TECHNICAL PAPER

KEYWORDS CAMARA TRACKING // OPEN SOURCE TOOLS // UNREAL ENGINE // OPTIC FLOW

Camera Tracking Systems and their Democratization

By Irene Muñoz López and Andrew Gilbert

Abstract
Computer vision encompasses the analysis, processing, and interpretation of visual data. Tracking is a subset of this field, where systems recognize objects or salient features in a scene to determine their displacement across subsequent frames in a video stream. This facilitates automation, increases efficiency, and expands the functionality of these systems to applications in surveillance, medicine, and entertainment, among other fields. In recent years, Virtual Reality (VR) and Augmented Reality (AR) systems have gained popularity, prompting the development of camera tracking techniques. Camera tracking assesses the geometry and poses of a camera within a scene. Many tools are available to analyze and process camera tracking information, but most are proprietary, making information about them scarce; their availability to the general public also varies. To determine the democratization of the technology, three different tracking systems were compared. Two of these systems are standard tools used in the industry; the third system was a tracker built using OpenCV's open-source tools. A dataset of tracking values under different video parameters was gathered for all three trackers. By comparing and examining these results, it was determined that the tracking system was built using OpenCV and met industry standards. The impact of noise and lower resolution on the tracking system's performance was also assessed qualitatively by comparing tracking results in Unreal Engine. These results revealed that the democratization of tracking technology is limited by the equipment that the general public can access. This research aimed to understand better the workflow, optimization, and democratization of camera tracking systems.

Computer vision, through training systems to analyze digital images or videos, enhances workflow automation and efficiency.[1] Identifying and tracking objects reliably is a popular field in computer vision, with wide-ranging applications in surveillance, security, medicine, and human-computer interfaces.[2] Tracking relies on following motion across a sequence of frames in a video stream, which may be done by focusing on specific scene areas or object features. Thus, in tracking, challenges may arise from noise, changes in illumination, and occlusions.[3-6] Another significant source of error is resolution.[7,8] These can all affect feature detection, hindering the matching of features across frames and depth estimation.

Camera Tracking estimates the camera's geometry and poses in a scene.[9] This is essential for Virtual Reality (VR),[10] and Augmented Reality (AR),[11] and the global pandemic also encouraged a more widespread use of the technology in Virtual Production (VP).[12,13] Camera tracking systems can be either marker-based or marker-less. Marker-based systems track motion by following the displacement of acoustical, mechanical, magnetic, or optical markers.[14] While reliable, these systems have a more complex setup and upkeep, and noise or marker occlusion from movement can hinder tracking performance. Marker-less systems make use of direct scene features for tracking. Relying on the correct detection of features rather than pre-made markers adds complexity and a greater risk of error to the tracking system, but it also provides greater flexibility, since the scene does not have to be prepared with the careful placement of markers before tracking.[15] Therefore, marker-less systems are more accessible to the general public.

There is a range of software tools available for markerless camera tracking, such as Adobe's After Effects[16] (paid-for) and Blender[17,18] (free access), with varying levels of public accessibility.

Unreal Engine[19] (UE) (free access), a popular software used

1545-0279/26©2026SMPTE

in the film and video games industry, offers open-source tools for simulations and pre-visualization. These systems have contributed to the democratization of camera tracking and VP, encouraging experimentation and creativity from a wider public.[20] Furthermore, OpenCV[21] (Open-Source Computer Vision Library) offers an open-source library of tools for building a tracking system. At a time of such advancements and claims of accessibility in camera tracking technology, its limits should be tested to properly assess the extent of its democratization.

This research aims to understand the technology behind camera tracking, determine the optimal parameters and limitations of common workflows, and assess the technology's accessibility to the broader public.

Contributions

1. The creation of a reliable camera tracker to compete with industry-standard tools (Blender and AfterEffects).

This provided a better understanding of the workflow of camera tracking systems used in industry software. Given the complexity of setup and maintenance for marker-based systems, this project focused on markerless systems. Genc *et al.*[9] proposed a system that first learns a scene's structure by tracking salient features, then computes an estimate of the camera's pose within the scene. The marker-less OpenCV system was tested against the tracking methods used by AfterEffects[16] and Blender.[17,18] These systems track the position, rotation, and scale of salient features in a scene to reconstruct camera motion. AfterEffects is a paid-for software tool, while Blender is free for public use. We tested these systems and compared their tracking results to assess the technology's accessibility, examining whether the paid options truly outperform the free ones and which one is better suited for low-budget productions that the general public might want to create.[22] Moreover, a tracker was built using OpenCV's open-source tools to further understand the components and workflow of a successful camera tracker and to determine whether it could meet industry standards.

2. Determine the democratization of the technology by researching how different video resolutions and noise levels affect tracking systems.

The detection and subsequent tracking of distinct features in a scene (areas of high contrast, edges, corners, texture changes) have been widely used in tracking,[3,4] but they can be susceptible to drift and inaccuracies in cluttered environments or in the presence of noise.[6] Even in recent systems, such as the one proposed by Zhou *et al.*,[5] noise remains a concern in tracking. Their model used deep learning and convolutional neural networks to improve robustness, yet the tracking accuracy degraded quickly in the presence of noise. Likewise, Handa *et al.*[7] and Younes *et al.*[8] agree that higher video resolution improves tracking accuracy. Both noise and resolution can affect feature detection in tracking, thereby hindering feature matching across frames and depth estimation. In researching how different video resolutions and noise levels affect camera tracking, we determined the technology's flexibility and availability to the general public.

3. The qualitative analysis of how different video parameters impact the performance of tracking systems in a virtual environment.

Significantly propelled by the global Covid-19 pandemic, virtual production methods have grown in popularity over recent years.[10,11] By testing the efficiency of the tracks in a virtual environment, the difference between technical precision and human perception,[23] and the real impact of the tested parameters, the study determined these differences.

Methodology

This section provides an overview of the method implemented to:

- Build an OpenCV camera tracker to compete with industry-standard tools.
- Develop a workflow to test the different tracking systems under varying video parameters and their effects in a virtual environment.

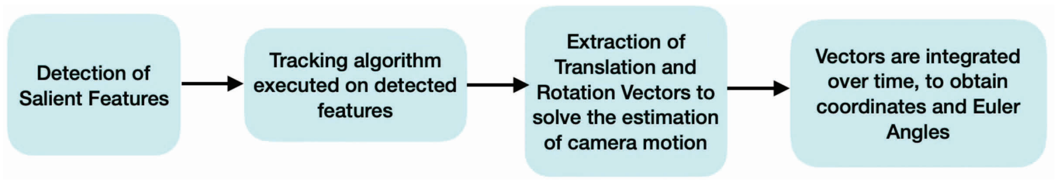
Detection of Salient Features
Tracking algorithm executed on detected features
Extraction of Translation and Rotation Vectors to solve the estimation of camera motion
Vectors are integrated over time, to obtain coordinates and Euler Angles

FIGURE 1. Overview of a tracking system.

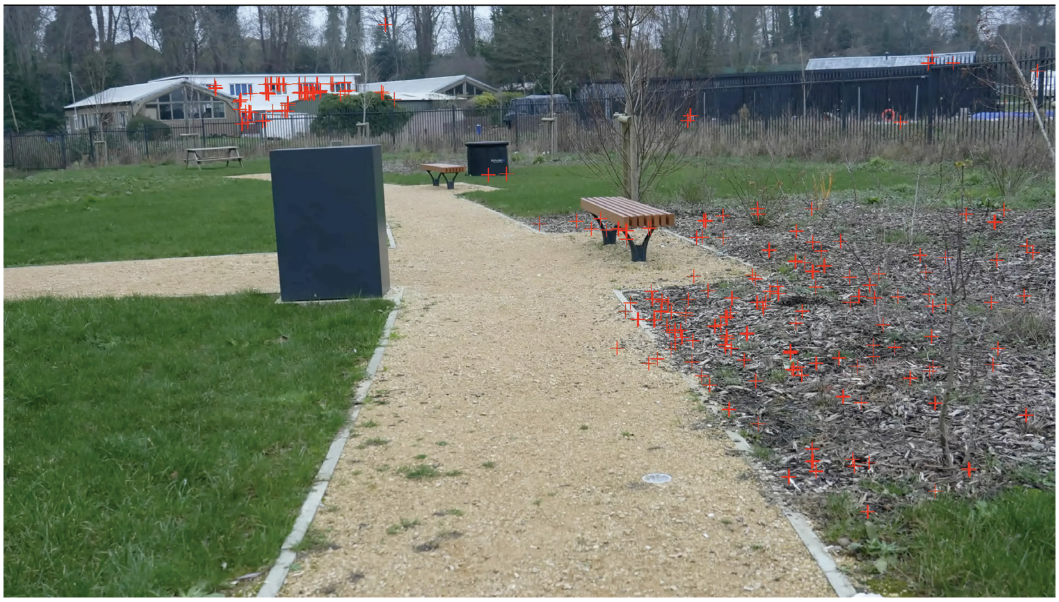

FIGURE 2. Salient features detected using OpenCV's SIFT LAM tools marked in a frame with red crosses.

