Enhanced Augmented Reality: Object Modeling and Rendering via Aruco Marker-based Computer Vision Techniques

Anubhav Parbhakar   Shea Mccormack
Department of Computing Science
University of Alberta
{aparbhak, samccorm}@ualberta.ca

Abstract

We investigate the applications of augmented reality (AR) as its implementation using computer vision techniques. We attempt to create a program that will replicate a popular emerging subsection of AR being utilized in marketing that allows users to model furniture in their own homes before completing a purchase. We approach this task of modeling objects in AR through the utilization of calibration sheets, with ARUCO markers printed on the sheets for tracking. We customize the aforementioned ARUCO markers in a way that generates ones that are optimized for tracking. We thereafter utilize pre-rendered objects loaded in the form of .obj files to represent the 3D object, with the faces, and the corresponding vertices of the object being represented in a 3D space. The faces define how the vertices are connected in a polygonal sense. We subsequently utilize computer vision techniques, such as camera matrix and homography estimation to procure a projection matrix for the 3D representation for the given frame. We utilize a perspective transformation to translate from 3D to 2D as necessitated, with the object appearing as an overlay on top of the screen, with the AR rendering being complete. To add value to rendering, we allow the user to rotate and translate the object, allowing for manipulation of the object in a setting of the user's choice, adding important value to the product, highlighting the feasibility and practicality of utilizing computer vision and computer geometry to create valuable and desired products.

1 Introduction

This section focuses on the background of augmented reality (AR), delving into the background and specifics of the usage of AR technology, and depicting its significance to users. It derives the current problems with this specific utilization of the technology from the presented scenarios, and thus outlines the objective of the project, modulating the overarching goal into smaller facets, explaining the part each aspect plays in the overall objective.

1.1 Background and Significance

The usage of augmented reality (AR) has become widespread, permeating throughout different aspects of the tech industry, being used for marketing, accessibility, and as an evolutionary step for entertainment. One subsection of the applications of AR is in marketing, specifically being used for modeling and visualizing furniture before purchase, as a way of allowing the user to envision the product in their own home, thus driving sales and value for the product in a revolutionary way. As seen in numerous applications made readily available to consumers, the key draw of such applications presents itself in the form of allowing the user to select the object/product of choice to be modeled, with a related 3D AR rendering being shown, presenting itself, as aforementioned, as an excellent way for users to envision owning and displaying the product in the setting of their choice, thus making them more likely to purchase the product. This added value of AR for businesses makes it a key topic of research and development in the industry, as well as an exciting new tool for accessibility and comfort for users.

1.2 Problem Statement

Despite this push towards the utilization of AR as a marketing tool through the enabling of object modeling, the objects themselves still suffer from common issues such as glitching, in which the object might go through another, or simply slide off the screen. Similarly, scaling and transformation of the object also present an issue, with the shakiness of the object hindering proper modeling and prospecting of the product by the user, as well as a lack of built-in agency and control for the user, forcing them to move the camera/device from which the objection projection is occurring, injecting a level of inconvenience and discomfort for the user. Additionally, as a result of the implementation of projection from the phone, a hovering effect may occur, as the projection is tethered to the camera itself, rather than the surface on which the object is modeled.

1.3 Objective

This project aims to address these issues through the utilization of ARUCO markers for object tracking, and a combination of computer vision techniques relating to computer geometry. The project utilizes an approach that divides the intricacies and challenges surrounding 3D AR objection projection into different aspects. Initially, the experiment revolves around the importance of accurate and optimized camera calibration, utilizing various techniques to maximize tracking accuracy, with this being treated as the equivalent of a setup/calibration step typically presented to users when engaging with similar technology as a consumer. Additionally, the importance of the ARUCO markers and the calibration sheet is also emphasized. Subsequently, the most essential part of the projection, the rendering, and projection of the object, dealing with the issue of varied dimensionality provided by different components involved in AR projection, is handled through the utilization of fundamental computer visions, with these exact methodologies being investigated further in the upcoming sections. Furthermore, the project revolves around giving maximum control to the control, and thus also focuses on allowing and enabling the user to manipulate the object in multi-dimensional ways. Summarily, this project drives to deliver a high-fidelity, optimized prototype with a vision of becoming a fundamental building block for a user-focused AR, object modeling, and visualization product.

