Depth-Driven Computational Imaging: Portrait Mode and Privacy Filter Leveraging Focal Stack and Point Cloud Data

Cameron Cipriano, Molly Housego

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

May 06, 2022

Technical Report No. 2022-01

Contents

List of Figures

1 Introduction

The field of lens-based photography has existed since the late 16th century [6] and has seen enormous technological improvements ranging from lens-filled pinholes in dark rooms to handheld/mobile cameras in today's smartphones. Consistent with this growth in camera technology, the miniaturization of all electronic and non-electronic components, and the exponential growth in computing power, photographic methods have consequently split into two categories: impulse imaging and computational imaging [7]. Impulse imaging techniques are those solely reproducible through physical camera systems by manual/automatic adjustments to shutter speed, aperture size, focus, and lens setup, such as extended depth of field (EDOF) imaging, motion deblurring, spectroscopy, light field capture, and illumination multiplexing. On the other hand, computational imaging can achieve (to varying levels of success) approximations of each aforementioned technique, with the addition of novel functionalities unachievable by conventional impulse imaging systems, such as image refocusing, perspective changes, omnidirectional field of view, recreation of scene structure through RGB+D cameras, stereo imaging, defocus, and diffusion [7].

In this exploration, we will be focusing on the novel functionalities granted by depth-based computational imaging, which has been powering modern technologies like Apple’s Portrait Mode, introduced on the iPhone 7 plus in 2016, highlighting customer demand for the ability to control the focus of an image based on depth of field. While the first iteration was not much different than how a traditional camera blurs the background of a particular subject utilizing a small f-stop, later developments introduced methods that involve merging two cameras’ images, or in the case of the TrueDepth camera on the iPhone X, an orthogonal grid of infrared dots to measure approximate scene depth. Additionally, with the extreme and growing prevalence of facial recognition software employed by large corporations with the intent of hyper-targeted marketing, government agencies for criminal identification and overall security, and even residential buildings with the intent to replace physical keys, one’s identity is becoming increasingly public [8]. Resulting from this, depth-based computational imaging can be used to develop a facial recognition “jammer”, applying a form of visual distortion, such as blurring,

pixelation, and feature morphing, to an individual's face, preventing it from being detected entirely by facial recognition software.

2 Literature Review

2.1 Portrait Mode

Projecting light onto an image to create a depth map is not a new concept, Moreno and Noguer implemented a system that projected a visible dot grid onto a scene, such that when the subject was photographed the blur and dispersion of the individual dot provides depth information directly on the image [1]. A drawback to this method was that in order to obtain the desired subject, the dots needed to be removed and underlying color and brightness information was interpolated adding distortion. However, this depth information allowed fairly accurate segmentation of the photo. This procedure allowed for not just refocusing of the foreground but any significant object in the scene as long as they are presented perpendicular to the camera.

Later Castellanos and Nguyen explored refocusing utilizing the Microsoft Kinect camera which implements an infrared light grid that would not be detectable to the human eye [2]. The use of infrared avoids needing to remove a visible grid from an image, however, the camera had limitations yielding invalid depths at subject edges. To rectify this a hole-filling algorithm was utilized setting intensity equal to the most intense pixel in a given window. Furthermore, since the depth information is coming from a different image than the RGB information source the depth map must be scaled and aligned to map the information prior to refocusing. The edge information acquired from the depth map is very important when generating focus boundaries therefore hole filling and alignment procedure accuracies are essential. Castellanos and Nguyen also utilize a machine learning alpha matte to determine if part of a scene is in the foreground or background of a focused object and improves boundary accuracy when applying artificial blurring. The limitations of this method mainly exist within the limitations of the equipment used to capture the images. The IR of the Microsoft Kinect is too weak to be applicable in scenes with heavy sunlight. Additionally, it could not capture a high enough resolution RGB image, the depth and color information had to be acquired at separate times- meaning this application could not be used for moving scenes.

Finally, there has been investigation into generating depth maps purely using information from the focal plane of an image. In theory, a rapid succession of images taken at increasing focal depth will provide depth information based on blur qualities of each image in the stack. Suwajanakorn et al. successfully utilized a common mobile phone, accounting for movement between images in order to acquire a depth map and refocus the image [4]. Later Wiberg was able to replicate the approach with a simpler process that consisted of alignment, generating an all-focused image, and a subsequent depth map with linear interpolation [5]. As computational tools improve, this method of depth perception is promising as it relies on reproducible and easily acquired equipment for image capturing when compared to light projection previously mentioned.

2.2 Privacy Filter

In order to develop a "privacy filter"/facial recognition jammer, it is important to understand the different methods for facial detection and recognition. Successful face detection algorithms of the past are heavily based on the geometrical relationships between facial landmarks, using these as the primary method of extracting facial features. As mentioned in [9], the success of these methods are highly dependent on the detection and location of these landmarks in an image, making software quite unstable in the presence of pose variation and illumination disparities. Moving from the purely geometric approach, facial recognition software began treating faces as patterns for which a more general one could find faces in an image, becoming known as the photometric approach [9]. The most successful algorithms/methods to detect faces in images under the geometric and photometric assumptions were Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Elastic Bunch Graph Matching (EBGM). While these methods are still effective, developments in deep neural networks have demonstrated a major breakthrough in facial recognition and many vision-based tasks through the use of convolutional neural networks (CNNs) [10]. From these methods, it remains clear that these methods heavily rely on facial features and landmarks to first, detect a face, and second, recognize who the person actually is. Based on this fact, a promising method of facial recognition software deterrence is facial blurring, effectively removing all geometric and photometric information these pieces of software will typically use to identify someone. In the event blurring or major distortion is not ideal, facial morphing poses another significant viable

option as a privacy filter as it introduces high correspondences between more than one person's face, causing a purposeful non/misidentification to protect one's own identity.

Facial blurring, pixelation, and other destructive methods of concealing one's identity have been implemented before as seen on television and in research done by [11], however, visually non-destructive methods of protecting one's identity are valuable in many cases of keeping images intact while protecting one's identity.

3 Problem Statement

3.1 Portrait Mode

Many Smartphones have introduced portrait mode, designed to take a high-quality image focused on a single subject in a shallow depth of field. Using depth information, could the application of portrait mode be expanded to multiple subjects at various distances? The following approach utilizes depth from defocus which uses the focal stack of a scene in order to obtain depth information to achieve multi-subject portrait mode.

The flow chart in figure 3.1.1 illustrates the sequence of processes completed to solve the proposed problem.

