Diminished Reality Based on 3D-Scanning

Erwin Andre, Helmut Hlavacs

Diminished Reality based on 3D-Scanning

Erwin Andre and Helmut Hlavacs

University of Vienna, Austria
Faculty of Computer Science, Entertainment Computing Research Group
helmut.hlavacs@univie.ac.at

Abstract In this paper a new method for diminished reality is explored, which uses a 3D model of a room to fill in missing regions when removing objects in real time on a video. The room is scanned before, so it is possible to recreate missing pieces in all their detail. The proposed method allows freedom in movement and rotation and is demonstrated with an application on a mobile device, detailed results are shown.

1 Introduction

Diminished reality (DR) aims at *removing* visual objects (a.k.a. *targets*) from our field of view. DR is thus considered to be the opposite of Augmented Reality (AR) [8], since AR aims at *adding* visual objects to our field of view. There are two common methods for removing targets from videos or images: the first one fills in the missing regions by using readily available background images or essential information captured beforehand, the other one uses the information around the target or the similarity of textures to artificially produce the filling content (usually using some deep learning technique), an approach also known as image inpainting.

Both methods have their limitations. For the first approach, it might be infeasible to capture the visual information used for filling in image regions occluded by the target, for instance in live video situations. In the inpainting approach, it may not be possible to correctly guess the occluded image information from the surrounding pixels, e.g. if the background contains many details.

In this paper a new method for diminished reality is explored by using 3D scans as stored information to remove targets from *live* video feeds produced by the camera of a mobile device. 3D scanning analyzes objects or whole environments (in this paper: whole rooms) by capturing depth data to reproduce the structure of objects and color data to reproduce their appearance. 3D models can be created by mapping these data correctly. As a consequence, the camera producing the video feed where targets are removed may move freely in 3D space, and also chose its orientation freely (resulting in 6 degrees of freedom), because the 3D room scan can be rendered from arbitrary positions and angles. Currently none of the existing DR techniques for live video streams from mobile devices allows this much freedom in movement and rotation during the diminishing process.

2 Related Work

Diminished reality and image inpainting have been topics of research over the past few years and there are already many different approaches [10]. Most are not used in live video feeds, but show how to remove targets from images or videos off-line.

Kawai et al. [7] discuss different approaches on removing targets, both inpainting and diminished reality methods. They also present their own method, which is a mix of diminished reality and inpainting. In their method they have to manually select a visual target and then use SLAM (Simultaneous Localization and Mapping) to track feature points. SLAM is also used in the work presented in this paper. Kawai et al. start with initializing the camera pose and feature points. Afterwards a target region is selected manually by the user and the image is analyzed and divided into multiple images to improve the inpainting quality.

Hackl et al. [3] explore a method for removing targets from live video feeds and show their results with an android application. They use previously captured images to remove objects, images are stored in a large data structure called the frame store. However, the authors only achieve change or orientation, a translation of the mobile device is not allowed.

Yagi et al. [14] present a method on how to remove pedestrians from videos taken with a hand-held camera. This method can only be executed after the video was taken and is not supposed to remove pedestrians in real time. First, the pedestrians get detected and marked. After that a 3D model of the background is created from the video frames with SfM (Structure from Motion) to replace objects later on, by projected 2D frames from this model. Afterwards the camera positions are estimated and frames from the model are extracted. Finally, the original and projected images are blended together and frames with removed objects (pedestrians) are generated. It has to be mentioned that the color of a hidden pedestrian is different from the rest of the image.

Broll et al. [5] propose a method which removes objects from a live video stream. This approach consists of two main tasks: object selection and tracking as well as image completion. Objects are manually selected by drawing a contour e.g. with a mouse on a laptop, then a mask is generated. Image completion uses a patch based inpainting algorithm to fill in the selected area. The quality of such a method is reliant on the input and the area around the target that will be removed. This was one of the first applications on a mobile device which was able to remove targets. The removal quality is not affected by the area around the target, which is the same in this paper's implementation.

All these different methods remove targets from videos or images, but only a few are able to perform this in real time. It is even more difficult to run diminished reality applications directly on mobile devices, like [11], who proposed a similar app like ours, but rely on the AR toolkit Vuforia.

3 The Structure Sensor

3D room scans can be created with various devices but for our work it was required to run the application on a mobile device. The recently launched Google ARCore does not have the capability to create 3D room scans, so the only feasible possibilities left were Google Project Tango and the Structure Sensor (cf. Figure 1).