2 Methodology

This section describes in detail the various techniques used to elicit optimized and accurate 3D AR projections, spanning from calibration as a setup step, diving into the specifics regarding proper camera setup, and covering the calculation of key calculations needed for apt object rendering. Subsequently, the rendering aspect is covered in detail, entailing the homography between the camera and the markers, as well as the formulation of the calculations behind handling the complications behind aspects with varying dimensionality. The overall process described below is diagrammed in Figure 1.

2.1 Calibration

Camera calibration is a key part of ensuring accurate augmentation of objects. Within the scope of this project, it is treated as the equivalent of a setup set often seen featured in applications utilizing similar technology. Utilizing similar approaches [1] as a point of reference, the methodology revolves around utilizing a set of ARUCO markers to constitute a board of ARUCO markers in a chessboard, known as a ChArUco board. An example of such a ChArUco board, which is generated from a separate script, is pictured in Figure 2. This chessboard-esque pattern aids in enhanced detection precision. This combination of ARUCO markers, which have been historically utilized for computer vision tasks, such as in robotics, and a chessboard pattern results in more robust and accurate calibration.

The process involves utilizing openCV, a commonly used Python library for computer vision tasks, to create a board full of ARUCO markers of shape 6 by 6, with one ARUCO marker generated coming from a dictionary of 250 unique recognizable patterns, hence the type DICT 6X6 250, which is a term commonly used in the code base for this project. Subsequently, as part of the setup process, the script requires the user to print out and capture this board from various angles and distances from the camera, with approximately 10-20 images required to be saved for later processing. Each of these images is then analyzed using OpenCV's built-in methods relating to ARUCO marker processing, e.g.,