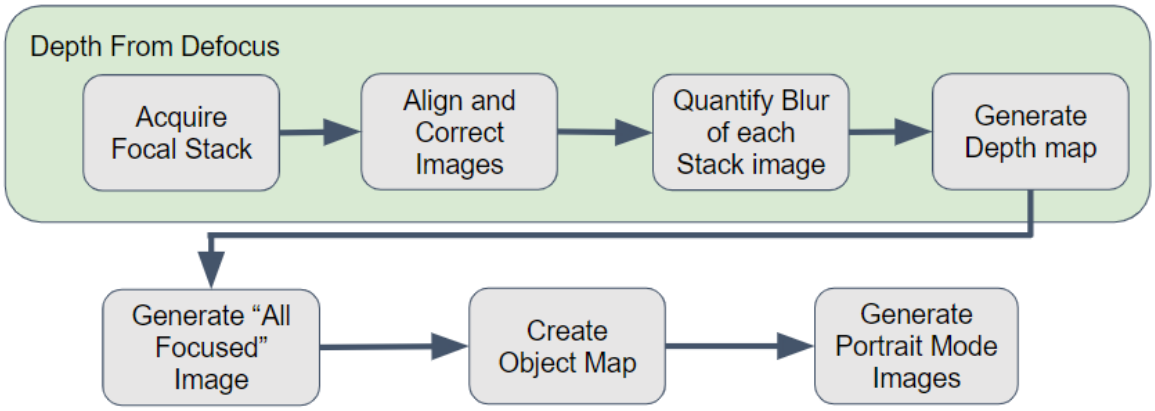


Figure 3.1.1. Portrait Mode Process Outline

Based on the proposed solution the following assumptions need to be made:

1. The Objects of interest are small and highly textured
2. The Objects of interest do not overlap visually
3. The background/ environment is low in texture

4. There are no rotational or linear translations between the images in the focal stack

3.1.1 Depth from Defocus

In order to gather optimal depth information from the blur that results from varying focal lengths of a camera lens, objects intended to be the subject of the image were positioned on the same horizontal plane at varying distances, 14 to 45 cm away from the camera. After testing a variety of capturing methods including a DSLR (manually adjusting focus and taking pictures with a remote), a smartphone (manually adjusting focus and taking pictures with a remote), the method determined to be the quickest and requiring the least amount of interference with the camera was using a smartphone via the "Open Camera" Application. The app has a setting called "Focus Bracketing" where the user is able to enter a distance range and the desired number of images and the app will iterate through taking an image at each focal length automatically, essentially creating the unmerged focal stack that will be processed for depth information. The final setup of the portrait mode scene is seen in figure 3.1.2. One data set consisting of 15 images with focal lengths ranging from 0.1 m to 1.3 m, was used as an input.

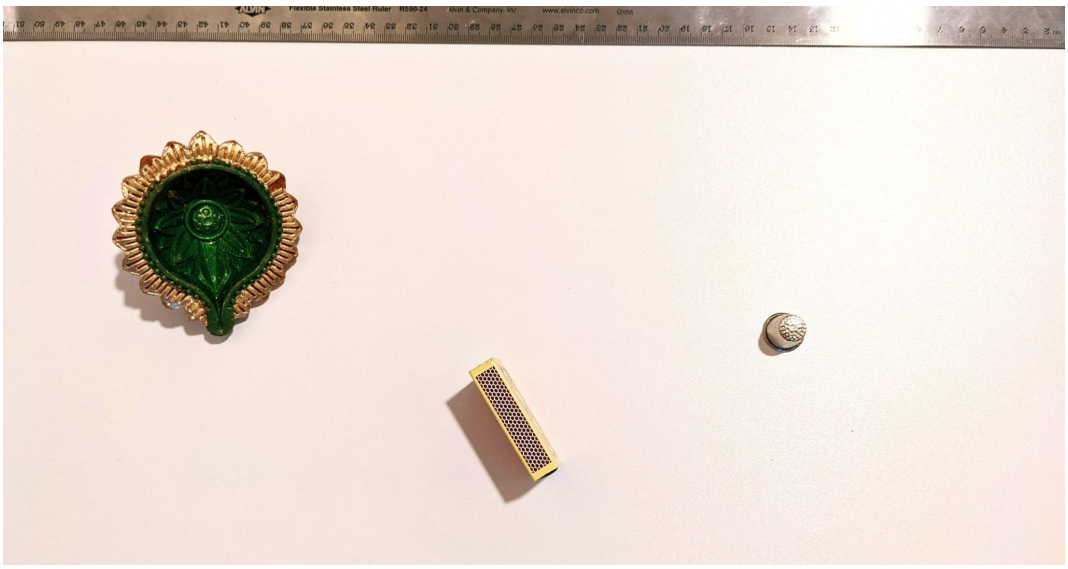
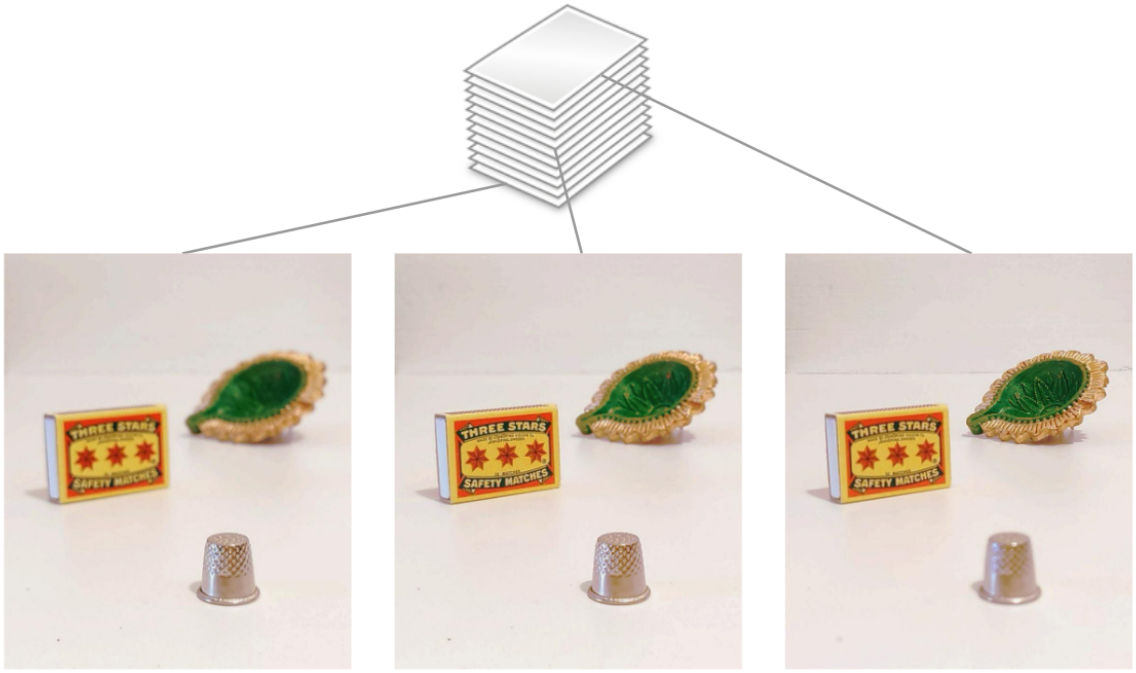


