Creating Augmented Reality Experiences : Leveraging Camera Geometry for 3D Object Projection

KHELIF Amine Khelif and BOUHEDJA Wail

I. INTRODUCTION

Augmented Reality (AR) is a groundbreaking technology that seamlessly integrates digital elements into the real world, enhancing our perception and interaction with our surroundings. By overlaying computer-generated information, such as graphics, sounds, or data, onto the physical environment, AR blurs the lines between the virtual and real, offering users a unique and immersive experience.

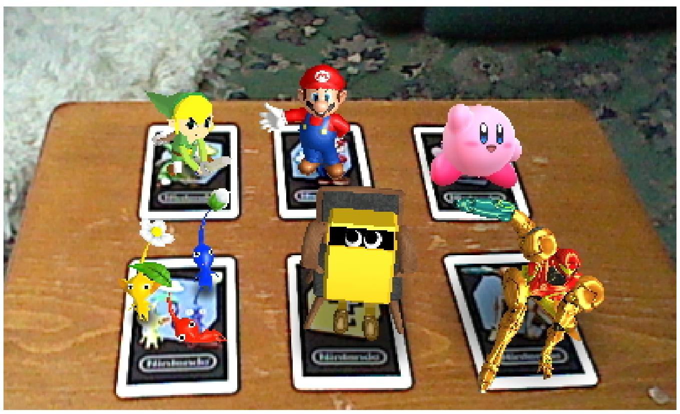

Fig. 1. Nintendo AR Card Games

In this project, we did the very few starting steps in this field, and had only a glimpse of what we can do with methods like calibration, homography, pose estimation and tracking combined. The goal was to create 3D-shape a cubic hologram, just by scanning a QR-code with a remote phone's camera.

II. METHODS USED

A. Camera Calibration

Camera calibration is a fundamental process in computer vision and photography that involves determining the intrinsic and extrinsic parameters of a camera system. In simpler terms, it's the method by which we adjust and refine a camera's settings to ensure accurate and reliable measurements in the images it captures. The intrinsic parameters encompass properties like focal length, lens distortion, and image sensor characteristics, which are unique to each camera. On the other hand, extrinsic parameters involve the camera's position and orientation in 3D space relative to the scene being photographed.

Accurate camera calibration is crucial for applications ranging from 3D reconstruction and object tracking to augmented reality and robotics. Thus, to attain this accuracy, we were given a 10x7 checkboard pattern to take several photos from different angles and orientations in order to calibrate the camera. What we did was slightly different but has the same outcome, we took a video moving around the checkboard pattern in every possible direction and wrote a Python script to extract about 400 frames of the video and use them as samples for the calibraion process. As for the process, by utilizing MATLAB's Computer Vision Toolbox, we identified the locations of checkerboard corners in the images and extracted the corresponding image points. Subsequently, these points were employed to calculate the camera parameters by establishing correlations with their respective real-world coordinates.