Figure 1: The Structure Sensor mounted to an iPad

At first Google's Project Tango looked promising, but it did not match the scanning quality of the Structure Sensor [6], this also might be due to the fact that the Lenovo Phab 2 Pro that we used is already 3 years old. Furthermore Tango has been discontinued by Google, and so is no longer an option.

When comparing scans from both sensors (cf. Figure 2) there are some significant differences. The textures of the scan with the Tango device are blurry and rendered images from this mesh would be a bad replacement for removed targets. When comparing both images, the scan with the Structure Sensor looks much better since it does a very good job on capturing plane surfaces. On complex structured surfaces, however, there are some white holes, meaning that the sensor did not capture textures there. The holes, though, can significantly be reduced with better lighting. Overall, the above comparison favours the use of the Structure Sensor for the task at hand.

In more detail, the Structure Sensor is a commercial depth sensor designed to work with iOS devices, e.g. an iPad or iPhone, but there are possibilities to run the sensor on other devices as well (with OpenNI). The device consists of a NIR laser-projector and a NIR camera [9] which work together with the native camera of an iOS device to add 3D vision. The recommended range for the sensor is between 0.4 and 3.5 m, longer ranges are possible but are hurting the sensor's accuracy. At a range of 0.4 m the accuracy is within 0.5 mm and at a range of 3 m within 30 mm. The sensor features two resolutions for capturing depth data, either 640×480 or 320×240. For our implementation the resolution of 320×240 is used.

(a) Lenovo Phab 2 Pro (Project Tango) (b) iPad 6th Gen. (Structure Sensor)

Figure 2: Comparison of a Google Project Tango room scan and a scan with the Structure Sensor

4 The Proposed Method

The main idea of our method is to use the 3D scan of a room to create a 3D model and remove real world targets by filling in missing regions with projected images from the 3D model. The quality of this method depends heavily on the quality of the 3D scan which depends on lighting and how steady the device is moved while scanning. For a good scan a well-lighted room and no harsh movement during the scan are required. The pipeline for this diminished reality approach consists out of five core parts:

1. **3D Scanning:** The first step to diminish objects is to create a 3D model of the room with 3D scanning enabled by the Structure Sensor. It is not necessary to scan a whole room, it is possible only to scan parts of a room.
2. **Positional Tracking:** For each synchronized depth and color frame output by the sensor the position are updated with the SLAM engine of the Sensor's Structure SDK.
3. **Object Detection:** Detection of objects which will be removed later on with either color or face detection.
4. **3D to 2D Projection:** This is used to render a 2D image from the 3D room model at the current camera's position and orientation, which is required for replacing the detected target.
5. **Stitching:** The current camera image is blended together with the extracted 2D image from the 3D model.

The above points are discussed in the following.

4.1 3D Scanning

Before any target can be removed the first step is 3D scanning the room and creating a 3D room model. A full room scan is not needed but possible. Objects will be removed only if that area was scanned before. Just before the scanning is started, the camera position is initialized. After that, the sensor data and camera data are used to store point clouds and keyframes during the scanning process. While calculating the 3D mesh the sensors need to be stopped, which only takes a few seconds. Before the sensors are stopped, the current position is saved to later initialize the tracker with this position, so the camera and the mesh are matched. The device should not be moved during the rendering. Unfortunately basic hand movement cannot be prevented 100 %, but since the projected image gets aligned with the camera image later on, this does not influence the overall quality too much. The triangle-mesh is calculated with the depth data and colorized with the color data stored in the keyframes. For the 3D scanning the SLAM engine of the sensor's Structure SDK is used, which includes 3D mapping, tracking and scanning features. For communicating with the sensor, the SDK's sensor controller is used, which allows to start and stop the sensor and to get the status of the sensor.

4.2 Positional Tracking

The positional tracking is always active if the sensor is running, since the projected image should have the same viewpoint as the iPad's camera, in order to match the current camera's view as closely as possible when replacing targets. The position is updated with each synchronized depth and color frame output by the camera of the iPad and the Structure Sensor. This process also uses the SLAM engine and is driven by the SDK.