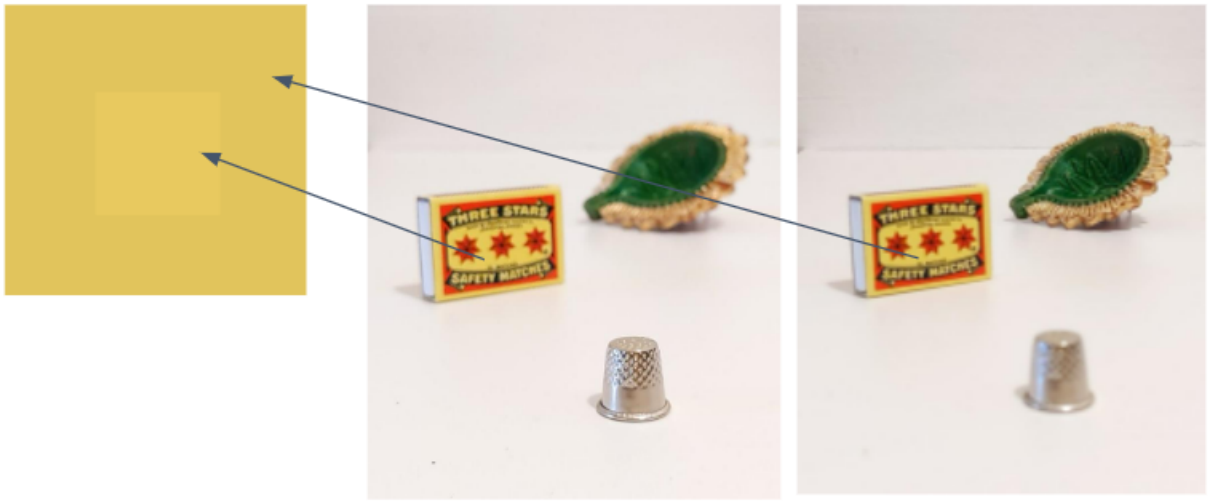
Figure 3.1.2. Portrait Mode Scene Setup



Stack Image 5 Stack Image 8 Stack image 14

Figure 3.1.3. Focal Stack Image results

While the phone was not moved during the acquisition of the focal stack, in order to adjust the depth of focus, the aperture was closing slightly with each image taken. Because of this, there is a slight inverse magnification effect and darkening of the scene since less light is hitting the sensor of the camera, as one travels deeper into the focal stack. figure 3.1.4 shows the impact on brightness by comparing two colors from the same location in two different stack images. This was corrected by using Matlab's histogram matching function in order to normalize the intensity of each image to the first photo in the stack.



Stack Image 5 Stack Image 14

Figure 3.1.4. Color Variation in Focal Stack

The magnification change through the focal stack was estimated using equation 3.1.1.

$$M = \frac{f}{d-f} \tag{3.1.1}$$

where f is the focal length and d is the distance of the object from the camera, since the object positions aren't moving, the magnification could be calculated based on the input range of focus set in Open Camera, and the number of images taken. Since the change in aperture impacts the radial positions of the subject, each image in the stack was resized based on its magnification difference compared to the first image in the stack [12].

Once the images are aligned and color corrected, the amount of blur, or inversely, the amount of focus needed to be quantified for each pixel, in each image in the focal stack. Since blur generally has a low-pass filtering effect on an image, a summation of the high frequencies within a neighborhood of the pixel in question was found. The frequency information was calculated by convolving the image with an edge detecting kernel defined in equation 3.1.2.

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.1.2}$$

This is a discrete approximation of a laplacian which computes the second order derivative of the x and y directions. The output of the convolution was squared and then summed over a region of neighboring pixels (equation 3.1.3) [13].

$$B(x,y) = \sum_{j=y-n}^{y+n} \sum_{i=x-n}^{x+n} (I(i,j) * K)^2 \quad (3.1.3)$$

Larger neighborhoods can be used to pick up frequencies on smoother textures but causes a loss in the edge accuracy of the depth information. Figure 3.1.5 shows the results of the focus quantity of the 5th image in the stack with a 3 pixel neighborhood.

Figure 3.1.5. Focus Quantity: Stack Image 5/15

In order to use the focus information to quantify depth, a matrix was generated where each location corresponds to the position in the focal stack where the corresponding pixel's focus quantity is the largest, thus giving a relative location of the distance of that region. When normalized between 0 and 1 a black and white depth map can be produced shown as shown in figure 3.1.6