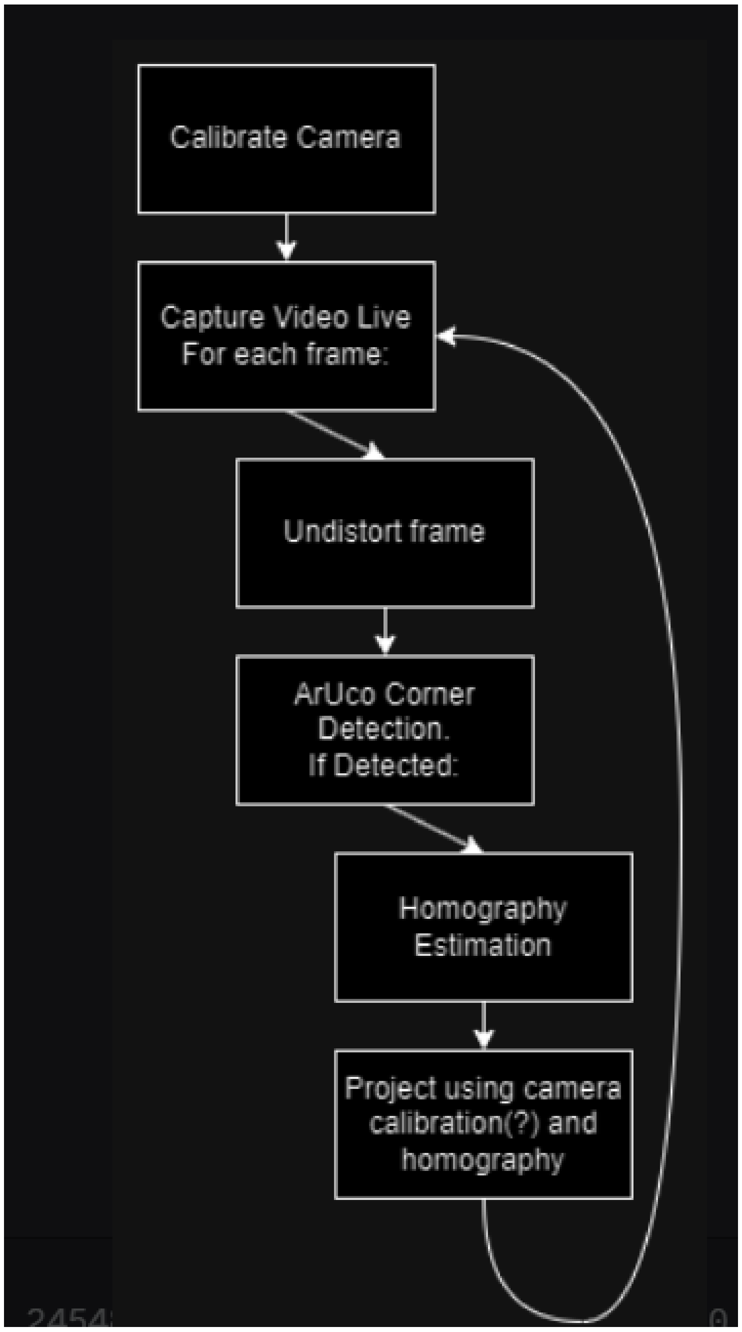

Calibrate Camera
Capture Video Live For each frame:
Undistort frame
ArUco Corner Detection. If Detected:
Homography Estimation
Project using camera calibration(?) and homography

Figure 1: overall calibration + rendering process

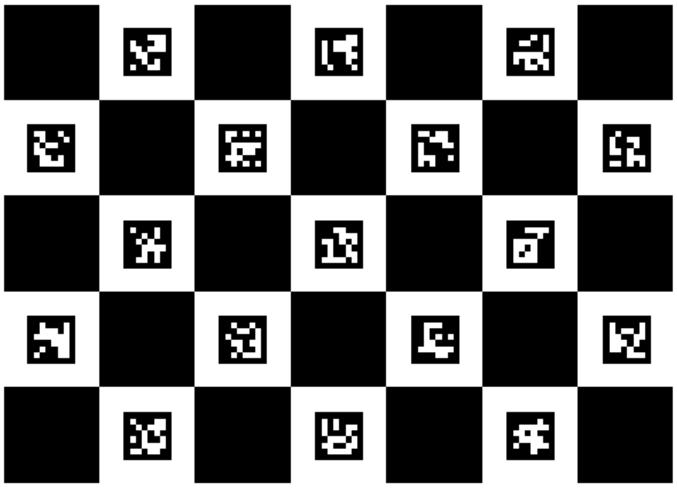


Figure 2: Example ChArUco marker

the function interpolateCornersCharuco, which is utilized to detect the corners of the ARUCO markers, with these corner estimates being used to improve calibration accuracy. At the end of this process, the detected corners, and their corresponding IDs from all images aggregate as arguments for another useful function, calibrateCameraCharuco, which outputs the calculated camera matrix, as well as the distortion coefficients. These outputted calculations contain the specifics behind the camera's internal characteristics, which are critical for object modeling in AR.

2.2 Rendering

A key part of the modeling of a 3D object in AR, rendering involved overlaying a 3d object onto a real-world image frame, with this being accomplished through real-time detection of AR markers to calculate transformations and subsequently render the model. Initially, after the setup stage revolving around calibration has been completed, the script initializes the necessary variables for the .obj object being rendered, the calibration sheet (Figure 3), and starts video capture, reading the video as a set of image frames, until the script is terminated. Through the cycle of analyzing frames, each frame is first processed through an undistorting filter, which essentially takes away all perspective from the frame that is read (Figure 4). This prepossessing step becomes necessary to improve ARUCO tracking, as well as in improving homography estimation (the transformation matrix that maps the plane with the ARUCO markers to the camera plane), and projection accuracy of the object.

