Adapting a method for tracking the movement of the camera in the visualization of augmented reality

Miroslav Galabov¹, Yulian Pastarmov²

¹ “St. Cyril and St. Methodius” University of Veliko Tarnovo, Veliko Tarnovo, Bulgaria
² Google Inc., Zürich, Switzerland

e-mail: lexcom@abv.bg

Abstract. This paper presents an adapted method for tracking camera motion in the visualization of augmented reality. A tracking algorithm based on significant points and descriptors without the use of optical flow is described. The tracking module is written in C++. Experiments have been carried out of the precision of the system using the camera motion-tracking module created.

1. Introduction

Virtual reality (VR) is a computer-generated simulation of a three-dimensional environment in which the user has the opportunity to interact with the content of the modeled scene. The main idea is to be an integral part of the environment and to have an impact on it in the same way as in the real world. Integrating the real word the virtual world provides us with the best solutions for qualitatively adopting new knowledge. When the object is far away, the light traveling to one eye is parallel to the light to the other eye. When the subject is approaching, these lines are no longer parallel, but converge, and the eyes must compensate for the difference in distance by focusing. When focusing on an object, the brain calculates the effort needed to focus. This information ultimately gives an idea of how far the object is.

The virtual reality could be totally unrelated to reality and reality arranged in a new, imaginary nonexistent way. It can also recreate places, physically distant from the observer. In this way, it allows them to be transported to another world or place in reality. Immersion in virtual reality takes place through various input/output devices on the computer. Its level of realism depends on the choice of technical means and the quality of the work done on the virtual work.

VR immerses the user by making them feel like they are experiencing the simulated reality firsthand, primarily by stimulating their vision and hearing. VR is typically achieved by wearing a headset equipped with the technology, and is used prominently in two different ways:
• To create and enhance an imaginary reality for gaming, entertainment, and play (Such as video and computer games, or 3D movies, head mounted display).
• To enhance training for real life environments by creating a simulation where people can practice before hand (Such as flight simulators for pilots).

An intermediate step between actual reality and the virtual world is an augmented reality. Augmented reality (AR) is a technology that layers computer-generated enhancements on the top an existing reality in order to make it more meaningful through the ability to interact with it. AR is developed into apps and used on mobile devices to blend digital components into the real world in such a way that they enhance one another, but can also be discerned easily.
The primary means by which the virtual reality is formed is the digital camera. The parameters determining the quality of digital cameras are the number of sensitized sensor cells, more specifically called its resolution, the focus range of the lens. The latter, meaning the minimum and maximum distance that objects placed in front of the camera can be focused on by their optics. These two parameters are independent of each other and can be controlled individually by combining the appropriate cameras and lenses. As a result of their interaction choice the viewing angle of the camera is also obtained.

2. Determining the position of the camera in the space and geometry of the scene
Since ancient times, scientists have found that light passing through a very small aperture exhibits properties similar to those when focused through lens. Thus, if a screen is placed at some distance behind the aperture, what is in front of the aperture will be visible on it, as long as it is bright enough.

The same would happen if a lens system is used in a telescope or binoculars. It turns out that in this case the light behaves in quite a similar way to what is used in digital camera, for which the lens field of view does not exceed 100 °. This model is known in the literature as a “pinhole camera” or miniature aperture camera. The model is shown schematically in fig. 1.

![Geometric camera model](image)

**Figure 1.** Geometric camera model.

In this model, it is assumed that the optical system of the lens is capable of focusing light at one point - optical center or focus. In order to avoid working with a mirror image, it is assumed that the image obtained from the camera is the one that which would be visible on a virtual projective plane symmetrically arranged along the optical axis at the same distance as the real plane, as shown in fig. 1 (a). The model thus constructed [1] has one specific property that makes it particularly suitable for the purpose of three-dimensional reconstruction: the design of points from space on the image can be described by linear transformations over homogeneous coordinates. Fig. 1 (b) shows the relationship between the normalized projective coordinate system \( I \) and the image plane in pixel coordinates \( p \) with non-square pixels of size \( dx, dy \).