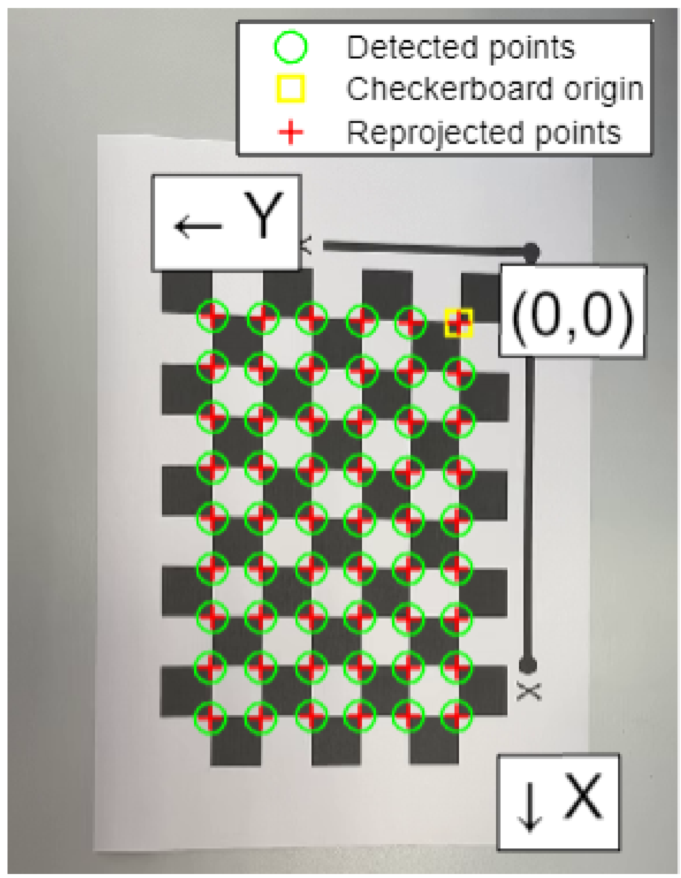

Fig. 2. Checkerboard Pattern images extracted from video footage

As for the process, by utilizing MATLAB's Camera Calibrator app, we identified the locations of checkerboard corners in the images and extracted the corresponding image points. Subsequently, these points were employed to calculate the camera parameters by establishing correlations with their respective real-world coordinates. After making sure that the camera calibration was accurate, we used MATLAB's Camera Calibrator app to export the camera parameters (Intrinsic and Extrinsic).

Intrinsic Parameters: Represented by a 3x3 matrix K:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f_x and f_y are the focal lengths along the x and y axes, c_x and c_y are the optical center coordinates and s the skew coefficient.

Extrinsic Parameters: Rotation (R) and Translation (T): Represented by a 3x4 matrix $[R|T]$:

$$[R|T] = \begin{bmatrix} R_{3x3} & T_{3x1} \end{bmatrix}$$

Combining both intrinsic and extrinsic parameters, the camera projection matrix (P) is given by:

$$P = K[R|T]$$

The projection matrix defines the transformation from 3D world coordinates to 2D image coordinates, incorporating both intrinsic and extrinsic camera parameters.

B. Homography

In computer vision, a homography refers to a projective transformation that maps points from one plane to another. Specifically, it describes the relationship between two sets of corresponding points in different images or scenes taken from the same camera. All the information about rotation, translation and other transformations can be found in a 3x3 matrix called the homography matrix. This latter gives us the coordinates of the points in the new view by multiplying it with the original coordinates. To find the elements of this matrix, we did as below:

We used QRCodeDetector from OpenCV python library to pick trivial points of a QRCode and establish a correspondance between the points on the next frame taken. (X,Y) being the coordinates in the first image and (x,y) being the coordinates in the second one, the problem can modeled as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

with

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

After developing the matrix multiplication, we obtain these three equations:

$$x_i = h_{11}X_i + h_{12}Y_i + h_{13} \quad (1)$$

$$y_i = h_{21}X_i + h_{22}Y_i + h_{23} \quad (2)$$

$$1 = h_{31}X_i + h_{32}Y_i + h_{33} \quad (3)$$

We divide the two first equations (1) and (2) by the third one to have :

$$x_i = \frac{h_{11}X_i + h_{12}Y_i + h_{13}}{h_{31}X_i + h_{32}Y_i + h_{33}}$$

$$y_i = \frac{h_{21}X_i + h_{22}Y_i + h_{23}}{h_{31}X_i + h_{32}Y_i + h_{33}}$$

Then we simplify them to:

$$0 = h_{11}X_i + h_{12}Y_i + h_{13} - x_i(h_{31}X_i + h_{32}Y_i + h_{33}) \quad (4)$$

$$0 = h_{21}X_i + h_{22}Y_i + h_{23} - y_i(h_{31}X_i + h_{32}Y_i + h_{33}) \quad (5)$$

We can rewrite these linear equations in a matrix product form :

$$\begin{bmatrix} X_i & Y_i & 1 & 0 & 0 & 0 - X_i x_i & -X_i y_i & -x_i \\ 0 & 0 & 0 & X_i & Y_i & 1 - y_i X_i & -y_i Y_i & -y_i \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (6)$$

Now that we have 9 unknowns in this linear system, the element $h_3 3$ is forced to the value 1, and we need a minimum of 4 points to generate at least 8 equations so that the problem can be solved and find the coefficients. The calculations can be done by the Singular Value Decomposition method to solve this system of linear equations.

The Singular Value Decomposition (SVD) is a mathematical technique used in linear algebra to decompose a matrix into three simpler matrices. It has various applications, and one of them is in computer vision for finding the homography matrix.SVD decomposes a matrix A into three matrices U, Σ, and V^T:

$$A = U \Sigma V^T$$

U and V are orthogonal matrices.

Σ is a diagonal matrix containing the singular values.

Now to find the homography matrix H, we follow the following steps:

Homogeneous System Representation: Assume you have corresponding points in two images, denoted as $(X_i, Y_i, 1)$ and $(x_i, y_i, 1)$. The homography matrix H transforms points from one image to the other.

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \approx H \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}$$

Constructing the Augmented Matrix A: Form the matrix A by combining the equations for all corresponding points.