4.3 Object Detection

For each frame output by the video camera, either color detection or face detection are applied to the image. In our app, the detection type can be chosen by the user. Multiple objects can be detected and removed at once and the shape and size of the objects can vary. A binary mask is generated during that process which is needed to remove the targets. In such a mask, targets that should be removed are shown with white pixels, while the rest of the mask are black pixels, meaning they do not get removed. When using face detection, if a face is detected, a circle is drawn around the found face and filled with white pixels In case of color detection, if the object color is detected it will be replaced by white pixels.

4.4 3D to 2D Projection

To extract the image from the mesh two matrices are needed: the camera extrinsic parameters (e.g. translation, rotation), which describe the coordinate system

transformations from 3D world coordinates to 3D camera coordinates and the camera intrinsic parameters which represent the characteristics of the iPad's camera (e.g. focal length, image sensor format) [13,4]. Both matrices are needed to render the image from the same position as the camera. The projection image is the same size as the camera image which is important to stitch them together later. The 3D to 2D projection is described by the following equations [6]:

$$P^2 = P^3 \times M \tag{1}$$

P^2 being the 2D image and P^3 being the 3D model which is transformed by M, the perspective projection matrix.

$$M = K \times R\,(I - C) \tag{2}$$

M is defined by the camera extrinsics K, the camera intrinsics R and the coordinate matrix of the camera origin C in the world coordinate frame (I is the identity matrix).

4.5 Stitching

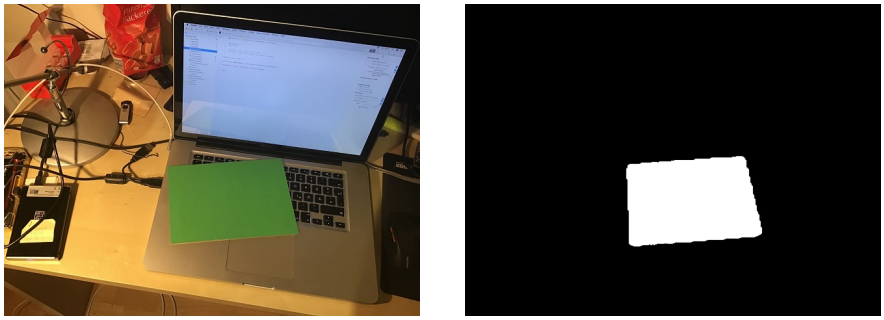
Due to inaccuracies in the tracking process, after a target is detected and the frame is extracted from the mesh, the camera image and the mesh image need to be aligned. This is done in order to blend both images together and make them look like one image. The alignment uses feature detection to find matching feature points, then a transformation matrix is calculated from the matches. After the projected image is transformed both are stitched together using the binary mask which is created during the object detection. One single image is created with the target object being removed.

4.6 Implementation

Since the Structure Sensor is designed to work mainly with iOS, we developed an iOS app using Xcode. The app is written in objective C and C++, the latter was needed to use the OpenCV library [1]. OpenCV 3.4.4 is used to manipulate the images from the camera and generate a binary mask to remove targets. OpenGL is used together with the Structure SDK to render the 3D mesh, render the color image from the camera to the view or to render an image from a specific position from the 3D model.

For detecting colors, color keying is used by the inRange method provided by OpenCV. Therefore, the image needs to be converted from RGB to HSV, because it is hard to define a range of colors in the RGB color space. In the HSV color space it is easier to define a range of similar looking colors. The inRange method specifies a color range between two HSV scalars. After the inRange method has been used, the image still has some white and black holes which are unwanted. That is why multiple morphological transformations are applied to the image. Before applying morphological transformations to the image it is downscaled to

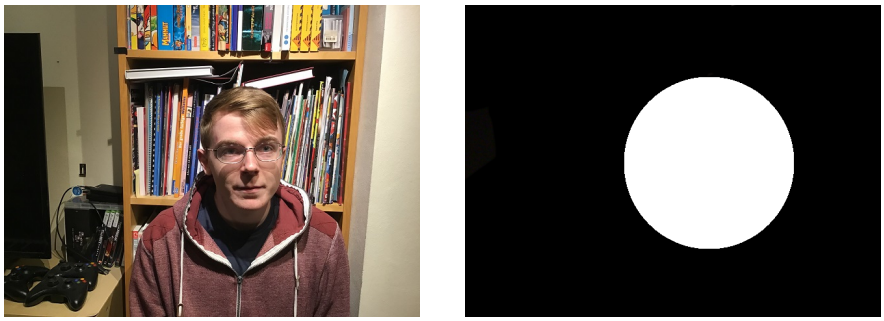
half the size which speeds up the mask generation tremendously (being about 70 % faster than without the downscaling), after all the transformations the image is upscaled again. A binary mask is created which is used later during the stitching process.