In this model, the camera is located in the center of its own coordinate system \( c \) with the optical axis along the z-axis. At distance \( f = l \) (for the normalized model) the projecting plane is located, on which the image of the camera is projected. It uses the two-dimensional coordinate system \( I \), centered at the point of intersection of the plane and the optical axis, and directions parallel to the x and y axes of the chamber coordinate system \( c \). The projection of the point \( m_c \), expressed in system \( c \) on the plane, is obtained by constructing a line with a starting point \( c \) through \( m_c \) to determine the intersection point \( m_n \) with the projective plane. If we express the points \( m_c \) and \( m_n \) as homogeneous coordinates, the projective equation can be written in matrix form:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1
\end{bmatrix}\begin{bmatrix}
x_c \\
y_c \\
z_c \\
1
\end{bmatrix} = \begin{bmatrix}
x_n \\
y_n \\
z_n \\
1
\end{bmatrix}
\]

**\( (1) \)**
In order to relate the normalized camera model to a real device, it should be taken into account that the focal length expressed in meters should be \( f \neq 1 \). Also that the projective plane is usually represented as a set of discrete points, called pixels, which are assumed to be located in a regular grid but are not necessarily square but diamonds, as shown in fig. 1 (b). Pixel irregularity is due to inaccuracies in the production process. The pixel size itself is denoted by \( dx \) and \( dy \), expressed in meters. The 'twist' angle of the pixels is indicated by \( \alpha \). Usually, the pixel coordinates start from the upper left corner of the field of view in the projective plane \( p \). With these designations, the formula for designing points from the metric coordinate system \( c \) to the pixel plane coordinate system \( p \) can be deduced [1]:

\[
\begin{bmatrix}
    x_p \\
    y_p \\
    1
\end{bmatrix} =
\begin{bmatrix}
    f & \frac{f \tan \alpha}{d_x} & h_x \\
    0 & \frac{f}{d_y} & h_y \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x_c \\
    y_c \\
    1
\end{bmatrix}
\]

(2)

Thus derived, the matrix \( K \) is called the matrix of internal parameters of the camera, and its parameters are abbreviated as:

\[
K =
\begin{bmatrix}
    f_x & s_{\alpha} & h_x \\
    0 & f_y & h_y \\
    0 & 0 & 1
\end{bmatrix}
\]

(3)

Often, in practice, the angle \( \alpha \) is assumed to be zero and this greatly simplifies the equation (2).

The so far made model makes it possible to project points, expressed in the camera coordinate system, on the projection plane of the camera. In practice, the points are usually represented in a global coordinate system \( w \), which is independent of the camera position. The connection between these two coordinate systems depends on the relative position \( c_w \) of the camera in \( w \) and the rotation required for them to have collinear axes \( R_{cw} \). Thus, points represented by their coordinates in \( w \) can be represented in \( c \) by the following equation:

\[
m_c = R_{cw}(m_w - c_w)
\]

(4)

In homogeneous coordinates equation (4) becomes:

\[
\begin{bmatrix}
    x_c \\
    y_c \\
    z_c \\
    1
\end{bmatrix} =
\begin{bmatrix}
    R_{cw} & w_c \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    x_w \\
    y_w \\
    z_w \\
    1
\end{bmatrix}
\]

(5)

By successively applying equations (5), (1) and (2), the general look of the projective camera equation can be deduced

\[
\begin{bmatrix}
    x_p \\
    y_p \\
    1
\end{bmatrix} =
\begin{bmatrix}
    f_x & s_{\alpha} & h_x \\
    0 & f_y & h_y \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    R_{cw} & w_c \\
    0 & 1
\end{bmatrix}
\begin{bmatrix}
    x_w \\
    y_w \\
    z_w \\
    1
\end{bmatrix}
\]

(6)
The reviewed model implies an optical system that is able to refract perfectly the beams in a way that they pass in a straight-line through the optical center over the entire range of the optical sensor, as shown in fig.1. This is of course practically impossible and systematic deviations from the model are often observed due to various manufacturing inaccuracies. That is how, for example, imperfections in the lens lead to radial distortion. If the photosensitive chip is not perpendicular to the optical axis, tangential picture distortions are observed.