Following this, a new camera matrix is determined for each frame through the utilization of the camera matrix and distortion coefficients calculated in the

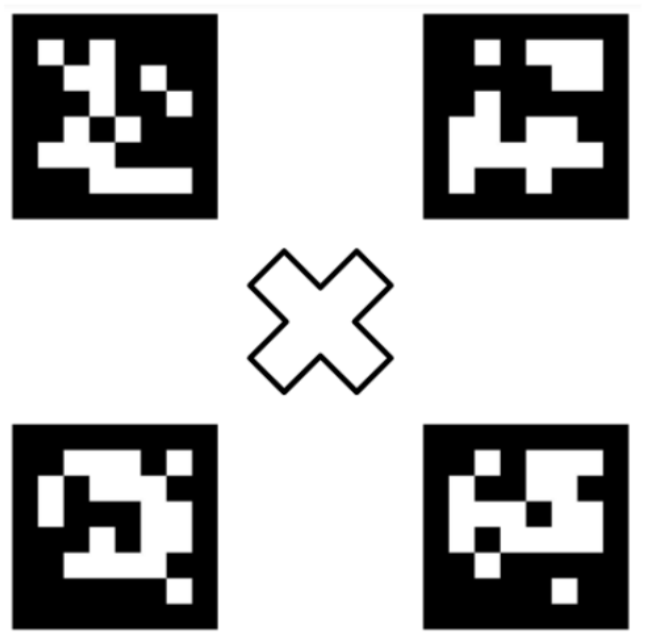


Figure 3: Source point board

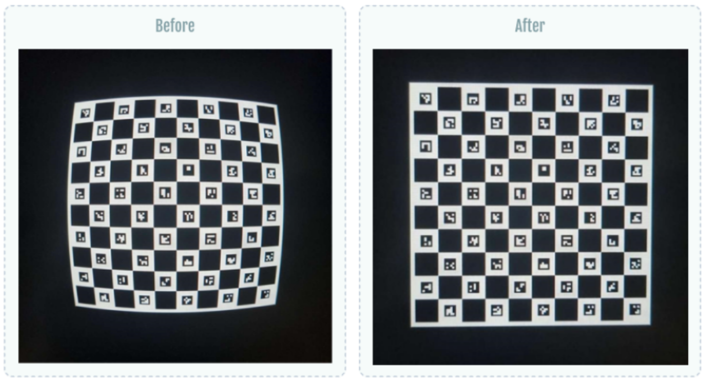


Figure 4: Undistorting prepossessing step

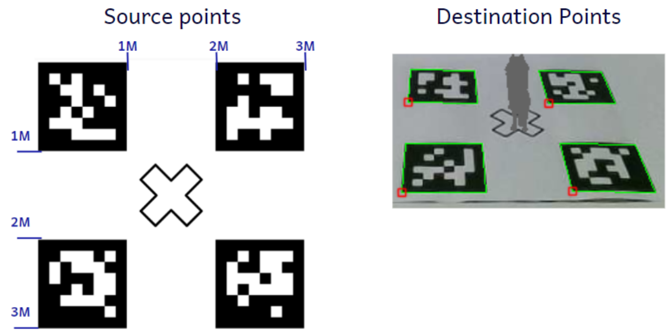


Figure 5: Tracking correspondences for tomography estimation

previous setup stage (Figure 5). Subsequently, the presence of motion is analyzed through the detection of the corners of the ARUCO markers, with the calculation of the 3D projection matrix occurring once non-zero homography has been calculated (calculated from the set of corresponding points between the marker and the camera plane). Additionally, only 2 ARUCO markers are necessary for homography estimation (8 corner coordinates), with the coordinates being utilized in the DLT Algorithm (Algorithm 1)[2]. The DLT algorithm outputs the homography described above through mapping from 3D to 2D, with a key advantage of using this algorithm over others being its requirement of only six correspondences for successful homography estimation [2].