Building an OpenCV Camera Tracker

This section breaks down the workflow of building a camera tracker using OpenCV. The tracker was built using OpenCV and follows the workflow shown in **Fig. 1.**

Detection of Salient Features

The detection and consequent tracking of salient features in a scene (areas of high contrast, edges, corners, texture changes) is widely used in tracking. Using OpenCV's salient-feature detection tools, areas of interest were detected and marked with red crosses as shown in **Fig. 2.**

Estimating Motion Between Consecutive Frames in a Video Sequence Using Optic Flow Functions

Optic Flow (OF) works by mapping pixel displacements across a sequence of frames into vectors, thus tracking motion.[25] OpenCV[25] provides an OF tracking function based on the Lucas-Kanade (LK) algorithm. The LK[26] system uses the spatial intensity gradient in images to find a good pixel match across frames. The LK algorithm has served as a building block for many systems while also remaining relevant as a stand-alone algorithm.[27]

Performing the OpenCV's OF-LK function on the previously detected salient features, an Essential Matrix (E) from the video is extracted. E is calculated using the Camera Matrix K and the corresponding matched feature points. K describes camera parameters, providing information about how 3D points are projected onto the 2D image plane. K is represented as:

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

where f_x and f_y are the focal lengths in the x and y directions, indicating the dimensions of the scene. The center point coordinates c_x and c_y set up the origin of the image plane in the camera's coordinate system. The skew value s allows for compensation for a known skew of the camera sensor used; except for older cameras, this is usually 0, meaning the image axes are orthogonal. The last row of the matrix is a constant representing a homogeneous coordinate.

K provides essential information about how 3D points are projected onto the 2D image plane, enabling the tracking system to solve for the camera's motion. In E, using OpenCV's OF-LK algorithm, the feature points detected are assessed in the context of K to determine the displacement between corresponding points in the sequence of frames.

From E, the rotation and translation information of the camera are extracted by estimating the relative camera pose between frames. The translation vectors extracted for each frame are then integrated over time to obtain the camera trajectory in x, y, z coordinates. Euler Angles are also extracted from the rotation matrix to accurately estimate the camera's pose. Euler angles provide the orientation of an object in 3D space relative to the initial frame of the video, in the form of *roll*, *pitch*, *yaw*, each corresponding to a rotation on the x,y, or z axis.[28]

These translation and rotation coordinates are in a format that UE can read to create a moving camera in a virtual environment. The motion, however, is jagged and abrupt. This is because the integration of the vectors is performed one frame at a time rather than considering the motion of the entire video. A running average[29] is implemented to smooth the motion.

Workflow to Test the Tracking Systems

The workflow designed to test the tracking systems under different video parameters is broken down in **Fig. 3.**

1545-0279/26©2026SMPTE

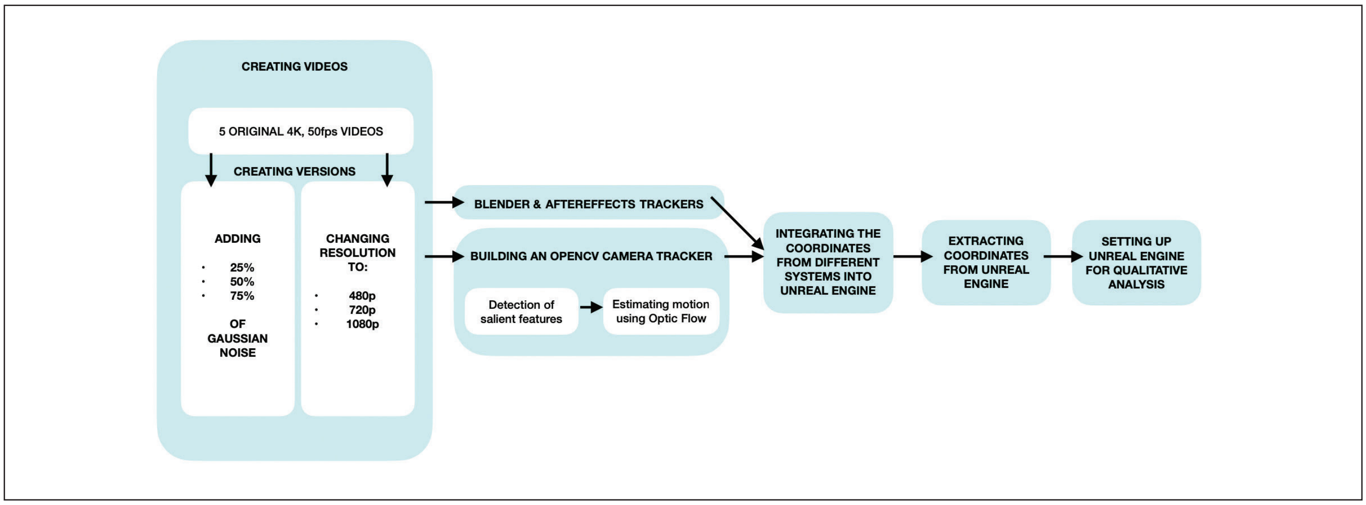
CREATING VIDEOS
5 ORIGINAL 4K, 50fps VIDEOS
CREATING VERSIONS
ADDING
25%
50%
75%
OF GAUSSIAN NOISE
CHANGING RESOLUTION TO:
480p
720p
1080p
BLENDER & AFTEREFFECTS TRACKERS
BUILDING AN OPENCV CAMERA TRACKER
Detection of salient features
Estimating motion using Optic Flow
INTEGRATING THE COORDINATES FROM DIFFERENT SYSTEMS INTO UNREAL ENGINE
EXTRACTING COORDINATES FROM UNREAL ENGINE
SETTING UP UNREAL ENGINE FOR QUALITATIVE ANALYSIS

FIGURE 3. Overview of the workflow followed by this research.

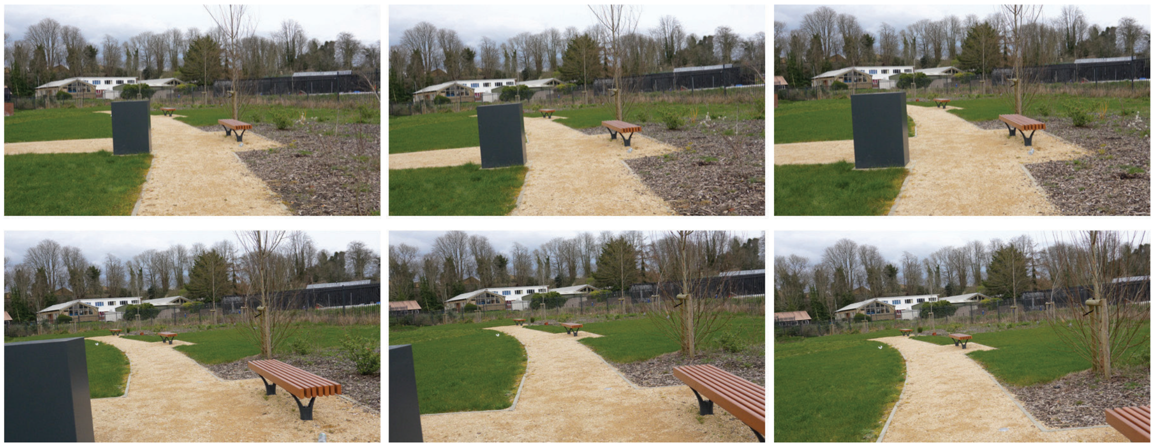

FIGURE 4. Frames taken from one of the videos in the dataset. All videos followed the same movement progression, starting with the top-left frame and ending with the bottom-right frame. The motion was a hand-held walk down a path.

First, the videos were created and versioned for testing. Then, the coordinates obtained from the different trackers were integrated into UE and exported in the proper format. Finally, the tracking systems were analyzed individually and in comparison, and the impact of different video parameters was assessed qualitatively in UE.

Creating the Videos

Five videos were created as a test dataset for the experiments. Each video was taken with a duration of around 10 sec and had the same motion of walking down a street. All videos were recorded at 4K (3840 × 2160 pixels) so that the resolution could be degraded.