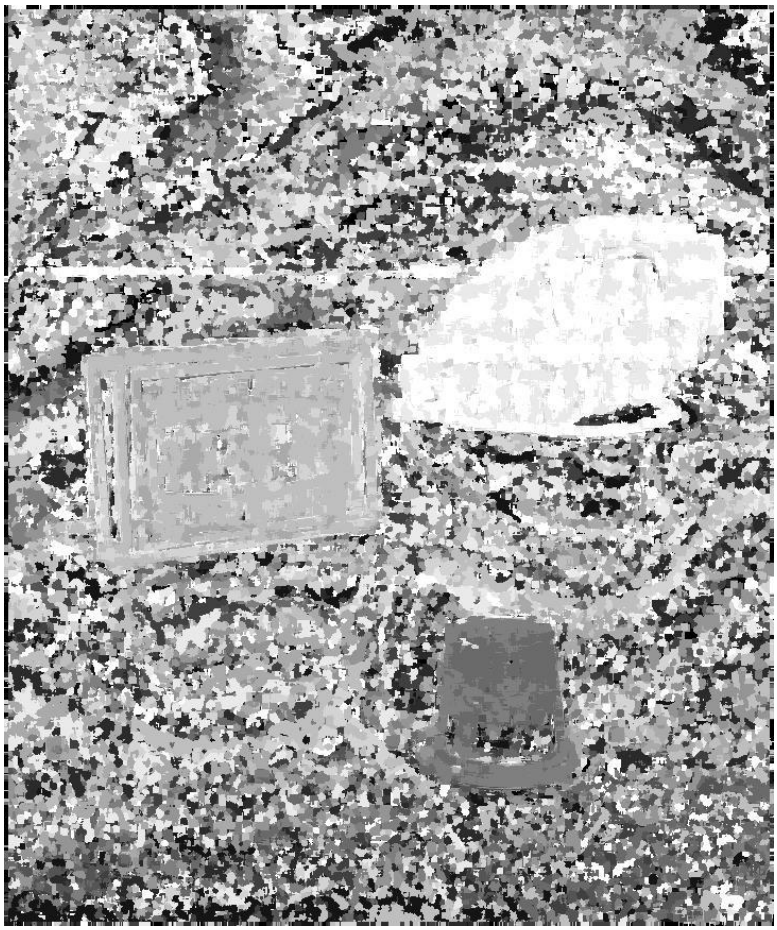


Figure 3.1.6. Object Depth Map

3.1.2 Portrait Mode Generation

Once the depth information is obtained and stored in a depth map, the depth map serves as a visual for relative depth in the image but also points to the stack location of focus quantity maximums. Using the depth map, the color information can be pulled from the focal stack where each pixel is most in focus and combined together. This creates an “all in focus” image (Figure 3.1.7), where no object sacrifices sharpness based on position.

Figure 3.1.7. “All in Focus” Image

Finally using the all in focus image, the edge detection convolution was repeated in order to create a binary image, mapping the location of the objects in the scene. Matlab functions were used to dilute and erode the edges until the full region of space each object existed in was mapped. The results of this manual process are shown in figure 3.1.7.

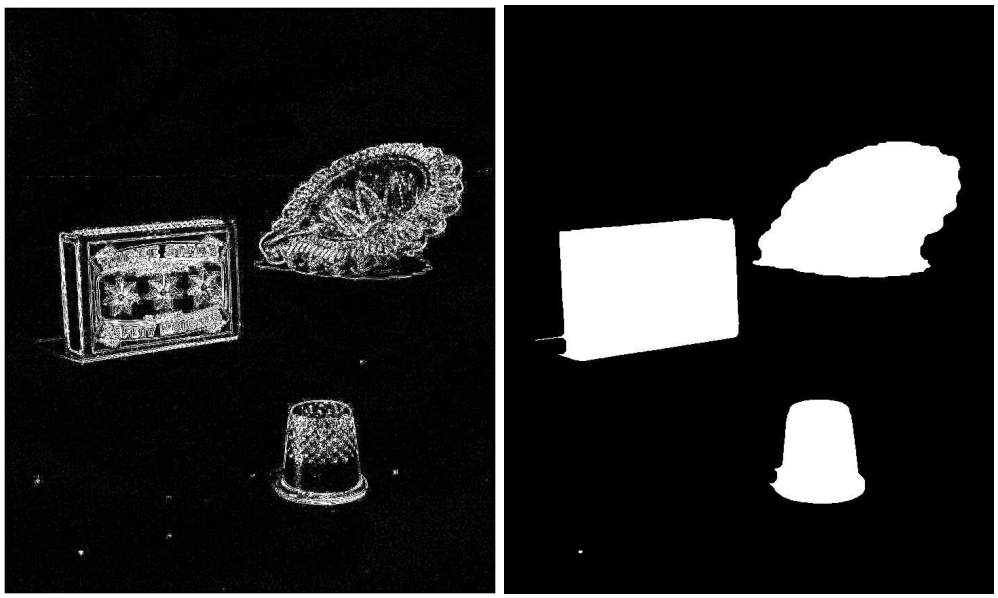


Figure 3.1.8. Binary Object Edges and Binary Object Map

Now that information pertaining to object depth as well as object location within the scene are known, the depth map and object map can be cross-referenced to return the final deliverable. Using a desired distance range, an object or group of objects could be identified. A gaussian blur was applied to the "all in focus image" and the pixels that fall within the distance range desired, and object map concurrently, would remain in focus. This produces a final "portrait mode" image that can contain more than one focal length as seen in section 5.1

3.2 Privacy Filter

With the extreme and growing prevalence of facial recognition software employed by large corporations with the intent of hyper-targeted marketing, government agencies for criminal identification and overall security, and even residential buildings with the intent to replace physical keys, one's identity is becoming increasingly public [8]. Resulting from this, depth-based computational imaging can be used to develop a facial recognition "jammer", applying a form of visual distortion, such as blurring, pixelation, and feature morphing, to an individual's face, preventing it from being detected entirely by facial recognition software. Our proposed solution to achieve facial recognition "jamming" is conducted via three distinct processes, demonstrated below in figure 3.2.1.