Following homography estimation, as aforementioned, we utilize the resulting homography to attain the 3D projection matrix, which transforms the 3D coordinates into 2D image frames. This matrix is obtained through the combination of the newly computed camera calibration matrix, and the homography, which results in the mapping of 3D points onto the 2D plane while maintaining the characteristics of the camera captured by the pre-computed camera calibration matrix and distortion matrix from the previous setup stage, as well as the spatial orientation of the marker itself. In particular, it follows the general structure of computing the rotation along the x and y axis along with any translation, after which, the orthonormal basis is computed post-normalization, with the 3D projection matrix finally being computed from the model to the current frame, with this being formalized in Algorithm 2 [3].

Algorithm 1: Direct Linear Transform (DLT) Algorithm

```
Input: Corresponding 2D and 3D points: $\{(x_i, y_i)\}_{i=1}^n$ and
       $\{(X_i, Y_i, Z_i)\}_{i=1}^n$
Output: Transformation matrix $H$
// Construct the matrix $A$
Initialize an empty $2n \times 9$ matrix $A$;
for $i = 1$ to $n$ do
    $A[2i-1, :] \leftarrow [0, 0, 0, -Z_i x_i, -Z_i y_i, -Z_i, Y_i x_i, Y_i y_i, Y_i]$;
    $A[2i, :] \leftarrow [Z_i x_i, Z_i y_i, Z_i, 0, 0, 0, -X_i x_i, -X_i y_i, -X_i]$;
end
// Compute the Singular Value Decomposition (SVD) of $A$
Compute the SVD: $A = U\Sigma V^T$;
Extract the last column of $V$ as $h$;
Reshape $h$ into a $3 \times 3$ matrix $H$;
return $H$
```

Algorithm 2: Projection Matrix Calculation

```
Input: Camera parameters $camera\_parameters$, Homography
       $homography$
Output: 3D projection matrix $projection$
$homography \leftarrow homography \times (-1)$;
$rot\_and\_transl \leftarrow \mathrm{dot}(\mathrm{inv}(camera\_parameters), homography)$;
$col\_1 \leftarrow rot\_and\_transl[:, 0]$;
$col\_2 \leftarrow rot\_and\_transl[:, 1]$;
$col\_3 \leftarrow rot\_and\_transl[:, 2]$;
$l \leftarrow \sqrt{\mathrm{norm}(col\_1, 2) \times \mathrm{norm}(col\_2, 2)}$;
$rot\_1 \leftarrow col\_1/l$;
$rot\_2 \leftarrow col\_2/l$;
$translation \leftarrow col\_3/l$;
$c \leftarrow rot\_1 + rot\_2$;
$p \leftarrow \mathrm{cross}(rot\_1, rot\_2)$;
$d \leftarrow \mathrm{cross}(c, p)$;
$rot\_1 \leftarrow \mathrm{dot}(c/\mathrm{norm}(c, 2) + d/\mathrm{norm}(d, 2), 1/\sqrt{2})$;
$rot\_2 \leftarrow \mathrm{dot}(c/\mathrm{norm}(c, 2) - d/\mathrm{norm}(d, 2), 1/\sqrt{2})$;
$rot\_3 \leftarrow \mathrm{cross}(rot\_1, rot\_2)$;
$projection \leftarrow \mathrm{stack}((rot\_1, rot\_2, rot\_3, translation))^T$;
return $dot(camera\_parameters, projection)$;
```

Rendering the object itself involves projecting the 3D vertices from each face of the .obj object to the 2D image frame after the points of each face of the object have been determined, utilizing a scale matrix to ensure scale, as well as adjusting the location of the object on the calibration to be the located in the center of the four markers. After which, through the utilization of the calculated projection matrix, a perspective transformation is performed to visually integrate the 3D vertices into a 2D plane, resulting in the set of 2D points that describe the 3D model in the 2D camera plane. As a last step, in order to complete the rendering process, the polygons between the projected points are filled in, which completes the rendering of the 3D object by accurately representing the 3D object from the perspective of the 2D camera plane.