Figure 4 illustrates the motion of the videos, walking in a straight line down a path.

Using AfterEffects, Gaussian Noise[30] was added to each video at 25%, 50%, and 75%. This means that for each percentage, the same proportion of random pixels in the video was selected and disturbed with random motion. Similarly, using Adobe's Media Encoder, versions of each video were created in resolutions of 1080p (1920 × 1080 pixels), 720p (1280 × 720 pixels), and 480p (640 × 480 pixels).

Both noise and lower resolution degrade tracking systems, as they can affect areas of high contrast, such as edges, corners, or changes in texture, which are typically used as salient features to track across frames. The aim of creating these versions is to analyze the impact of these parameters on tracking performance, both quantitatively and qualitatively. This determines the democratization of this technology to the broader public, which might not have high-end equipment at their disposal, but rather lower-end, noise-prone, and lower-resolution equipment.

Figures 5 & 6 illustrate different versions of a video created at varying levels of Gaussian Noise and other resolutions.

Existing Camera Trackers

While the proprietary nature of AfterEffects[16] and Blender[17,18] makes details on the inner workings of their tracking tools difficult to obtain, the available documentation describes models based on salient feature detection and optic flow tracking.

In Blender, the first frame of the video is analyzed to de-

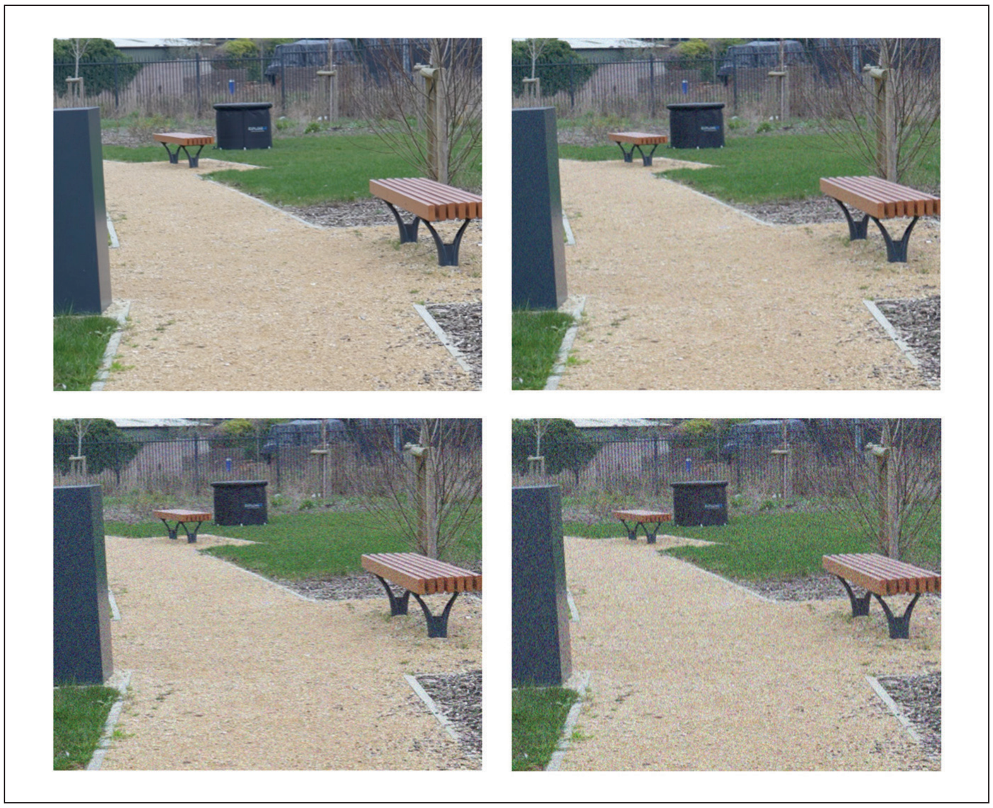

FIGURE 5. The zoomed-in and cropped first frame of each version of a video with varying amounts of noise. On the top left, the original video with no added noise. In the top-right corner, Gaussian Noise was added to 25% of the pixels in the video. On the bottom left, Gaussian Noise was added to 50% of the pixels in the video. In the bottom-right corner, Gaussian Noise was added to 75% of the pixels in the video.

FIGURE 6. The first frame of each version of a video with varying resolution. From left to right, resolutions decrease from the original 4K to 1080p, then 720p, and lastly 480p.

tect salient features, as shown in **Fig. 7.** The video is then analyzed, tracking the motion of each marked feature throughout.[17,18]

Likewise, AfterEffects[16] analyzes the first frame for salient features and then tracks across the video.

Understanding Differences in Coordinate Systems

While the trackers (OpenCV, Blender, and AfterEffects) render coordinates in the same format as UE, integrating them into the virtual environment remains challenging due to differences in Cartesian coordinate alignment.

UE uses a left-handed, Z-axis up system. Very few systems follow this alignment, so when translating coordinates from these other systems to UE, a one-to-one *x,y,z* copy will cause errors.

In OpenCV,[32] the camera's principal axis is aligned with the Z-axis of the coordinate system and follows the right-hand rule (where the Y-axis is oriented downward). This means *x, y, z* coordinates must be converted to *-z, -x, y* to align with the coordinate system in UE. An API capable of communicating with UE can be imported into Python to enable the OpenCV tracker to communicate with UE. The workflow for

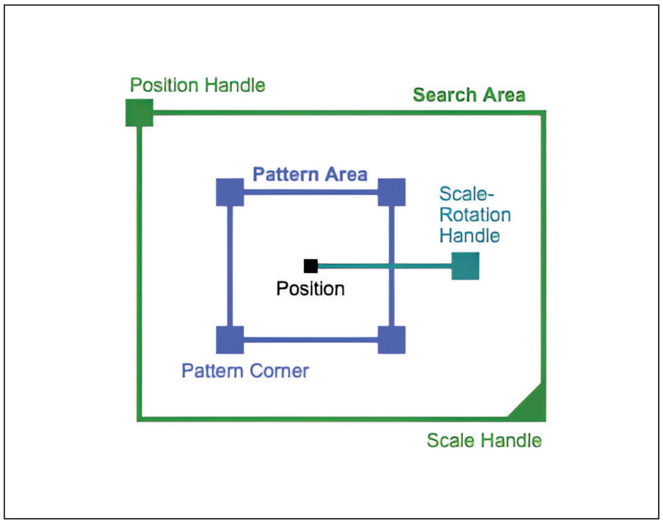

FIGURE 7. Blender Tracking Marker Schematics.[31]

obtaining camera coordinates from the OpenCV tracker to UE is shown in **Fig. 8.**

Likewise, Blender uses the right-hand rule, but with Z pointing upward, a mirror image of the left-hand rule used by UE. AfterEffects exports the coordinates using the left-hand rule, like UE, but with the Y-axis pointing upwards instead of the Z-axis.

Extracting coordinates from UE to Equalize Coordinates Across the Different Tracking Systems

For the AfterEffects and Blender tracking systems, detecting different features between videos means the selected *ground planes/floor* and *origins used* to orient the tracking might vary. Therefore, to make a fair comparison of the different tracking systems, the coordinates for each tracker were extracted from UE, offset to begin at the origin (0,0,0), and new machine-readable CSV files were created with these equalized values.

Setting up Unreal Engine for a Qualitative Analysis of the Performance of Tracking Systems

For the qualitative analysis of the tracking systems and the effect of different video parameters on tracking performance, a virtual scene is created in UE that includes elements matched to those in the source videos. As shown in **Fig. 9**, the right-hand trees are matched to the benches in the original scene from the source video. This aids in comparing the fluidity of movement and the spatial accuracy of the track.

The scaling for the different tracking systems had to be adjusted proportionally to the magnitude of the virtual scene to ensure the track matched the source videos in pace and displacement. All tracks were then exported from the UE sequencer and compared with the original videos to qualitatively assess the impact of different video parameters on tracking. This further assesses the democratization of the tracking technology by examining the optimal parameters for the seamless replication of motion in a virtual environment.

Results and Analysis

To determine whether a tracker built using OpenCV can meet the same standard as the other two trackers, their performances must be compared. **Tables 1 and 2** show the mean error per frame and the standard deviation of tracking for the five videos, compared with tracking of versions of the same videos at different resolutions and with varying percentages of Gaussian noise added.