(a) input image (b) binary mask

Figure 3: mask creation from green colored objects

For detecting faces OpenCVs Haar feature-based Cascade Classifiers [12] are used. There is another option available with LBP-based Cascade Classifiers but the Haar feature-based Cascade Classifiers are usually more accurate. Before the detectMultiScale method is applied to the image, it has to be converted to a greyscale image. The detected objects are marked with a rectangle.



(a) input image (b) binary mask

Figure 4: mask creation from a detected face

4.7 User Interface

The User Interface is kept very simple, so it is easy to use and understand. Anybody should be able to use the application after watching a demonstration video of the app. There are no complex settings or preferences implemented which the user can choose from. After starting the application, a short information message is shown to the user welcoming him and explaining what he can do with this app. Before the scanning is started, the range of the scan can be adjusted using the slider located at the bottom of the screen. By default it is set to 5 m which is already a long range considering the sensor works best between 0.4 and 3.5 m. A range between 0.5 and 10 m can be selected. The scanning range makes sure that no objects further away than the selected range can be scanned.

5 Evaluation and Discussion

In this section the results and the quality of the target removal of our application are evaluated. We provide a comparison to inpainting and a case study on the iPad in this section.

5.1 Removal Quality

The quality of the application's target removal capabilities seem good when compared to other solutions like inpainting or the application of Hackl et al. [3], but there are some factors which determine the quality of the diminished images. Since room scanning is very instrumental for this method, the quality of a room scan is also important to the whole method. For a good room scan the user has to stand on one position and scan the whole room from this position, also the handling of the device is important since harsh or too fast movements result in a bad scan. Blurry scans or cuts in the textures of the scan tremendously reduce the effectiveness of the feature matching algorithm used to align both images and results in stuttering of the diminished output video. That is why a steady hand movement can make a huge difference although it has to be mentioned that the sensor is very fast and when something unexpected happens the scan can be restarted easily.

Another factor for the quality of the removal is the light, there needs to be good lighting in order for the application to achieve optimal results. Faint light makes the color detection very hard because under different light the HSV values do not match anymore and for instance no green color may be detected.

Lastly the structure of the scanned environment is also important to the generated 3D model and influences the removal quality. The more complex the structure of the environment is, the more clunky the scanned 3D model seems to be, this might be caused by the fact that a lot of data has to be processed at once. Object scans are a lot more accurate in comparison to a room scan because only a single object is being scanned.

In Figure 5b, problems of the Structure Sensor can be seen, caused by scanning complex surfaces. The sensor has difficulties creating an accurate 3D model.

At the top of the replaced texture the book seems to be curved a little bit, although it should be straight if compared with the input image (cf. Figure 5a). In general the textures at the top do not match properly, but that is because of the inaccurate scan.

As mentioned before scanning should be done from only one position, but it is possible to change position during the diminishing process and still receive accurate textures although the angle has changed. Of course textures which are hidden during the scanning will not be there during the diminishing process. Changing positions is made possible thanks to the 3D model which can be rendered from different view points. Obviously this is also having limitations on giving good results, there are around 45 degrees in each direction from the originally scanned position where the rendering of the transformed viewpoint works well. That means there are at least 90 degrees overall where the object removal works good.

5.2 Comparison to Inpainting

In this section the results of the proposed method are compared to inpainting, providing the input image and the results of both methods. For this comparison the inpainting functionality of Affinity Photo is being used which is similar to the inpainting from Adobe Photoshop.