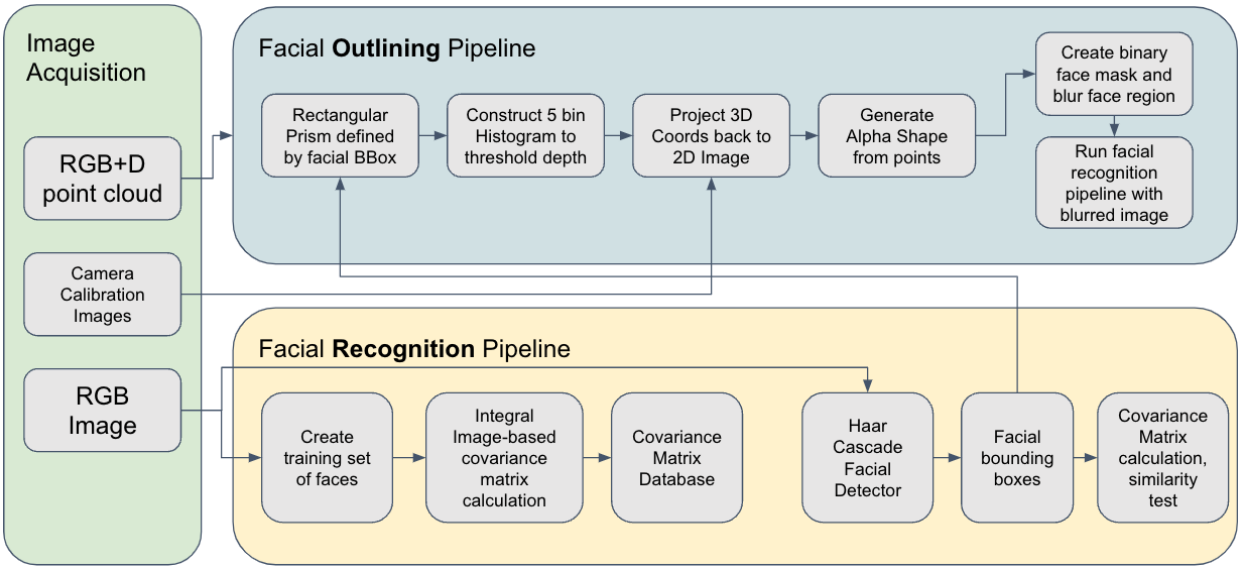


Figure 3.2.1: Proposed Privacy Filter Solution

3.2.1 Image Acquisition

In this critical stage, standard high-resolution RGB and RGB+D images are captured from an iPad Pro 2nd generation's main camera and LiDAR sensor respectively. In order for this privacy filter to be accurate and effective, a key assumption is made that corresponding RGB and RGB+D images are taken simultaneously, mitigating the possibility of image misalignment, dynamic changes in the environment such as illumination, object movement, etc., and the presence of additional noise. Unfortunately, due to the lack of availability of a specialized application to perform the aforementioned simultaneous capture, this assumption does not hold very well in practice as even slight disturbances will have a drastic impact on the output quality. To capture high-quality images, the following constraints were imposed on image acquisition: presence of only one subject in-frame, sufficiently illuminated environments, no obstructions present in the line-of-sight from the camera to the person's face, and no excessive deviation from a straight-facing pose to ensure the remaining two processes would function well. Examples of captured images can be seen below in Figure 3.2.2

Figure 3.2.2: Image Acquisition Examples

3.2.2 Facial Recognition Pipeline

In order to verify a facial recognition jammer's effectiveness, the ability to facially recognize a person is just as critical as image acquisition. For this task, the choice of using a region-based covariance matrix matcher was employed [14]. This method was chosen purely because of its mathematical foundation, allowing for its quick and verifiable implementation, as well as foregoing a potentially arduous and time-consuming training process under a deep learning-based approach. To recognize faces in a query image, a database of faces that comprises the searchable set of faces must be constructed. Constructing this face database involved using the high-resolution RGB images in the image acquisition step as training data, manually cutting out faces using a standard clipping tool (Figure 3.2.3).

Figure 3.2.3: Facial Database

Once these training images are obtained, they can be transformed into covariance matrices that represent them, enabling comparison amongst other covariance matrix representations of query

faces. To do this, an image is first converted into a higher-dimensional tensor through a mapping function.

$$F(x,y) = \left[x \quad y \quad R(x,y) \quad G(x,y) \quad B(x,y) \quad \left|\frac{\partial I(x,y)}{\partial x}\right| \quad \left|\frac{\partial I(x,y)}{\partial y}\right| \quad \left|\frac{\partial^2 I(x,y)}{\partial x^2}\right| \quad \left|\frac{\partial^2 I(x,y)}{\partial y^2}\right|\right]^T$$

With this higher-dimensional tensor of the form W x H x 9 from W x H x 3, we can calculate the covariance matrix of a region that is a subset of $F(x,y),\ R \subset F$ using integral images. The covariance matrix to be used takes the form:

$$C_R(i,j) = \frac{1}{n-1}\left[\sum_{k=1}^{n} z_k(i)z_k(j) - \frac{1}{n}\sum_{k=1}^{n} z_k(i)\sum_{k=1}^{n} z_k(j)\right]$$

where $z_k(i)$ is a 9-dimensional feature point inside R, which can be constructed using the first and second order integral images:

$$P(x', y',\ i) = \sum_{x \le x',\, y \le y'} F(x,y,i) \qquad i = 1 \ldots 9$$

$$Q(x', y',\ i,\ j) = \sum_{x \le x',\, y \le y'} F(x,y,i)F(x,y,j) \qquad i,j = 1 \ldots 9$$

Using these equations, the covariance matrix of any region can be computed very efficiently as these integral images can be computed in a single pass over the image. The final covariance matrix can be calculated using the following equation:

$$\mathbf{C}_{R(x',y';x'',y'')} = \frac{1}{n-1}\Big[\mathbf{Q}_{x'',y''} + \mathbf{Q}_{x',y'} - \mathbf{Q}_{x'',y'} - \mathbf{Q}_{x',y''} - \frac{1}{n}\left(\mathbf{p}_{x'',y''} + \mathbf{p}_{x',y'} - \mathbf{p}_{x',y''} - \mathbf{p}_{x'',y'}\right)\left(\mathbf{p}_{x'',y''} + \mathbf{p}_{x',y'} - \mathbf{p}_{x',y''} - \mathbf{p}_{x'',y'}\right)^T\Big]$$

where (x', y') is the top left corner of the region and (x'', y'') is the bottom right corner of the region. These covariance matrices are then stored to be used as a facial recognition database.

To extract a query image of a face, the Viola-Jones facial detection algorithm was used to extract a bounding box around the detected face (Figure 3.2.4). This box was then converted into its covariance matrix representation using the integral images process just described. With a covariance matrix database and the newly computed one, a similarity test is required to ascertain who the face represents. Covariance matrices do not lie in Euclidean space because they are not

closed under multiplication by a negative scalar [16]. As such, they are converted to the Euclidean