After inspecting the results affected by noise in **Table 1,**

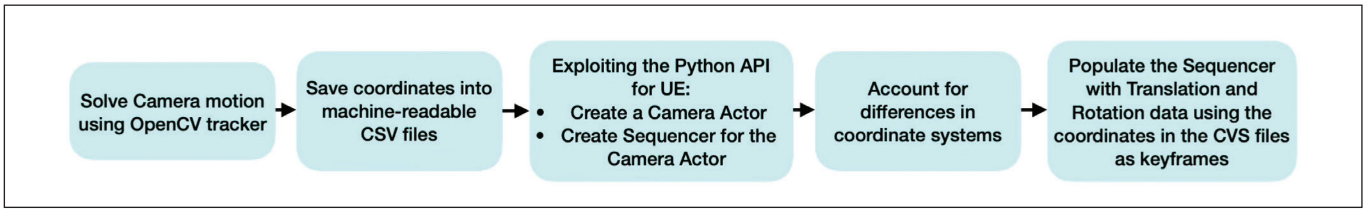
Solve Camera motion using OpenCV tracker
Save coordinates into machine-readable CSV files
Exploiting the Python API for UE:
• Create a Camera Actor
• Create Sequencer for the Camera Actor
Account for differences in coordinate systems
Populate the Sequencer with Translation and Rotation data using the coordinates in the CVS files as keyframes

FIGURE 8. Steps taken to successfully plot coordinates obtained from the OpenCV tracker into Unreal Engine.

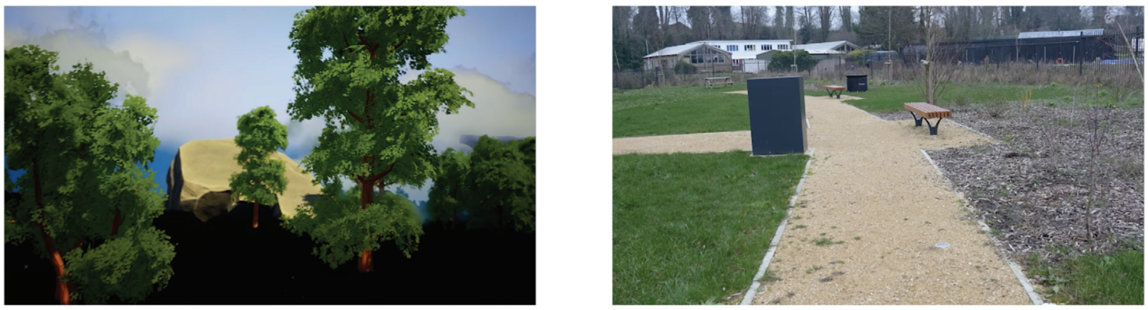

FIGURE 9. Example comparison of a source video (right) and a virtual scene with matching elements (left) to compare the tracking performance.

TABLE 1. Average Impact of Noise on the Performance of the Trackers.

Tracker	% of Gaussian Noise added to video	Average Mean Error (pixels) across the dataset	Average Standard Deviation (pixels) across the dataset videos
OpenCV	25	16.7	7.96
	50	18.8	8.02
	75	16.2	10.41
Blender	25	100.7	66.5
	50	90.54	54.4
	75	112.8	75.5
AfterEffects	25	6273.28	3985.2
	50	8574.82	5807.98
	75	10473.43	7231.93

TABLE 2. Average Impact of Resolution on the Performance of the Trackers.

Tracker	Compared Video Resolution	Average Mean Error (pixels) across the dataset	Average Standard Deviation (pixels) across the dataset
OpenCV	480 SD	105.31	62.04
	720 HD	99.94	60.16
	1080 HD	55.42	27.5
Blender	480 SD	114.62	67.46
	720 HD	94.24	62.96
	1080 HD	100.68	64.06
AfterEffects	480 SD	8258.12	5476.3
	720 HD	6968.68	4635.16
	1080 HD	4529.96	3001.42

the OpenCV tracker achieved the lowest error across the dataset. The mean error per frame for the OpenCV system was the lowest and the least spread across the three noise percentages. This average error is more pronounced in the standard deviation of the tracking relative to the original, as predicted, with the track's deviation increasing with the percentage of noise added to the videos. Even so, the OpenCV tracker continues to outperform the other two trackers in this aspect, too, with the average deviation from the original videos still not surpassing ±2 pixels in coordinates between noise percentage.

The tracking system from Blender also follows the trend of, on average, the highest percentage of noise rendering the most deviation from the original track. The average mean error per frame is much higher than that of the OpenCV tracker, and the differences across the three noise levels exceed ±10 pixels in coordinates, indicating greater sensitivity to noise in the tracker's performance.

The AfterEffects tracker has the poorest performance overall. Its average mean error per frame across all noise percentages is over 100 times that of the OpenCV tracker and over 10 times that of the Blender tracker. This, however, might be due to different scaling and coordinate systems. While the pattern of the greatest noise percentage rendering the most significant error remains true, the average standard deviation almost doubles between 25% and 50% noise added, indicating the most significant impact of noise on the system's tracking performance.

The results from changing resolutions in **Table 2** show similar patterns. In this case, the OpenCV tracker continues to outperform the other two systems, but its results are closer in range to those of the Blender tracker this time. For the OpenCV system, both average error per frame and standard deviation double when moving from 1080p to 720p. Results between 720p and 480p differ less noticeably, suggesting that once the tracker has reached a specific resolution, the rapid degradation of tracking performance levels out.

The progressive deterioration in the tracker's performance as resolution decreases is less apparent in the Blender system. The expected pattern is not observed, with the least error obtained from the 720p video rather than the higher-resolution 1080p. The changes between resolutions, however, amount to only ±4.5 pixels in the average standard deviation from the original videos. This would indicate that while the average error is slightly higher in the Blender tracker compared to the OpenCV tracker, the actual impact between resolutions in Blender is far less noticeable.

The AfterEffects tracker behavior seems to resemble the OpenCV system more closely. The expected pattern of lower resolution negatively impacting the track's performance is observed. Unlike the OpenCV tracker, the impact of resolution on this system's performance is more evenly distributed, with approximately equal intervals (±2000 pixels) of increasing average error as resolution decreases.

To further analyze the three systems, more specific examples can be examined.

OpenCV Tracker

The following examples are representative of the results obtained across the dataset using the OpenCV tracker and are also proportional to those of the other tracking systems.