Each row in A is constructed based on the equations for a pair of corresponding points.

$$A = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & x'_1 \\ 0 & \dots & 0 & -x_n & -y_n & -1 & x_nx'_n & y_nx'_n & x'_n \\ 0 & \dots & 0 & -1 & 0 & \dots & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & 0 & 0 & -1 & x_ny'_n & y_ny'_n & y'_n \end{bmatrix}$$

Applying SVD Decomposition: Employ Singular Value Decomposition (SVD) on matrix A, decomposing it into three matrices U, Σ, and V^T.

Extracting Homography Matrix H: Retrieve the homography matrix H from the last column of V, reshaped into a 3x3 matrix.

$$H = Vt[-1].reshape((3,3))$$

with :

$$\mathbf{V} = \begin{bmatrix} v_{91} \\ v_{92} \\ \vdots \\ v_{99} \end{bmatrix}$$

Normalize H: Divide H by its last element v_{99}.
At the end, we obtain the Homography matrix H normalized.

C. Pose Estimation

As said above, the projection matrix P can be deduced from the Intrinsic Parameters matrix K (found by the calibration process) and the Extrinsic Parameters matrix $[R|T] = \begin{bmatrix} R_{3x3} & T_{3x1} \end{bmatrix}$ (R for Rotation and T for Translation). The projection matrix is fundamental for representing the transformation from a 3D world space to a 2D image plane. It enables us to map the 3D points of a shape onto the 2D image plane. So by considering (X,Y,Z) the coordinates of a point from a 3D shape, and (x,y) the coordinates of the point projected on the plane, we have :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K[R|T] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Now, by neglecting the depth factor and putting, it transforms into:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Which turns out to be the Homography matrix H calculated before with the SVD method:

$$H = K[R_1, R_2|T]$$

Multiplying both sides by K^{-1} leads to

$$K^{-1}H = [R_1, R_2|T]$$

Thus, we can extract the columns R_1, R_2, T, and calculate R_3 by crossing $R_1 and R_2$ to have the three columns R_i perpendicular with one another.

Although everything seems fine, a little probleme remains. In the decomposition of a homography matrix H, the resulting matrices approximating the rotation and translation of the camera may not perfectly yield an orthogonal rotation matrix. Ideally, a rotation matrix should be orthogonal, meaning its transpose is equal to its inverse, and it should have a determinant of +1 to preserve scale and handedness in the coordinate system.

To address this, we introduce a scaling factor α, calculated as the fourth root of the determinant of the matrix $[R_1, R_2, R_3]$ calculated previously. Then, we divide each $R_1 and R_2$ by α, and divide R_3 by α^2. We obtain the new matrix $[R'_1, R'_2, R'_3, T]$.

D. Point Tracking

Point tracking, also known as feature tracking or point correspondence, is a fundamental concept in computer vision and image processing. It refers to the process of locating and following specific points or features in a sequence of images or frames over time. The objective of point tracking is to establish the correspondence between points in different frames, enabling the analysis of the motion or changes occurring in a scene. For this part of the project, we used the Kanade-Lucas-Tomasi (KLT) algorithm, which is a popular method for point feature tracking in computer vision. It is widely used for optical flow estimation, which involves tracking the movement of keypoints or features between consecutive frames in a video sequence.The KLT algorithm is known for its efficiency and has been widely used in applications such as object tracking, motion analysis, and video stabilization.

Fig. 3. KLT algorithm tracking cars

III. Experimentations and results

This section details the empirical studies and results of the project, highlighting the experimental design, methodologies, and conditions for testing the augmented reality (AR) application. It compares these results with existing AR solutions to evaluate the project's effectiveness. Finally, it critically assesses the AR application's performance, identifying its strengths and areas for improvement.

A. Camera Calibration

The process of calibrating the camera was conducted using MATLAB's dedicated Camera Calibration App. Instead of individual images, a video was captured and split into 488 frames using Yamera, an open-source application known for its manual focus feature, essential for obtaining sharp frames vital for precise calibration. This approach is key to ensuring accurate calibration results. After converting the video into individual frames and importing them into the calibration tool, the calibration procedure commenced. The primary metric for evaluating calibration precision was the "Overall Mean Error," a statistical term referring to the average discrepancy in reprojection. Reprojection error quantifies the variance in pixels between where the image points actually appear and where the calibration model predicts they should be. This metric serves as an indicator of the fidelity of the camera calibration model in mapping image points. The outcomes of this process are illustrated in the subsequent histogram. (see Figure 4).