3 Experiment

In this section, we delve into the experiment itself, which revolved around giving the user autonomy and control over the object projected. We discuss the different types of transformations that the object experienced as a result of user manipulation, as well as the different objects that were available to model, and fixes that were made mid-testing to further optimize performance and fidelity.

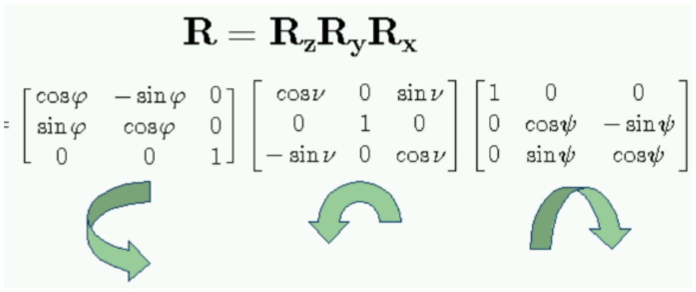
3.1 User Input

Within the experiment, the user was afforded the ability to dynamically interact with the 3D object presented in AR. As aforementioned, adding user input and control over the object was essential to the approach taken, in an attempt to apply AR in a value maximizing way. In order to enact such interaction, the script captures key presses from the user, namely the standard keys for translational movement (W,A,S,D), for movement along the x and the y axis, being represented in the Python script as X and Y displacement variables. Furthermore, additional manipulative agency is given to the user through the utility of the R,F,T,G,Z, and X, keys which are used to manipulate the rotation across the x, y, and z axis, resulting in allowing the user essentially full control over the object.

Note: Although our project is geared towards modeling, for testing purposes, .obj models of furniture exceeded our current hardware capabilities, as such a decision was made to test full rotational control over the object, rather than the typical one-dimensional rotation seen, although the switch over is trivial, and one may still choose to only rotate along one axis.

3.2 Rendering changes

As can be surmised, adding displacement in the various ways outlined above results in the need to add changes to the method of calculation for various matrices. Initially, for each key press, for each frame, there is a 10-pixel offset recorded for translational movements, and a 5-pixel radial offset recorded (negative or positive), for rotational changes. These offsets are then recorded in global variables in the script, in the form of translational displacement scalar values and rotational vectors for rotational changes. For rotational changes, the model's vertices are directly updated, through the utilization of Euler angles, which utilizes the computation of three different rotation matrices Rx, Ry, Rz,



$$\mathbf{R} = \mathbf{R_z}\mathbf{R_y}\mathbf{R_x}$$

$$= \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\nu & 0 & \sin\nu \\ 0 & 1 & 0 \\ -\sin\nu & 0 & \cos\nu \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix}$$

Figure 6: Euler Angles computation

to attain their product R = Rz * Ry * Rx, which represents the combined rotation matrix used to transform the object vertices (Figure 6). Furthermore, translational changes are implemented into the calculations at the time of rendering the object from the vertices. More specifically, during the process of procuring the points relating to each face of the object via the vertices of the object, when the location of the object is updated such that it initially starts at the center of the calibration sheet, the x and y displacement are also factored, for each frame. As such, for each frame, translational movement is able to be directly implemented when implementing the 3D rendering of the object from the perspective of the 2D camera plane. Additionally, it is worth noting that user input is enabled through the usage of the waitkey function implemented in the openCV library of functions, which allows the script to wait a set period of time for each frame for user input, thus allowing the user time to insert any desired manipulation to the object itself.