These things considered, it is not the resolution itself, but the physical size of the sensor chip and its distance from the focal point that determine the viewing angle.

Whatever are the specific applications and devices used in the augmented and virtual reality apps, there is one common task that is central to achieving the set goals. For visualization of virtual objects, the point of view and the location of the virtual objects in the scene should be determined. In the case of an augmented reality the observer's point of view in the real world should also be found and transferred to the virtual reality viewer. In this way, real and virtual objects are visualized in a common frame and in the movement of the observation point are depicted in a common coordinate frame and there is no undesirable displacement between them.

In order to trace the movement of the camera, it is possible to use points from the environment, easily distinguishable from its surroundings [2,3], and based on their position in the space the position of observation is determinate. In this process the positions of new points in the environment are calculated, which in turn allows the process to continue. Thus, the problem of calculating the position of the camera and the calculation of the geometry of the surrounding environment are interconnected and carried out simultaneously in a process called Simultaneous Localization and Mapping (SLAM) [4].

There is an extensive base of algorithms to determine the geometry of space and to track the movement of the camera, each of which has its advantages and disadvantages [5]. Although they are well studied individually, it is necessary to analyze their feasibility in the field of virtual and augmented reality and to look for ways to combine them in order to achieve universality and applicability in a wide range of tasks.

3. Tracking the movement of the camera

The camera movement in tracking mode is determined based on the observed change in the image. The movement can be characterized as smooth and incremental as it does not exhibit any jumps in time. This time coherence enables the making of justified assumptions about the expected location of certain points of the model in the image. This drastically reduces the time needed to find correct correspondences and decreases the probability of incorrect ones.

In the framework of the development, the algorithm for tracking based on distinct points and descriptors without the use of optical flow was studied.

Optical Flow is a vector field defined on the image grid that assigns direction and speed to each point or sub-image. This vector field can be computed based on two consecutive frames without extracting specific points from it. Although global, the calculation process is quick. Its implementation can be found in the OpenCV library [6, 7].

When working with mobile devices, particularly sharp changes in speed and direction of movement and rotations at high angular speed scan are observed. These movements often go beyond the applicability range of methods to extract the optical image stream and cause interruptions in motion tracking. That is why in this case the motion tracking method based on distinct points in the image is used.

In essence, this is the same process used in the initialization to detect the match between the model points and the first image but applied to successive image frames. In this case, time coherence can also be applied as an additional criterion for speeding up the search for matches.

In the process of initializing image points, three-dimensional coordinates are assigned using the information obtained from the model. Tracking this information can also be obtained for additional
points using the calculated camera positions and triangulate the coordinates of the points observed in successive frames. In this way the base of starting point $A$ is expanded by the newly acquired points.

In addition to adding points to the database, the back process of its weak point that cleaning is also performed. This is important in order not to delay the tracking process. Thus, points with a small number of observed frames or those whose position is not confirmed by new observations are cleared from the database.

Determining the position of the camera is only based on those corresponding points that have enough consistent observations and already have assigned 3D coordinates.

The tracking process takes place in the following steps:

1) Retrieve an image from the camera and correct the deformations caused by the optics.
2) Significant points of the image are extracted: $b_i$.
3) For each point $b_i$ of the image is determined the potential match with point $c_i$ from the previous frame.
4) Based on the detected matches between the current frame and the previous one for each $b_i$ point with a correspondence already having three-dimensional coordinates, coordinates are compared in the three-dimensional space $m_i$, from the base $A$. Since there is no guarantee that all matches are true for the calculation of the camera position, RANSAC [8] method should be using the above input data as shown earlier to avoid calculation errors.
5) The "POSIT" algorithm [9], with its implementation in the OpenCV library, is used on the RANSAC minimum set of points. If this step cannot find a large enough consensus subset, the frame to frame tracking process is interrupted and goes into a re-initialization mode [10].