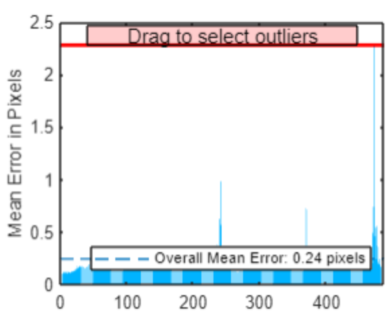

Fig. 4. Histogram of reprojection errors across the calibration images.

Once the calibration accuracy reached a satisfactory level, we used the "Export Camera Parameters" feature to transfer the camera parameters as an object into the MATLAB workspace.

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1121.1 & 0 & 358.7 \\ 0 & 1123.6 & 637.9 \\ 0 & 0 & 1 \end{bmatrix} \quad (7)$$

In ensuring the calibration's high precision, the "Overall Mean Error" was an essential metric. Our calibration process involved meticulous adjustments, leading to a reduction of the "Overall Mean Error" to a value under 1. This level of accuracy is critical, particularly for augmented reality (AR) applications where precise calibration plays a key role.

B. Homography

We began by creating a mathematical model for Homography and then developed the corresponding Python code. This code included both an analytical approach and an alternate method using Singular Value Decomposition (SVD) for cases where the matrix A was singular. The purpose of this code was to compute the homography matrix from four points detected and tracked by a QR code detector, along with the predetermined real-world coordinates of the QR code.

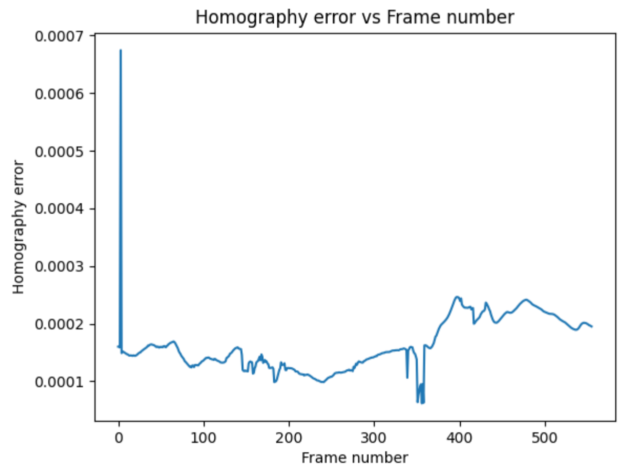

Fig. 5. Illustration of the homography error

The homography matrix computation involved these eight points (comprising four pairs of corresponding points). This matrix is crucial as it allows for the transformation of the perspective of a selected plane to a new viewpoint, determined by the destination points. To assess the precision and effectiveness of our computed homography, we conducted a comparison with the homography function implemented in OpenCV. We measured the homography computation error frame by frame using calculating the norm of the difference between our homography matrix and the one computed by OpenCV for each frame, providing a clear metric for evaluating the accuracy of our homography computation.

C. Projection

The section on Projection in our study highlights the successful application and evaluation of the projection matrix. This matrix was instrumental in mapping 3D world coordinates to 2D image coordinates, showcasing the precision of our approach. The effectiveness of the projection was evident through the accurate alignment of the projected points with their actual locations in the image.

A key aspect of our evaluation involved a detailed analysis of the projection's accuracy. This was accomplished by visually representing the projection results and quantifying them through metrics like the mean pixel error. The compelling accuracy of the projection matrix is showcased in the results, which are visually depicted below. Additionally, a video attached to this paper further illustrates the effectiveness of the projection in real-world scenarios, providing a dynamic and comprehensive view of the project's success in this area. The results affirm the robustness and precision of our projection methodology, making it a valuable component in our study.

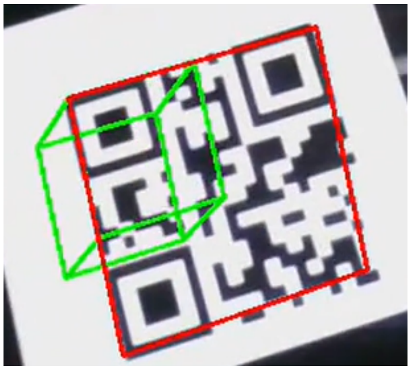

Fig. 6. Illustration of the projection

D. *Point Tracking*

In our augmented reality (AR) system, we have employed the Kanade-Lucas-Tomasi (KLT) algorithm, a method renowned for its accuracy and efficiency in tracking points across sequential images. This technique is integral to AR applications, as it allows for the precise overlay of virtual elements onto real-world imagery by maintaining a consistent reference across successive frames.

The process commences with the initialization of the point tracker. This essential step involves selecting a set of key points, typically prominent features or corners in a reference image, that the algorithm will track throughout the video sequence. These points serve as the anchors for the dynamic overlay of virtual objects.