1545-0279/26©2026SMPTE

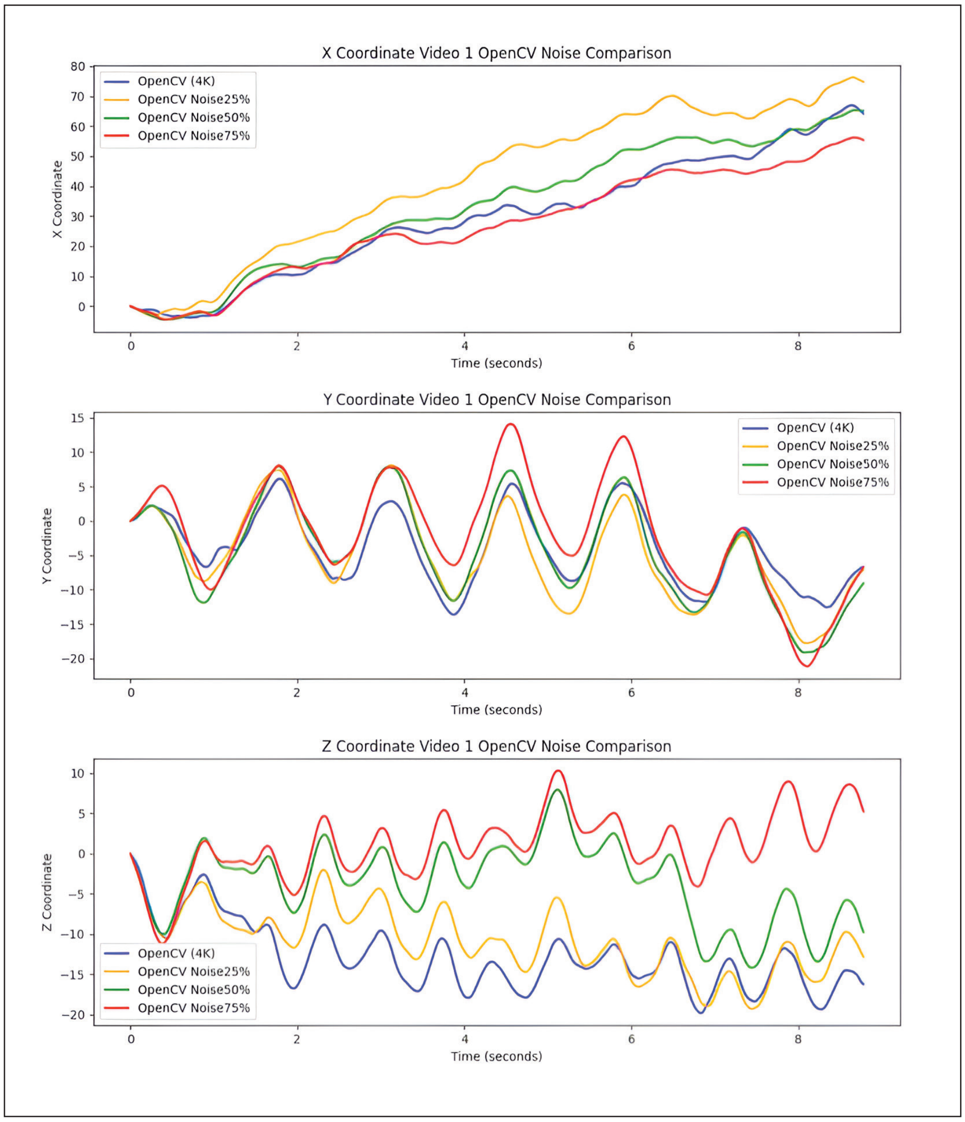

FIGURE 10. Graphical representation of the effect Noise % had on *x*,*y*, and *z* coordinates for Video 1. The original 4K video is drawn in blue. Noise levels cause deviations from the original values, thus degrading the track's accuracy.

Taking Video 1 as an example, **Table 3** and **Fig. 10** illustrate the OpenCV system's performance under varying noise levels. The tracker maintains stable performance in the presence of noise, with the mean error remaining under 5 pixels across noise levels. The *z*-coordinate values are most affected at noise levels of 50% and 75%, changing from negative to positive, signaling a change in the direction of motion relative to the source.

Table 3 and **Fig. 11** inspect the impact decreasing resolutions have on the OpenCV system's tracking when compared to the original Video 1. Examining the mean error per frame, the best tracking performance is achieved at 1080p, while lower resolutions yield significantly larger error values. Coordinate *x* indicates the direction of motion, while *y* and *z* refer to side-to-side and up-down motion, respectively. Where the original 4K and 1080p resolutions show a positive increase in *x*, representing motion forward, the resolutions of 720p and

TABLE 3. Performance Results of the OpenCV Tracking System on Different Noise % Versions of Video 1 Compared to the Source Video.

Standard Video	% of Gaussian Noise added to video	Mean Error (pixels)	Standard Deviation (pixels)
Video 1 4K	25	14.7	6.42
	50	10.4	3.78
	75	13.9	5.87

TABLE 4. Performance Results of the OpenCV Tracking System on Different Resolution Versions of Video 1 Compared to the Source Video.

Standard Video	Compared video Resolution	Mean Error (pixels)	Standard Deviation (pixels)
Video 1 4K	480 SD	99.0	61.1
	720 HD	71.9	42.6
	1080 HD	29.9	10.5

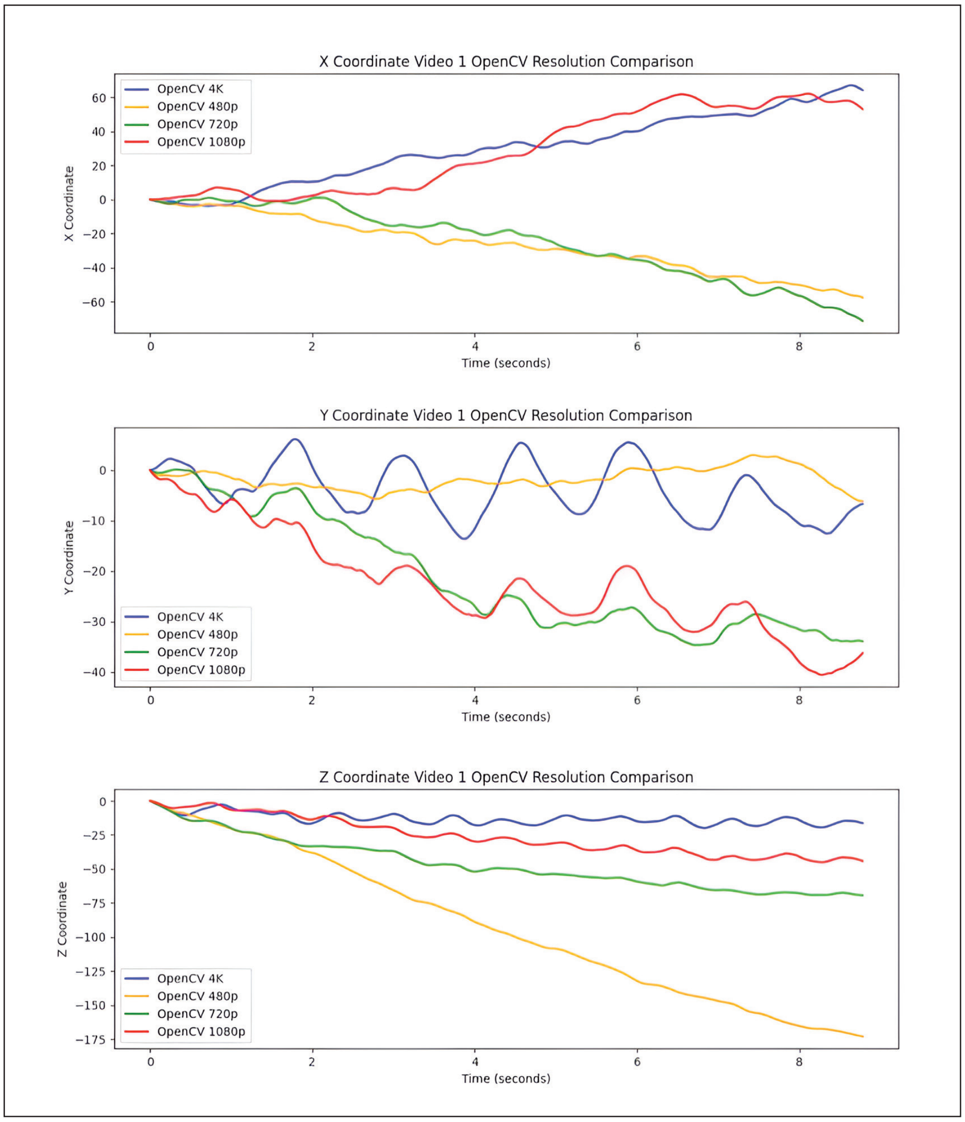

FIGURE 11. Graphical representation of the effect different Resolutions had on x,y, and z coordinates for video.

480p show a decrease in x, implying an opposite motion relative to the source.

1. The original 4K video is drawn in blue; the lower resolutions cause deviation from the original values, thus degrading the accuracy of the track.

Assessing the Qualitative Impact of Noise and Resolution in Unreal Engine

The performance of the built OpenCV tracker was also evaluated in a virtual environment built in UE.

Figure 12 establishes the expected results by showcasing how the first and last frames of the original version of Video 1 translated into the virtual environment in UE. In selecting a virtual scene, the aim was to match certain elements to the source video to better evaluate the camera's tracking by comparing their positions and distances in the beginning and end frames. In this case, the two trees on the left-hand side of the virtual environment frame were roughly matched to the two benches in similar positions from the original video.

Figures 13 & 14 show frames taken at the same times for each video version (noise-free and with 25%, 50%, and 75% added noise). These two examples are taken at seconds 2 (frame 100) and 6 (frame 300) of the 8-second video (400 frames). While there are differences throughout the versions 2 seconds in, these are much less noticeable than those from the frames 6 seconds into the video. The tracking differences worsen over time, causing the drift from the original track to increase as time goes by.

Much more drastic differences are observed when comparing the results for the 4K Video 1 track with those from the tracks performed at resolutions 1080p, 720p, and 480p.