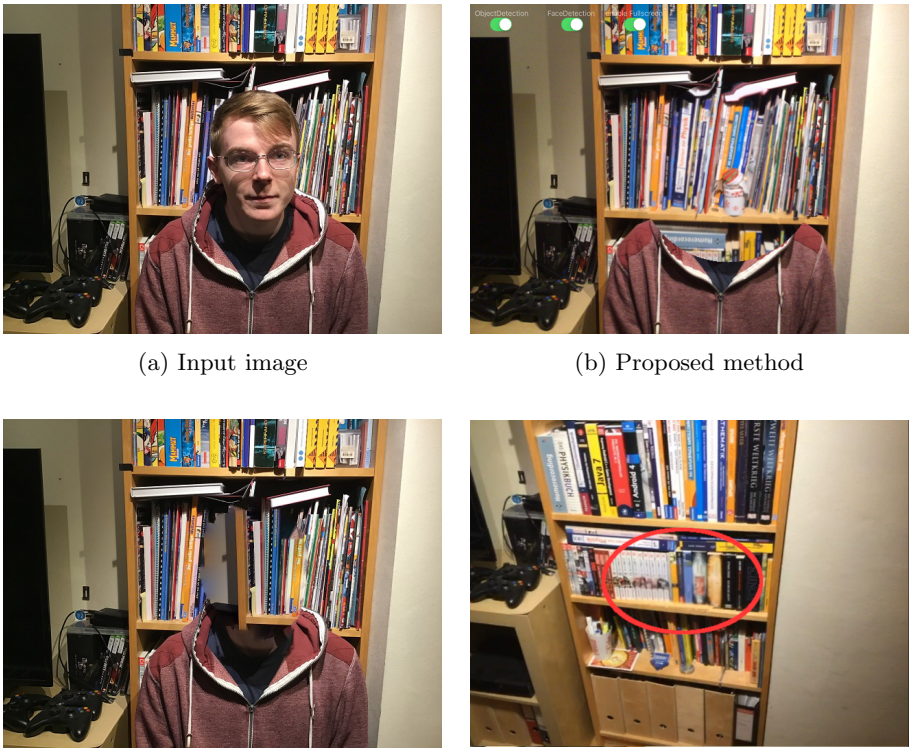
When both outputs of the images from the face detection are compared, inpainting is not able to reproduce the textures hidden by the face and a wrongly inpainted image is created (cf. Figure 5c). The proposed method however is able to fill in the missing regions quite accurately (cf. Figure 5b).

5.3 Case Study

The purpose of this case study was to find out if people recognize that there are two images stitched together in the diminished view. For someone who knows the application spotting some stuttering in the replaced region is easy although it is difficult to be seen. The research question was: Does someone who does not know what the app does notice the stuttering caused by minimal inaccuracies in the filled in region?

We captured a short video sequence (approx. one minute) using color detection and moving around the iPad. Minimal stuttering in the diminished view can be noticed. The video was shown to the participants telling them to mention everything they notice during the video immediately. 10 people participated, saw the video, and should mention it the second they saw something out of the ordinary.

When looking at the table (cf. Table 1) it is interesting that 5 people did not notice the filled in region at all, which means 50 % did not notice it. However from this group all are older than 46 years and 2 participants are in their mid 70s. To some extent not noticing any stuttering might be caused by visual perception, which worsenes when getting older [2]. Three people could spot the diminishing during the first 10 seconds of the video. Surprisingly one person was able to spot



(a) Input image

(b) Proposed method

(c) Affinity Photo Inpainting

(d) Stuttering

Figure 5: Comparing Face Detection to Inpainting, and stuttering

the stuttering after three seconds. It can be said that 70 % of the participants did not spot the diminishing in the first 40 seconds of watching the video. Between 37 to 40 seconds there is some stuttering in the video, which can be seen in the images (cf. Figure 5d) but it is easier to notice when watching the video because it looks like something is moving. It is also noticeable that there is a trend that people with less visual acuity spot the difference faster than people with higher visual acuity. It must also be mentioned that most participants were really concentrated, showing higher attentiveness to small video artefacts than usual viewers might show.

In summary it can be said that it seems to be difficult to detect the stuttering artefacts when watching the video for the first time. However, highly concentrated and focused viewers can identify it to some extent.

During 3D scanning the app runs with 30 fps, when targets are removed the fps drop, depending on the target detection method used. For color detection, the performance is between 10 and 12 fps which is not completely smooth, but good enough to use the application to demonstrate this method. When using face detection, between 6 and 7 fps are possible. When comparing the application to

age	sex	visual acuity (in diopters)	Time to recognized (s)
24	male	0	10
27	male	0.75	40
27	female	0	6
39	female	0	41
46	male	6	-
49	male	0.75	3
51	male	0	-
52	female	3.5	-
74	female	3	-
77	male	2.5	-

Table 1: Case study data

the one developed during the work of Hackl et al. [3], similar speed is achieved with this prototype.