Dynamic Tracking Across Video Frames: Once initialized, our system continually applies the point tracker to each frame in the video. The KLT algorithm's core function is to detect subtle movements of these points between consecutive frames. It accomplishes this by analyzing the surrounding pixel intensities and determining the optimal match for each point in the following frame. Consequently, the system generates a collection of new point positions, each accompanied by a 'validity' indicator. This indicator is crucial, as it signifies whether a point has been consistently tracked from one frame to the next.

Enhancing Tracking with QR Code Detection: To augment the reliability of our tracking system, we incorporate QR code detection. This feature is particularly beneficial when the existing tracking points become unreliable or are lost. In such scenarios, the system detects QR codes within the frame, utilizing the detected codes' corner points as new references for tracking. This dual approach of KLT point tracking and QR code detection ensures a robust and continuous tracking mechanism, vital for maintaining the integrity of the AR experience.

Continuous Evaluation and Adaptation: The performance of our tracking system is under constant evaluation to ensure its effectiveness across diverse scenarios and conditions. By meticulously monitoring the accuracy of point tracking and adapting to new points provided by QR code detection, our system demonstrates a high degree of precision and stability. This approach is pivotal in crafting an immersive and interactive AR experience, where virtual and real-world elements coexist seamlessly.

1) Results: The produced video demonstrates a digitally-rendered cube positioned seamlessly on the QR code, giving the illusion of being an actual object in the environment. This cube consistently aligns with the QR code, maintaining this synchronization despite changes in the camera's perspective and distance, all thanks to our implemented point tracking technology.

We have developed a testing server that enables script access through a web browser, allowing for cross-device functionality. To utilize this feature, run the app.py file and navigate to the provided HTTPS link in your browser. This setup permits camera access on mobile devices, but it requires all devices to be connected to the same network. Once set up, you can explore the tool's capabilities.

Alternatively, you can directly execute the image_processing_script.py file to witness real-time rendering on your current device. For processing pre-recorded videos, modify the line cap = cv2.VideoCapture(0) in the main function to cap = cv2.VideoCapture('video_path'), replacing 'video_path' with the path to your video file. This adjustment enables the script to process and render the specified video instead of using a live camera feed.

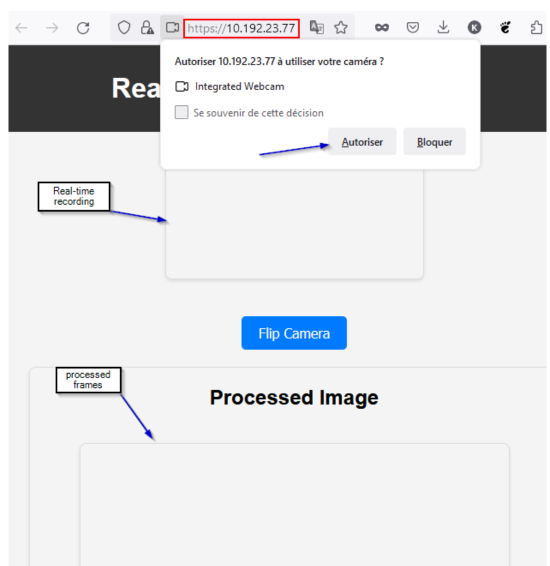

Fig. 7. Front-end processing script.

IV. CONCLUSION

In this project, our team delved deeply into the essential mathematical and image analysis techniques that are fundamental to Augmented Reality (AR). This exploration led to the practical application of these theories, notably in projecting a 3D cube from a QR code image within an AR context.

The burgeoning growth of AR technology in diverse fields, ranging from healthcare education to entertainment, highlights the significance and timeliness of our endeavor. By achieving the projection of a 3D object onto a real-world backdrop, we have made a significant stride in forging an integrated AR experience. This achievement not only validates the effectiveness of our AR application but also showcases the transformative potential of AR in altering our interaction with the surrounding world. The completion of this project has enriched our understanding of AR's practical applications and the mathematical intricacies underlying its implementation.

REFERENCES

[1] A Systematic Review of 10 Years of Augmented Reality Usability Studies: 2005 to 2014 - Frontiers in Robotics and AI, 2015.

[2] Augmented reality and virtual reality displays: emerging technologies and future perspectives - Light: Science Applications, 2020.

[3] Augmented Reality: A Systematic Review of Its Benefits and Challenges in E-Learning Contexts - ResearchGate, 2019.

[4] Simon Baker, Iain Matthews, "Lucas-Kanade 20 Years On: A Unified Framework".