Figure 15 illustrates the drastic negative impact of resolution on tracking performance by comparing frame 200 across all video versions (original 4K resolution, 1080p, 720p, and 480p). The disparity between frames and the

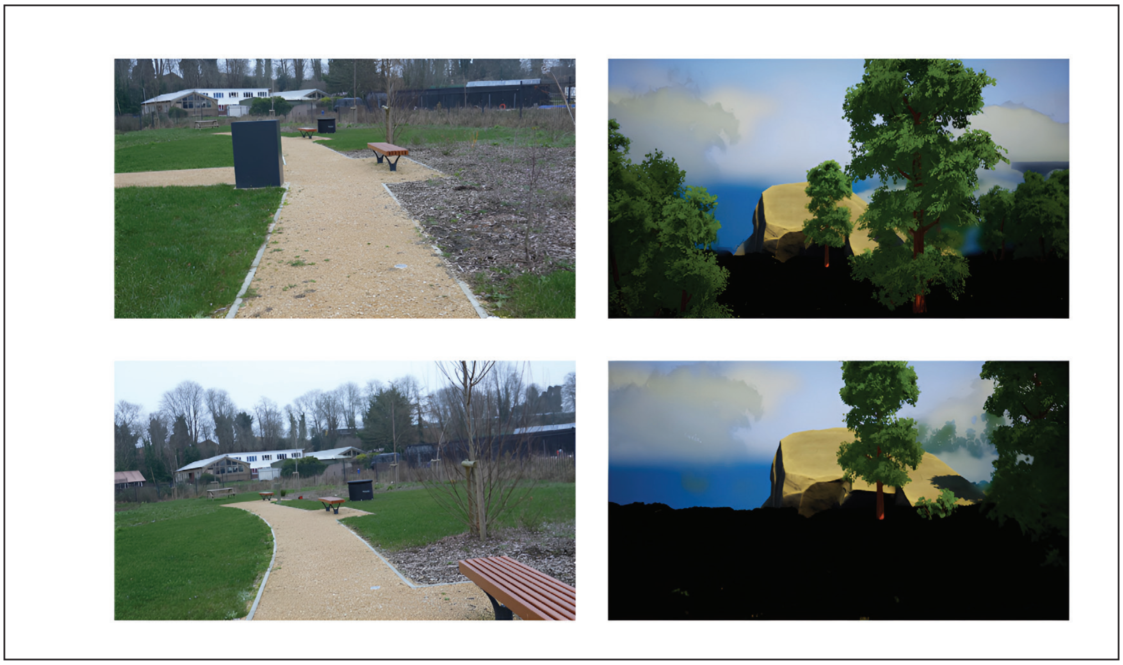

FIGURE 12. A comparison of the first frame of Video 1 (top left) with that same frame in the virtual environment created in UE (top right). On the second row, the bottom-left image shows the last frame of the video, and the bottom-right image shows it in the virtual environment.

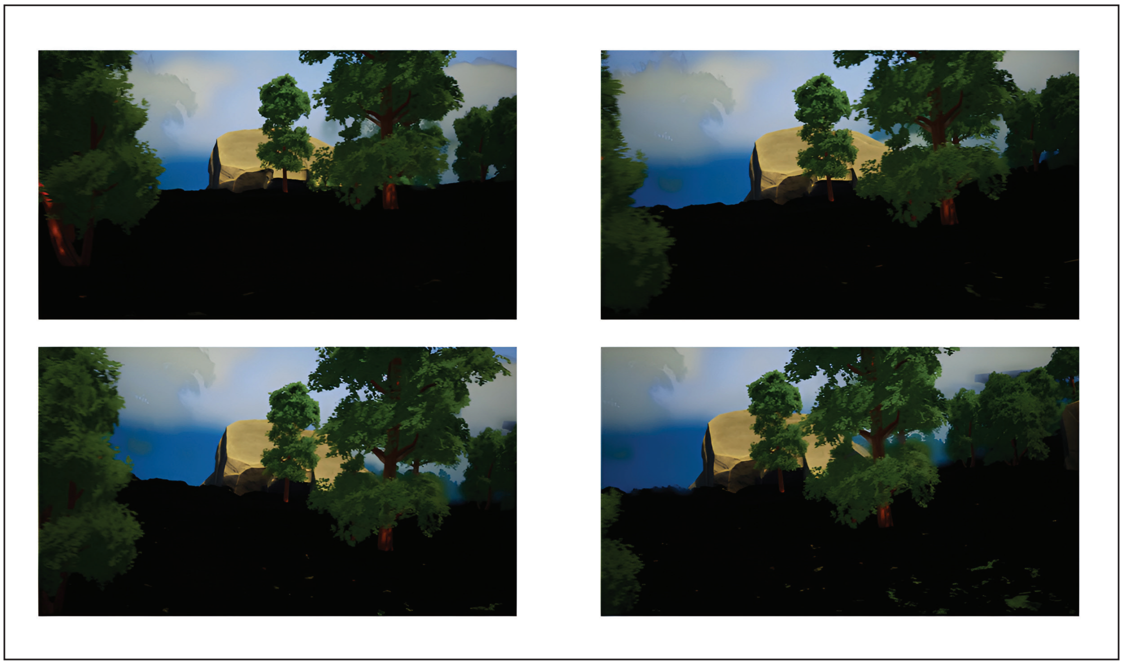

FIGURE 13. Frames taken at 2 seconds into Video 1 from the four versions. The noise-free version (top left), the version with 25% noise added before the track (top right), the version with 50% noise added before the track (bottom left), and the version with 75% noise added before the track (bottom right).

lack of resemblance to the original video suggest the drifting away from the original track is noticeable and entirely at odds with what the motion should be. The OpenCV tracker produces unusable results at resolutions below 4K.

Discussion

Both noise and resolution affect the performance of all trackers. Overall, however, trackers generally have lower error due to noise than to decreasing resolution. The OpenCV system seemed to outperform the other two trackers, consistently attaining the lowest mean error per frame and standard deviation relative to the original tracks across the dataset at different noise percentages and resolutions, followed by Blender and, lastly, AfterEffects.

The analysis of the results also matches the track's performance for each version in UE. Using Video 1 as an example, noise at any level negatively affected the motion's smoothness but preserved the original trajectory. The differences

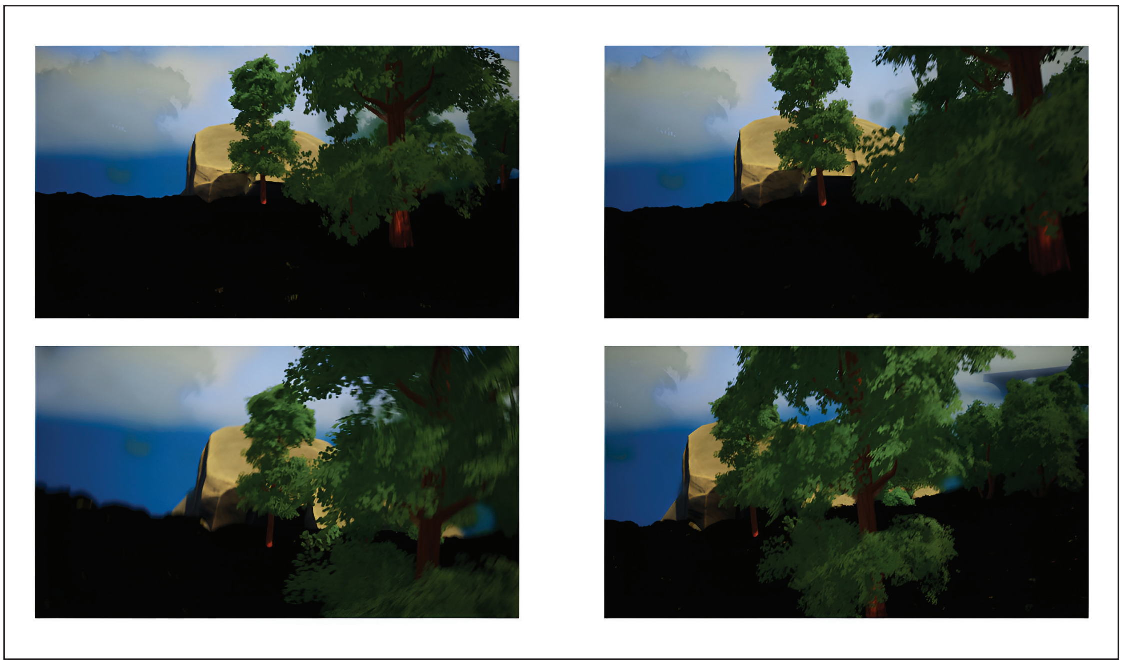

FIGURE 14. Frames taken at 6 seconds into Video 1 from the four versions. The noise-free version (top left), the version with 25% noise added before the track (top right), the version with 50% noise added before the track (bottom left), the version with 75% noise added before the track (bottom right).