3.3 Methodology Optimization

Through the course of initial experimentation, a certain problem presented itself that could not be ignored, or be attributed to small adjustments in the orientation of the calibration sheet. Namely, the problem that asserted itself revolved around the fidelity of the rendered object itself, as, upon rendering, the object would often start to move locations, appearing to jump back and forth between various coordinates, without any user manipulation, thus definitively giving the impression of a low fidelity object, which was a common problem cited in the introduction of this paper. As a result, in order to address this issue, two solutions working in tandem were proposed, and enacted, with the first being associated with a simple yet fundamental principle of computer vision, temporal difference. In essence, the temporal difference gives the motion, or amount of change, between two frames in a video feed, thus, thresholding this difference, or placing a minimum amount needed to count as motion, allows for stabilizing of the rendering, and discounting small changes from factoring into the 3D modeling in AR of the object. Subsequently, the second solution revolves around utilizing the RANSAC algorithm when computing the tomography. The RANSAC algorithm detects outliers when computing tomography, subsequently excluding these values from the calculation, resulting in a substantially more accurate tomography estimation, more stable and high-fidelity rendering, and minimization of tracking error.

4 Results

This section showcases the results of the various user input transformations and translations, as well as other objects that were available to the user for modeling.

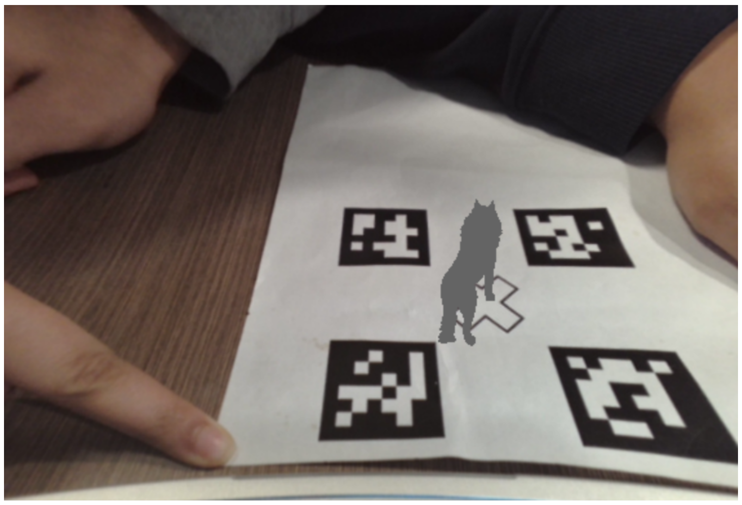


Figure 7: Wolf object centered

5 Conclusion

In conclusion, through the utilization of numerous computer vision techniques, this project was able to accomplish the task of 3D modeling using AR, with the combination of computer vision and computer geometry techniques resulting in a high-fidelity rendering of objects, as well as total control over the manipulation of the objects, with the end goal of this prototype being used towards furniture being all but realized, with the functionality being able to be scaled to furniture, and any other product, giving the user ability to properly model their products before purchasing, enabling the envisioning on ownership of products, making it more likely for the product resultingly to be bought, and ultimately displaying the high value of this prototype.