The calculation of the 3D model takes around 0.5 to 3 s depending on how much was being scanned. This is very fast when comparing to the Lenovo Phab 2 Pro were the calculations took up to a minute, but this device is also 3 years older than the iPad.

6 Conclusion and Future Work

During this project a method was explored, which uses 3D scanning for diminished reality. Our application is able to remove target objects from live video feeds recorded on a mobile device by using a previously created 3D model of the environment to replace them. Diminished reality is a field where already some methods have been published over the last years, but no method so far has utilized 3D scanning. The proposed method delivers good results and compares favourably when compared to existing diminished reality methods, especially due to its 6 DoF for the mobile device, and the real time use case. Furthermore, the case study shows that the quality of the replaced image is very good and a difference between the real image and the diminished image is hard to spot.

Though our results are promising, there is still room for improvement regarding the application's performance. Currently all the image manipulations are calculated by the CPU which slows down the application a lot. If OpenCL would be available for iOS, parallelization could be used to make this application much faster. Maybe on other platforms it is easier to use the GPU. Although the performance is the biggest weakness of the prototype, the application can still be used to demonstrate this method.

The next step to improve this diminished reality method could be to use surface or pointcloud matching. Surface matching matches 3D data, e.g. depth data from the sensor and the 3D mesh. No more alignment would be needed, only the mask needs to be applied to the images. The tracking is not needed too,

incoming depth data will be matched with the 3D mesh. Also storing the last position during the calculation will not be necessary. Surface matching would be hard to do if implemented with the Structure SDK because the pointclouds from the depth frames cannot be accessed easily. Some conversion methods would be needed in order to use surface matching algorithms. OpenCV for example provides a surface matching algorithm.

References

1. OpenCV library. `https://opencv.org/`. Accessed 2019-01-09.
2. Faubert, J., Dixon, P. e., and Bennett, P. J. e. Visual perception and aging. *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie exprimentale, 2002, Vol.56(3), pp.164-176*.
3. Hackl, A., and Hlavacs, H. Diminishing reality. In *International Conference on Entertainment Computing (IFIP ICEC 2018)* (2018).
4. Hartley, Richard, a. V., and Zisserman, Andrew, a. *Multiple view geometry in computer vision /*, 2nd ed. ed. Cambridge University Press,, Cambridge, UK ; New York :, 2004.
5. Herling, J., and Broll, W. Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments. *2010 IEEE International Symposium on Mixed and Augmented Reality, Oct. 2010, pp.207-212*.
6. Kalantari, M., and Nechifor, M. Accuracy and utility of the structure sensor for collecting 3d indoor information. *Geo-spatial Information Science, 02 July 2016, Vol.19(3), p.202-209*.
7. Kawai, N., Sato, T., and Yokoya, N. Diminished reality based on image inpainting considering background geometry. *IEEE Transactions on Visualization and Computer Graphics, March 2016, Vol.22(3), pp.1236-1247*.
8. Kawai, N., Sato, T., and Yokoya, N. From image inpainting to diminished reality. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2014, Vol.8525(1), Part 1, pp.363-374*.
9. Kersten, T. P., Przybilla, H.-J., Lindstaedt, M., Tschirschwitz, F., and Misgaiski-Hass, M. Comparative geometrical investigations of hand-held scanning systems. *The International Archives of the Photogrammetry, 01 June 2016, pp.507-514*.
10. Mori, S., Ikeda, S., and Saito, H. A survey of diminished reality: Techniques for visually concealing, eliminating, and seeing through real objects. *IPSJ Transactions on Computer Vision and Applications, 2017, Vol.9(1), pp.1-14*.
11. Queguiner, G., Fradet, M., and Rouhani, M. Towards mobile diminished reality. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)* (2018), pp. 226–231.
12. Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2001, Vol.1, pp.I-I*.
13. Wikipedia. Camera resectioning - Wikipedia. `https://en.wikipedia.org/wiki/Camera_resectioning#Parameters_of_camera_model`. Accessed 2019-01-18.
14. Yagi, K., Hasegawa, K., and Saito, H. Diminished reality for privacy protection by hiding pedestrians in motion image sequences using structure from motion. *2017 IEEE International Symposium on Mixed and Augmented Reality, Oct. 2017, pp.334-337*.