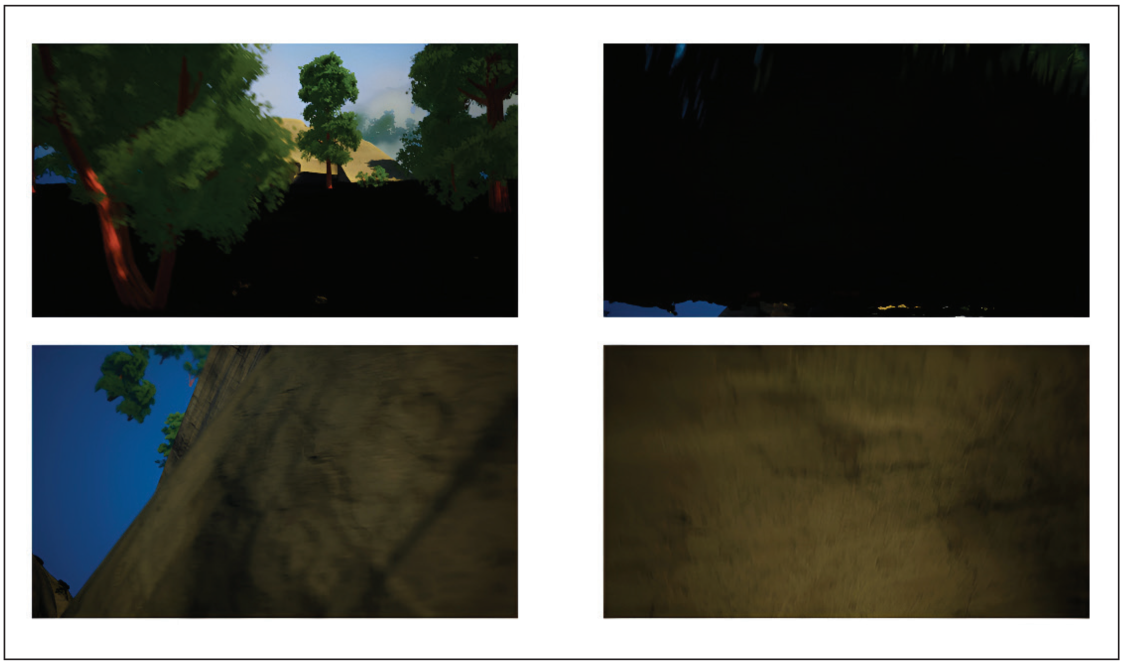

FIGURE 15. Frames taken at 4 seconds into Video 1 from the four versions. 4K version (top left), the version at resolution 1080p (top right), the version at resolution 720p (bottom left), the version at resolution 480p (bottom right). No frame is the same; the differences are noticeable, suggesting that all the tracks have drifted drastically away from the original trajectory.

in tracking between videos at different noise levels were minimal in the virtual environment. This suggests that the OpenCV tracking system is resilient to high levels of noise in a video during tracking and proves that while noise might hinder the quality of a track, the results are still usable, albeit prone to sporadic bursts of abrupt movement.

The noise-infused pixels disturb salient features, such as edges or areas of high contrast, that tracking systems latch onto to perform tracking and solve the camera motion. It appears that the tracking systems don't latch onto individual pixels as tracking points, but rather select a group of pixels that form the salient feature. This technique prevents the loss of tracking elements due to noisy pixels, since in the chosen area, the probability that all pixels are noisy is reduced (unless 100% of the pixels in a video are noisy). This would also explain why, while still having an impact on the track, the average mean-error per frame and standard deviation from the original render lower results compared to the changes in resolution. The areas selected as tracking points contain enough pixels to be tracked with almost the same success as a noiseless video.

Decreasing the resolution also worsened the trackers' performance. The OpenCV tracker seemed to be relatively the most affected by resolution. In contrast, Blender and AfterEffects both maintained error values closer to the same range as those obtained by varying amounts of noise. This is perhaps because the areas that OpenCV needs to track contain more pixels. As previously discussed, this would make them more resilient to noise but would be negatively impacted by a reduction in resolution. Lower resolutions mean fewer pixels available to display the information in the image. This disrupts the detail in salient features, and if the areas tracked by OpenCV remain fixed in size, the same number of pixels in a lower-resolution image would encompass a larger area. This would imply a loss of detail in the features to be tracked, since within the same area, there can be overlapping points of interest that introduce errors into tracking. In the AfterEffects and Blender tracking systems, the markers used, as shown in **Fig. 7**, appeared physically larger on the screen, thus appearing to encapsulate relatively larger areas, but in reality, these were scaled due to the lower resolutions. The way these two other systems compensated for this was to reduce the number of markers used, effectively selecting less salient features, to avoid confusing overlapping areas that might induce error in the tracker. This allowed these two systems to maintain their errors within a similar range to the noise, while still suffering from lower resolution.

It can be stated that both noise and resolution affect tracking, with both parameters degrading track performance. This would limit the extent of the democratization of the tracking technology. The extent of the impact of these parameters on tracking performance, however, can vary depending on the tracking system and the measures in place to limit the effect of the error.

Further Research

The following section will propose future research areas to deepen the understanding of camera tracking technologies and help overcome their limitations.

Improving the Performance of Trackers in Lower Resolutions

The results in Section 3 indicate that the OpenCV tracker struggled to a greater extent at low video resolutions than with noise, whereas the other two tracking systems appear to maintain similar error ranges across both parameters.

It was theorized that this was due to the number of pixels per tracking point used by the OpenCV system. If this amount is consistent across all resolutions, it would lead to overlapping tracking points, rendering contradictory analyses of motion. Therefore, following the systems of the other two trackers, one possible solution to this issue is to develop a model in which the number of tracking points detected and used decreases with resolution, thus avoiding the damaging overlap.

Jain *et al.*[33] found similar issues in their research, where lower video resolutions increased the system's processing rate but reduced track accuracy. To overcome this, Svanström *et al.*[34] found that tracking with visible thermal and acoustic markers yielded good performance even at lower resolutions. While marker-based tracking systems are more expensive to set up and maintain and thus less accessible to the broader public, it would be beneficial to explore this technology to better understand how the problem of resolution might be addressed.

Investigating the democratization of Real-time Camera Tracking Technologies

It would be interesting to expand the scope of this research further by assessing the democratization of real-time camera tracking techniques. The best approach might be to examine the implementation of Simultaneous Localization and Mapping (SLAM) techniques, which create a map of the camera's environment as it tracks its own position within it.

As proposed by Salas *et al.*,[35] the use of 3D object detection and recognition, rather than just salient features, can be paired with SLAM models to improve the mapping and tracking of elements in a virtual scene in real-time. This approach would, however, be better suited to scenes with repeated elements, which might not always be the case in general public use. Ullah *et al.*[36] suggest a different approach using Kalman Filters, for which OpenCV provides dedicated tools. This would allow the camera's localization to be more accurately determined in unknown environments.

Unfortunately, these systems often use Deep Neural Networks (DNNs), which require large-scale datasets and thus fall beyond the scope of what is available to the general public. As the research in this area progresses, however, it might be possible to adapt these systems into less computationally demanding models, thereby further democratizing camera tracking technology.

Conclusion

Computer vision is an ever-growing field with many useful applications. By analyzing, processing, and solving the motion in a video, camera tracking techniques can be developed for VR, AR, and VP. This inspires continued research in the field to understand better and manipulate the technology. This project aimed to determine the full range of the democratization of this technology by posing three objectives.

The results illustrate the success of the tracker built with open-source tools in competing with industry-standard tools. To determine the optimal parameters for noise and resolution to achieve good tracking performance, the results show that both noise and resolution hinder the efficiency of the tracking systems. Further inspection revealed that, in general, tracking systems could still produce useful tracking results in the presence of noise, but struggled quite acutely at lower resolutions. Therefore, this limits the democratization of tracking technology to equipment available to the broader public, which can achieve higher resolutions. Lastly, the practical impact of noise and resolution in a virtual scenario was assessed. This examination further determined that while videos affected by noise could still render usable outcomes in virtual scenes, those with lower resolutions struggled to perform well.

In conclusion, while there are still limitations to the full democratization of camera tracking systems, tools and resources are available for the public to experiment and push the boundaries of small-scale productions. Continued research into tracking systems will further encourage their development and aid their democratization.