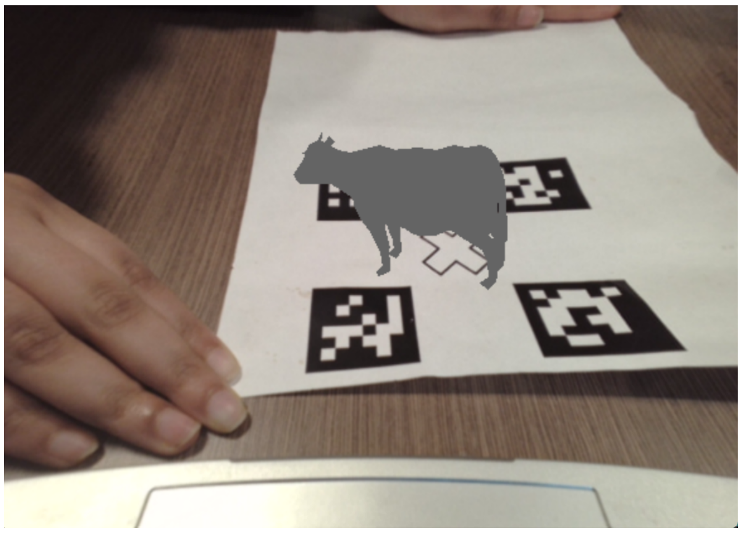


Figure 8: Cow object with rotation applied

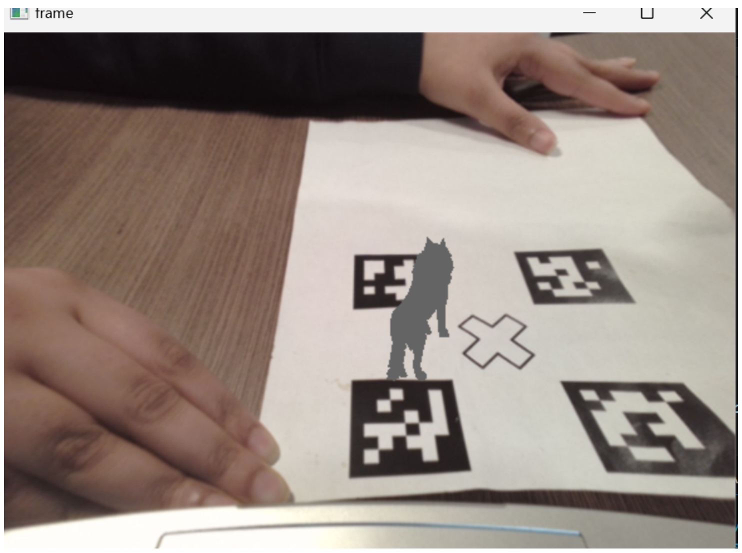


Figure 9: Wolf object with translation applied

6 References

[1] Nielsen, Nicolai. Camera Calibration in less than 5 minutes with OpenCV. https://www.youtube.com/watch?v=_-BTKiamRTg. [2] Acin, Juan Gallostra. ar_main.py. https://github.com/juangallostra/augmented-reality/blob/master/src/ar_main.py

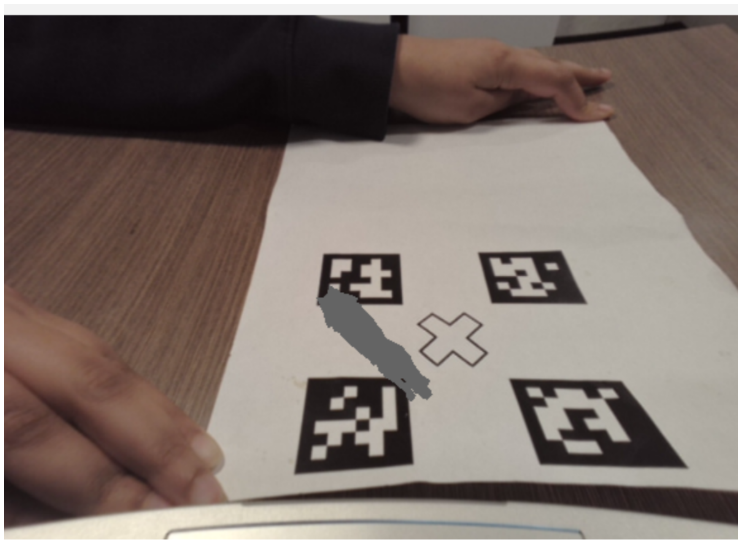


Figure 10: Wolf object with translation and rotation applied

[3] National Library of Medicine. Camera Calibration with Weighted Direct Linear Transformation and Anisotropic Uncertainties of Image Control Points. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7071080/: :text=The