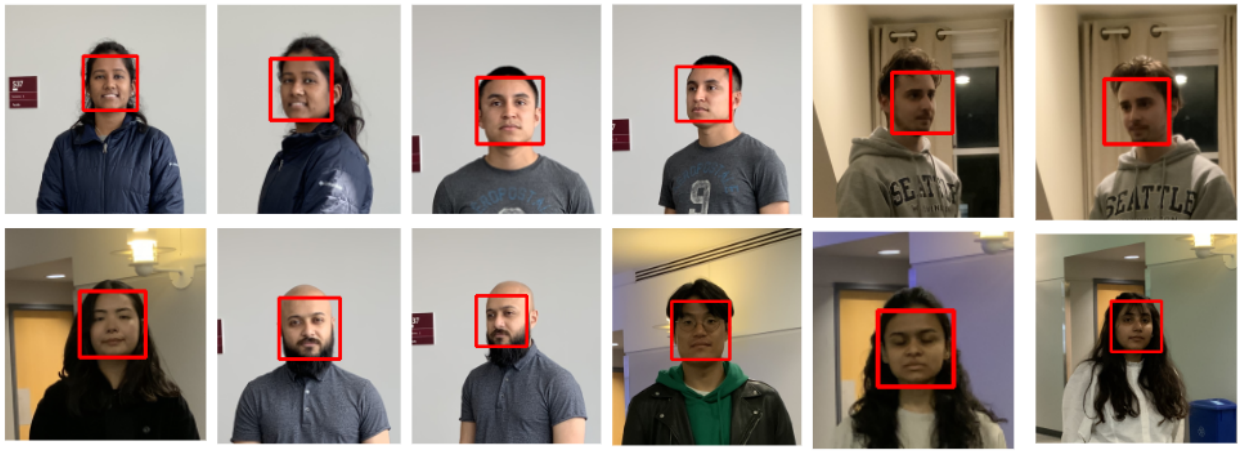


Figure 3.2.4: Facial detection bounding boxes from the Viola-Jones (Haar Cascade) Detector

space by first taking the singular value decomposition (SVD), obtaining three matrices V, D, $V^{\top}$. Using the D matrix, representing the eigenvalues of the original covariance matrix, the element-wise logarithm is taken, and matrix multiplication is performed to obtain the transformed covariance matrix. With these transformed matrices, the standard Euclidean distance can be taken to find the minimum distance between a face in the database and the queried face. Figure 3.2.5 demonstrates the output of facial recognition using this nearest neighbor approach.

Figure 3.2.5: Facial recognition using Nearest-Neighbor covariance matrix

3.2.3 Facial Outlining Pipeline

Incorporating the depth information into the facial recognition jammer, we begin with the output of the Viola-Jones (Haar Cascade) facial detection algorithm. The two-dimensional bounding boxes' coordinates are transferred into the RGB+D image under the assumption that it has the same resolution as the standard RGB image. This is a necessary assumption due to the actual

resolution of the RGB+D image standing at a mere 231x173 points versus the RGB's 4032x3024. With this assumption in place, the facial bounding box defines the face of a rectangular prism in three dimensions that is extended to the farthest depth in the image, encapsulating many points that include the face, hair, and most likely some background points (Figure 3.2.6). To achieve the initial goal of minimal face distortion from the future-applied Gaussian blur, we must remove the spurious points that constitute the background (Figure 3.2.7). To do this, a 5-bin histogram is constructed of the depth of each point, thresholding it at any point within the first two bins.

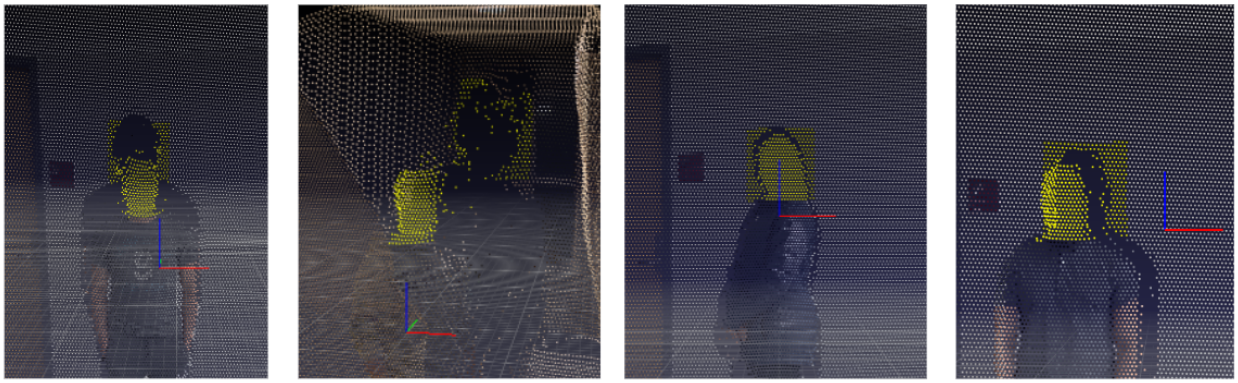


Figure 3.2.6: Facial detection bounding box translated into RGB+D image

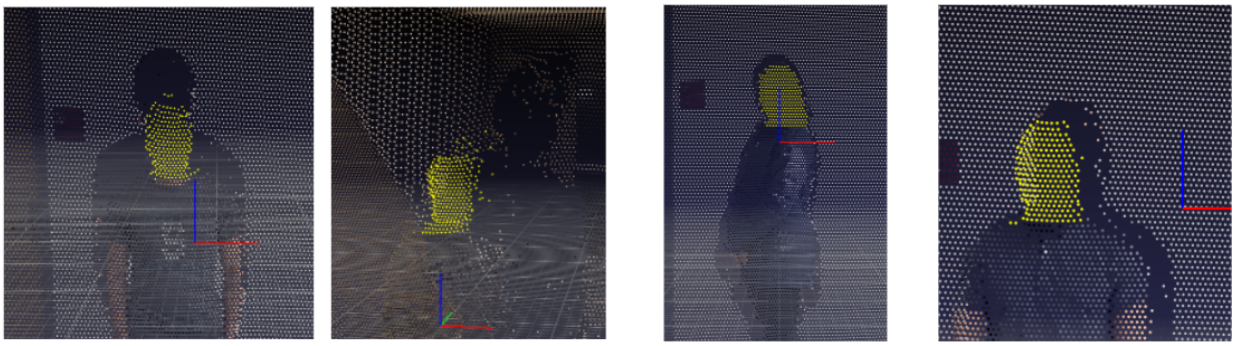


Figure 3.2.7: Removal of background points via 5-bin histogram thresholding

With thresholding complete, the coordinates of these 3D points must be converted into their respective 2D coordinates through the camera's projection matrix. Assuming the world coordinates are centered at the camera, with x and y axes corresponding to the image plane, and +z-axis along the outward direction through the optical axis, we use the camera's intrinsic matrix, K, to project these points back onto the image. To find K, a well-known chessboard-calibration algorithm was run to estimate the values of the focal length in the x and y directions, as well as the optical center's point in the image, known as the principal point. Using

the 2D distribution of these face pixels, a concave hull was calculated to find the minimum shell defined by the set of points, thus becoming the blurring facial mask of the person (see figure 3.2.8).