References

1. X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li, "Computer Vision Algorithms and Hardware Implementations: A Survey," *Integration, the VLSI Journal*, 69 (11): 309–320, Nov. 2019, doi:10.1016/j.vlsi.2019.07.005.
2. Y. Çelik, M. Altun, and M. Güneş, "Color-Based Moving Object Tracking With an Active Camera Using Motion Information," *Proc. Int. Artif. Intell. Data Process. Symp. (IDAP)*, pp. 1-5, Sep. 2017.
3. B. Chen, P. Li, C. Sun, D. Wang, G. Yang, and H. Lu, "Multi-Attention Module for Visual Tracking," *Pattern Recognit.*, 87: 80–93, Mar. 2019, doi:10.1016/j.patcog.2018.10.005.
4. A. Pareek and N. Arora, "Re-Projected SURF Features-Based Mean-Shift Algorithm for Visual Tracking," *Procedia Comput. Sci.*, 167: 1553–1560, 2020, doi:10.1016/j.procs.2020.03.366.
5. H. Zhou, B. Ummenhofer, and T. Brox, "DeepTAM: Deep Tracking and Mapping With Convolutional Neural Networks," *Int. J. Comput. Vis.*, 127(3): 411–430, Mar. 2019.
6. J. Zhan, H. Zhao, P. Zheng, H. Wu, and L. Wang, "Salient Superpixel Visual Tracking With Graph Model and Iterative Segmentation," *Cogn. Comput.*, 13 (4): 848–863, Aug. 2021.
7. A. Handa, R. Newcombe, A. Angeli, and D. Davidson, "Real-Time Camera Tracking: When Is High Frame-Rate Best?" *Computer Vision – ECCV 2012*, A. Fitzgibbon et al., Eds., Springer, Lecture Notes in *Comput. Sci.,* vol. 7573, pp. 300–315, Oct. 2012.
8. G. Younes, D. Asmar, E. Shammas, and J. Zelek, "Keyframe-Based Monocular SLAM: Design, Survey, and Future Directions," *Robot. Auton. Syst.*, (97): 1–22, Jan. 2017.
9. Y. Genc, S. Riedel, F. Souvannavong, C. Akinlar, and N. Navab, "Marker-Less Tracking for AR: A Learning-Based Approach," *Proc. Int. Symp. Mixed Augmented Reality (ISMAR)*, pp. 1–10, Oct. 2002.
10. K. Malvika and S. Malathi, "Insights Into the Impactful Usage of Virtual Reality for End Users," *Proc. Fourth Int. Conf. I-SMAC (IoT Soc. Mobile Analyt. Cloud)*, pp. 123-128, Oct. 2020.
11. T. A. Syed, M. S. Siddiqui, H. B. Abdullah, S. Jan, A. Namoun, A. Alzahrani, A. Nadeem, and A. Alkhodre, "In-Depth Review of Augmented Reality: Tracking Technologies, Development Tools, AR Displays, Collaborative AR, and Security Concerns," *Sensors*, Vol. 23, 2023.
12. C. Okwuowulu and C. Michael, "A Technical Report on the Virtual Production of Marginalised Areas in the Pandemic Era," *Nigerian Theatre J.*, Vol. 21 (1):1–15, 2022.
13. S. Kamat, *Atmospheres of Change: Virtual Production*, Ph.D. dissertation, Massachusetts Institute of Technology: Cambridge, MA, 2022.
14. P. Nogueira, "Motion Capture Fundamentals," *Proc. Doctoral Symp. Informatics Eng.*, pp. 1–6. Sep. 2011.
15. J. Meza, L. Romero, and A. Marrugo, "MarkerPose: Robust Real-Time Planar Target Tracking for Accurate Stereo Pose Estimation," *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 1–10, Jun. 2021.
16. Adobe, "Tracking 3D Camera Movement," *Adobe After Effects Help*, Nov. 21, 2023. [Online]. Available: https://helpx.adobe.com/after-effects/using/tracking-3d-camera-movement.html
17. Blender Foundation, "Track," *Blender Manual*, online documentation, 2023. Accessed Nov. 24, 2023). [Online]. Available: https://docs.blender.org/manual/en/latest/movie_clip/tracking/clip/editing/track.html
18. Blender Foundation, "Motion Tracking—Introduction," *Blender Manual*, online documentation, 2023. [Online]. Available: https://docs.blender.org/manual/en/latest/movie_clip/tracking/introduction.html (accessed Nov. 24, 2023).
19. N. Kadner, *The Virtual Production Field Guide*, Epic Games: Cary, NC, 2019.
20. A. Chia, B. Keogh, D. Leorke, and B. Nicoll, "Platformisation in Game Development," *Internet Policy Rev.*, vol. 9 (4): pp. 1–22, Dec. 2020.
21. OpenCV Team, "About OpenCV," *OpenCV*. Accessed Feb. 20, 2024. [Online]. Available: https://opencv.org/about/
22. T. Ylänne, "3D Camera Tracking for Low-Budget Production," M.S. thesis, Helsinki Metropolia Univ. Appl. Sci., Helsinki, Finland, 2011.
23. D. Terzopoulos, "Perceptive Agents and Systems in Virtual Reality," *Proc. ACM Symp. Virtual Reality Software and Technology (VRST)*, pp. 1–10. Oct. 2003.
24. V. Gaur, "Lucas–Kanade Optical Flow Machine Learning Implementations," *J. Student Res.*, 11 (3), Aug. 2022.
25. OpenCV, "Optical Flow," *OpenCV Documentation*, online documentation. Accessed Feb. 20, 2024. [Online]. Available: https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html
26. B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique With an Application to Stereo Vision," *Proc. Int. Joint Conf. Artificial Intelligence (IJCAI)*, pp. 674–679, Aug. 1981.
27. L. Mendes, A. Ricardo, A. Bernardino, and R. Ferreira, "A Comparative Study of Optical Flow Methods for Fluid Mechanics," *Exp. Fluids*, Vol. 63, 2022.
28. E. W. Weisstein, "Euler Angles," *MathWorld—A Wolfram Web Resource*, 2009. Accessed Feb. 20, 2024. [Online]. Available: https://mathworld.wolfram.com/
29. A. Sivakumar, K. Shaw, and D. Pathak, "Robotic Telekinesis: Learning a Robotic Hand Imitator by Watching Humans on YouTube," *arXiv* preprint arXiv:2202.10448, 2022.
30. M. Stoiber, M. Pfanne, K. Strobl, R. Triebel, and A. Albu-Schäffer, "A Sparse Gaussian Approach to Region-Based 6DoF Object Tracking," *Proc. Asian Conf. Computer Vision (ACCV)*, pp. 666-682, Nov.-Dec. 2020.
31. Blender Foundation, "Tracking Marker," *Blender Manual*, online documentation, 2023. Accessed Nov. 24, 2023. [Online]. Available: https://docs.blender.org/manual/en/latest/movie_clip/tracking/clip/marker.html
32. S. Condino, G. Turini, V. Mamone, P. Parchi, and V. Ferrari, "Hybrid Spine Simulator Prototype for X-Ray-Free Pedicle Screws Fixation Training," *Appl. Sci.*, Vol. 11, 2021.
33. S. Jain, X. Zhang, Y. Zhou, G. Ananthanarayanan, J. Jiang, Y. Shu, P. Bahl, and J. Gonzalez, "Spatula: Efficient Cross-Camera Video Analytics on Large Camera Networks," *IEEE Trans. Mobile Comput.*, 2020.
34. F. Svanström, C. Englund and F. Alonso-Fernandez, "Real-Time Drone Detection and Tracking With Visible, Thermal and Acoustic Sensors," *2020 25th International Conference on Pattern Recognition (ICPR)*, Milan, Italy, pp. 7265-7272, 2021, doi: 10.1109/ICPR48806.2021.9413241.
35. R. Salas-Moreno, R. Newcombe, H. Strasdat, P. Kelly, and A. J. Davison, "SLAM++: Simultaneous Localization and Mapping at the Level of Objects," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 1352–1359, Jun. 2013.
36. I. Ullah, X. Su, X. Zhang, and D. Choi, "Simultaneous Localization and Mapping Based on Kalman Filter and Extended Kalman Filter," *Wireless Commun. Mobile Comput.*, (1): 1-12, Jun. 2020.

Supplementary Materials:

- All code can be accessed at this GitHub repository: Camera-Tracking-Systems-and-their-Democratisation [Online]. Available: https://github.com/irenem-lresearch-rgb/Camera-Tracking-Systems-and-their-Democratisation?tab=readme-ov-file

- For example, videos of the tracking output follow this: [Online]. Available: https://drive.google.com/drive/folders/1cseqY5Spty1Kzi8FG00IxmJGFAsL__Nf

About the Authors

Irene Muñoz López is a film production and broadcast engineering graduate. She specializes in virtual production, blending creativity and technical skill, camera tracking, and award-winning directing. She is currently working on her PhD in AI for Accessibility in Media at the University of Surrey.

Andrew Gilbert is an Associate Professor at the University of Surrey, where he co-leads the Centre for Creative Arts and Technologies (C-CATS).

This paper received the 2024 SMPTE Student Paper Award. Copyright © 2026 SMPTE.

DOI: 10.5594/JMI.2026/YGPF6760
Date of publication: 26 January 2026