightweight. With the wide popularity and support of Android's AR Core module, it would be interesting to research more on prediction size of a t-shirt in real time i.e. the application should be able to detect size from the image of an user wearing a T-shirt. Further we are planning to extend the application usage to accommodate prediction of female t-shirts. Moreover we would also like to extend the application functionality to accommodate the size prediction of various apparels. The current dependency on Android platform can be overcome in future by developing application for Windows and iOS users.

REFERENCES

[1] https://www.consumerreports.org/cro/news/2014/07/fitting-apps-help-you-getthe-right-size-the-first-time/index.htm

[2] Shashank Prasad, Shubhra Sinha: Real-time object detection and tracking in an unknown environment, 2011 World Congress on Information and Communication Technologies, 11-14 Dec. 2011

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi: You Only Look Once: Unified, Real-Time Object Detection, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 27-30 June 2016

[4] Xinyi Zhou, Wei Gong, WenLong Fu, Fengtong Du : Application of deep learning in object detection,2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS) (2017)Wuhan, China , May 24, 2017 to May 26, 2017

[5] Zhenglong Zhou, Bo Shu1, Shaojie Zhuo, Xiaoming Deng , Ping Tan, Stephen Lin : Image-based Clothes Animation for Virtual Fitting, SA '12 SIGGRAPH Asia

2012 Technical Briefs,Article No. 33, Singapore, Singapore âĂŤ November 28 December 01, 2012

[6] Dan Song,Ruofeng Tong, Jian Chang,Tongtong Wang, Jiang Du, Min Tang, Jian J. Zhang: Clothes Size Prediction from Dressed-Human Silhouettes, Third

International Workshop, AniNex 2017, Bournemouth, UK, June 22-23, 2017

[7] https://multithreaded.stitchfix.com/blog/2016/09/30/ photo-based-clothingmeasurement/

[8] Dilpreet Kaur, Yadwinder Kaur: Various Image Segmentation Techniques: A Review , IJCSMC, Vol. 3, Issue. 5, May 2014, pg.809 âĂŞ 814

[9] Lanjakorn Sewata, Salil Boonbrahm : The extracting of shirt composition using Canny edge detection with close contour, ACIS 2013 : The Second Asian Conference on Information Systems

[10] http://www.cse.usf.edu/ r1k/MachineVisionBook/MachineVision.files

/MachineVision_Chapter5.pdf

[11] Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679âĂŞ698, 1986.

[12] https://docs.opencv.org/3.3.1/d4/d73/tutorial_py_cont ours_begin.html

[13] Rafael C. Gonzalez, Richard E. Woods: Digital Image Processing Second Edition [14] https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filter ing.html

[15] J Canny. A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp. 679-698 (1986).

[16] J Shi and C. Tomasi. Good Features to Track. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 593-600, June 1994.