Figure 3.2.8: Concave hulls of the projected 2D points. Blurring facial masks.

From this point, the face mask would be overlaid onto the RGB image and applying Gaussian blur of differing sigma values to the region defined by the facial mask. This therefore minimally distorts the image to only blur the facial region.

4 Implementation

4.1 Portrait Mode

As described in section 3.1, the proposed portrait mode solution will utilize an original setup of objects that meet the scenery assumptions. The google store application “open camera” is used to obtain a focal stack of the scene, varying the focal length of the smartphone camera automatically without the need to touch it and introduce misalignment. All following processing steps are performed in MatLab using image processing functions such as imread, imfitl, im2gray, imfill, etc, in addition to original code that implements the described mathematical calculations and collects the functions into a linear sequence.

4.2 Privacy Filter

As described in section 3.2, the proposed privacy filter solution was implemented through a python Jupyter Notebook making extensive use of the popular libraries, Numpy, OpenCV, and SciPy. Numpy and SciPy allowed for the implementation of the region-based covariance matrix detector through Numpy’s matrix operations and SciPy’s SVD function. Numpy was also used for the Facial Outlining Pipeline, creating the depth-based histograms, as well as storing the

point cloud data retrieved. OpenCV was used to perform the iPad's camera calibration procedure, as well as facial detection and blurring. It is important to note that the region-based covariance detection code was found online, but altered from the research paper's original implementation [15].

5 Experimental Results

5.1 Portrait Mode

The implementation of section 4.1 was performed on one scene of objects, and two portrait mode images were produced. Figure 5.1.1 shows the portrait mode image of the mid-distance object, distance range between 0.4 and 0.6.

Figure 5.1.1. Portrait Mode "Mid Distance" Object

Figure 5.1.2 shows the portrait mode images of the closest and furthest distance objects, distance range less than 0.3 and greater than 0.7.

Figure 5.1.2. Portrait Mode “Short and Long Distance” Objects

These preliminary results successfully meet the goal of the project, to create multi-distance multi-subject portrait mode images using depth from defocus. The largest variation from the intended results has to do with the automation of the process. Since this image generation process requires object detection, the current object mapping procedure would require manual adjustment with any change in scenery, which is not ideal. Additionally in the final portrait mode images show some background particles around the edges of the objects in focus. An implementation of an alpha matte would treat that effect and produce higher-quality images.

5.2 Privacy Filter

With facial recognition in place, the experimental goal of the privacy filter was to determine how much distortion is required to throw off the output, incorrectly registering a person's identity. This was done in two stages: the first is the re-detection stage where it was tested if facial detection can first find a face given a predefined level of distortion. This stage is important because if there is no facial detection, there cannot be any facial recognition. Performing the re-detection stage involved finding faces in an image by applying the Viola-Jones detector defined in section 3.2. Once a face was found, the facial outlining pipeline was conducted, blurring the facial mask of the individual in the image. Unfortunately, due to the disparities in RGB and RGB+D resolutions, the facial mask was not able to align perfectly, thus becoming mostly unusable. To compensate for this, the entire bounding box detection was used to simulate what blurring and facial recognition would encounter. As mentioned earlier, a Gaussian blur was applied with a sigma value ranging from 1 to 5, thus increasing blur in the facial region as the sigma value increased. Once blur was applied to the facial region, the image was re-run through facial detection, testing to see if the face could still be detected. Figures 5.2.1 and 5.2.2 below demonstrate how facial detection is quite robust to heavy blur, having faces remain detected even with a sigma value of 5. It is important to note that only the images where the subject's face was oriented mostly straight to the camera would be detected in this manner, suggesting pose has a strong effect on this as well.

Figure 5.2.1: Sigma value of 1, faces still detected at every pose

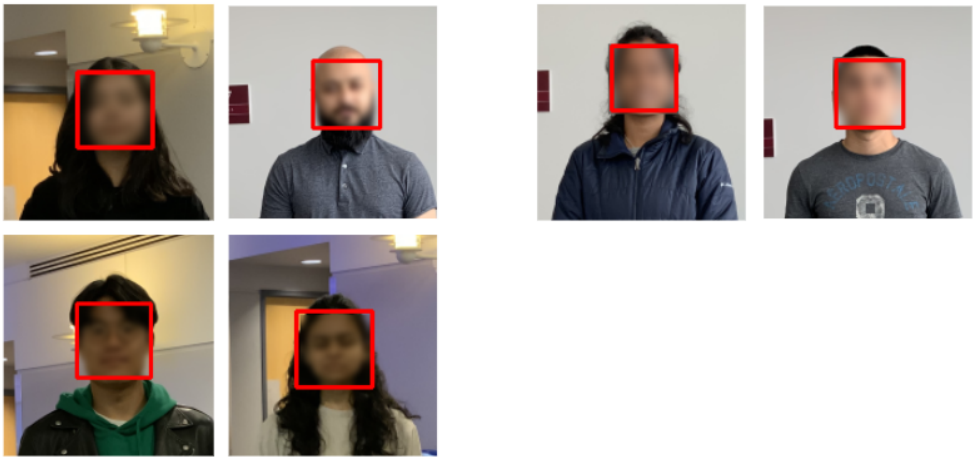


Figure 5.2.2: Left Group: Sigma value of 4. Right Group: Sigma value of 5

Once this re-detection stage was completed, the second stage of re-identification was performed where the blurred facial region images were sent through the Facial Recognition Pipeline. This test demonstrated (Figures 5.2.3 and 5.2.4) that facial recognition is extremely sensitive to distortion in the facial region, most likely due to its implementation details involving the covariance matrix of features such as the gradient magnitude and second derivative of intensity magnitude. These results are more or less consistent with the original goal of minimally distorting a face to prevent facial recognition from accurately identifying an individual. Despite the imbalance in resolution in the RGB and RGB+D image, the process proved successful and would have produced the results intended had the facial mask region aligned in the RGB image.

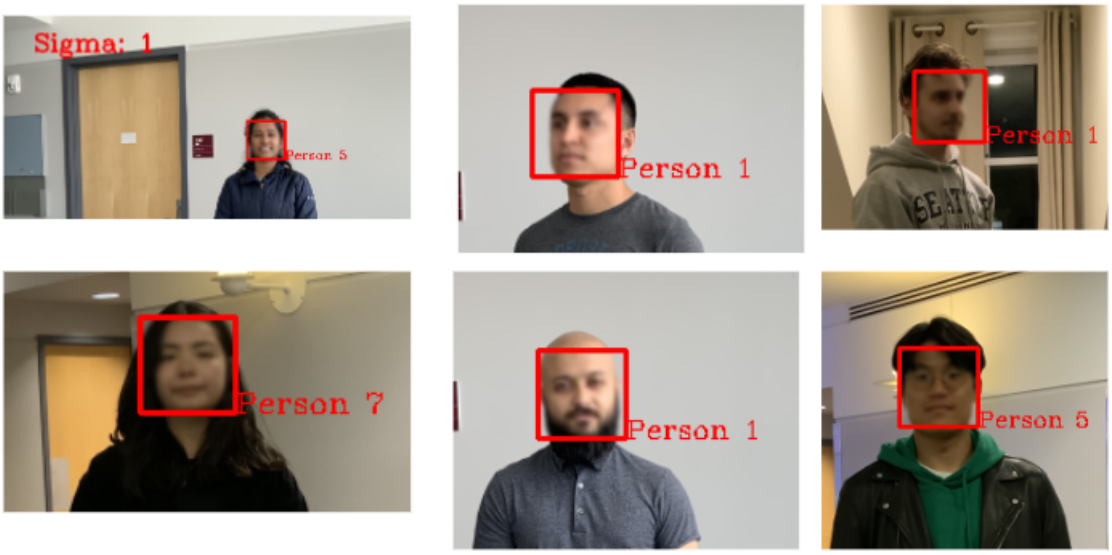


Figure 5.2.3: Facial recognition of blurred facial regions, sigma of 1.

Figure 5.2.4: Facial recognition of blurred facial regions, top row: sigma of 3. bottom row: sigma of 4

6 Conclusions and Possible Improvements

6.1 Portrait Mode

Since it is commonplace for today's smartphones to be able to obtain depth information, the process of using depth from defocus for this application would not likely be the most viable. Depth from defocus, and especially this methodology is highly restrictive to settings in which it is effective. Using smaller, highly textured objects that do not overlap in low-frequency settings is too specific for a wide use case, however, depth from defocus is still commonly used in microphotography so the practice is not obsolete. On the other hand, a smartphone portrait mode capability that could focus on subjects at various distances is something that could be implemented using existing technology. The biggest challenge would be object detection which most likely would require real-time machine learning to allow the user to select regions they want to focus on. Alternatively, there could be an open-loop setting that allows for an input of a desired focal length range, not unlike the ones called out in section 5.1. With respect to this project, the largest weakness is the lost information in the depth map due to low frequency regions. The best improvement that would allow this project to have a wider range of applications would be improvement of the depth map, one way to do this would be to linearly interpolate distance on top of the additional depth information from focus.

Additionally, the impact alignment and color correction had in the generation of a high quality output, was one of the most surprising outcomes of the process. Figure 6.1.1 shows the result of the all focused image without either of those processing steps.

Figure 6.1.1. All in Focus Image Without Alignment or Color Correction

Despite how digitized the average person's experience with photography is today, it is enlightening to appreciate the smartphone camera as a mechanical system and how that system still dictates the end image result.

6.2 Privacy Filter

Analyzing the results of the privacy filter experiment, it is evident that it is quite simple to cause facial recognition to claim an incorrect identity once a small level of distortion is applied to one's face. Again, this is most likely due to the implementation details of the facial recognition system being highly dependent on features that change drastically in the presence of a blurring filter like the gradient and second derivative of intensity. Some improvements that could be made to the privacy filter include obtaining a denser depth map as the lower resolution point cloud disallowed the facial mask from fitting over the original image's facial region. A further improvement would be a more robust facial recognition algorithm that is potentially deep-learning-based at the cost of higher training setup and time which could handle higher levels of distortion that are more likely to be implemented in real-world settings.

References

[1] F. Moreno-Noguer, P. N. Belhumeur, and S. K. Nayar. Active refocusing of images and videos. ACM Transactions On Graphics (TOG), 26(3):67, 2007.

[2] C. Castellanos, A. Nguyen. Artificial Refocusing of High Resolution SLR Images From Microsoft Kinect Depth Maps. 2017.

[3] Y. Zheng and C. Kambhamettu. Learning based digital matting. In Computer Vision, 2009 IEEE 12th International Conference on, pages 889–896. IEEE, 2009.

[4] S. Suwajanakorn, C. Hernandez, and S. M. Seitz. Depth from focus with your mobile phone. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3497–3506, June 2015.

[5] B. Wiberg, A Method for Refocusing Photos using Depth from Defocus. Stanford University. 2018.

[6] Encyclopædia Britannica, inc. (n.d.). Perfecting the medium, c. 1900–c. 1945. Encyclopædia Britannica. Retrieved March 11, 2022, from https://www.britannica.com/technology/photography/Perfecting-the-medium-c-1900-c-1945

[7] Cossairt, O., Gupta, M., & Nayar, S. K. (2012). When does computational imaging improve performance?. IEEE transactions on image processing, 22(2), 447-458.

[8] The New York Times. (2020, July 15). Facial recognition is everywhere. here's what we can do about it. The New York Times. Retrieved March 11, 2022, from https://www.nytimes.com/wirecutter/blog/how-facial-recognition-works/

[9] Introna, L., & Nissenbaum, H. (2010). Facial recognition technology a survey of policy and implementation issues.

[10] Scherhag, U., Rathgeb, C., Merkle, J., Breithaupt, R., & Busch, C. (2019). Face recognition systems under morphing attacks: A survey. IEEE Access, 7, 23012-23026.

[11] Pulfer, E. M. (2019). Different Approaches to Blurring Digital Images and Their Effect on Facial Detection.

[12] Magnification. kielia.de – Maritimes Leben. (n.d.). Retrieved April 19, 2022, from https://www.kielia.de/photography/calculator/magnification/

[13] Nayar, S. K. (1989). Shape from focus. https://doi.org/10.21236/ada215959

[14] Tuzel, Oncel, Fatih Porikli, and Peter Meer. "Region covariance: A fast descriptor for detection and classification." European conference on computer vision. Springer, Berlin, Heidelberg, 2006.

[15] “Region Covariance: Covariance Matrix and Computer Vision.” Linuxtut.com, 9 May 2020, https://linuxtut.com/en/8e61da17b11011ddb661/.

[16] Dryden, Ian L., Alexey Koloydenko, and Diwei Zhou. "Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging." The Annals of Applied Statistics 3.3 (2009): 1102-1123.