![Figure 2. Schematic diagram of the tracking module.](image-url)
6) The values obtained for the position of the camera \( t \) and its orientation \( R \), also known as its pose, is improved by minimizing the target function:

\[
\arg \min_{R, t} \sum_i \| m_i' - m_i \|^2,
\]

where \( m_i' \) is the projection of the \( m_i \) point through the camera at position \((R, t)\).

7) For each match it is tested whether the new coordinates lie on the epipolar line defined by the camera position in the previous frames where the point is detected. If the test is complete the match is added to base \( A \). If three or more observations are available for compliance a three-dimensional position is calculated by point triangulation, and if the projection error is not large it is written to the base. If the error is large, all the recorded data for the base point is deleted.

8) To maintain the volume of the base \( A \) constant, a sufficient number of points are cleared from it until the desired size is reached. A selection criterion is the projection error for each point by first removing the points with the greatest projection error.

9) The multiple \( b_i \) is copied to the \( c_i \) set and cleared. The result is passed to the preview system and passed to step 1.

The block summary of the module is shown in Fig. 2.

There is an example of the tracing process in fig. 3. Image a) shows a frame from the initial stage. It shows the open points on the front surface of a mouse box, which are pre-set in the database of points \( A \). Image b) shows a later frame where the side surfaces of the box become visible but no points on them are available yet, so the tracking continues only based on the front surface. In the last image (c), we have multiple points detected on all the visible sides of the box and the tracking uses their locations.

4. Realizing the camera-tracking module

The algorithmization of the process of tracking the movement of the camera can be divided into three stages:

1) Initialization - this step provides the initial position of the camera in the coordinate system of the surrounding world. It must be able to work autonomously by detecting specific points, objects or tags in the images.

2) Tracking - This step determines the movement of the camera frame by frame, assuming continuity of shooting. Temporal and spatial connectivity of the frames is assumed to determine the direction of movement. If sudden movements occur that cannot be calculated by the above methods, resort to those based on specific points that do not require continuity in camera movement but are significantly time consuming.

3) Reinitialization - this step serves to restore the position of the camera in the event that the tracking process is interrupted or fails. Unlike the initial initialization, there is additional information here about the scene and its generating geometry collected during tracing, which can be used to speed up the process.

The camera tracking module is implemented as a C++ language library. For connection to the camera, FlyCapture and v4l libraries on Linux are used. All common classes are departments as their
own library available for use by all parts of the program. Basic features for storing, processing, and analyzing images are provided by the OpenCV library.

To store the data needed to track the movement, a database class is defined to associate the point descriptors with additional information such as the number of observations to date, image positions and point of observation, as well as their coordinates in the space after they are determined.

5. Evaluation of the camera-tracking module
The precision of the system was measured based on a synthetic sequence of frames with known movement of the camera and the results compared to the motion detected by the tracking module. The results for the three axes of motion are shown in Fig. 4.

The black line shows the real trajectory for each axis, and red dots shows the calculated positions for each frame.

The image analysis itself for tracking purposes is not performed on the full resolution picture but on a 2 or 4 times reduced image depending on the freedom of movement of the camera to improve the tracking performance. With full freedom of movement, a 960x600 pixel image is used, and for a fixed tripod - 480x300 pixels.

With respect to the system's operating speed, it was reported that on a high-end computer system, the camera tracking was less than 33 milliseconds, ensuring a tracking speed of at least 30 frames per second. For the purpose of the application, this delivers the necessary realism. On desktops with sufficient computing power, the program easily achieves up to 60 frames per second, which is more than enough for even the most critical observers.

The main factor determining the system's performance is the ability of the database to quickly process queries for subset of points. Besides the choice of descriptor is critical for the speed of detection and comparison of image dots. Scale Invariant Feature Transform (SIFT) is not the fastest but it is an optimal compromise between speed and quality [11].

Figure 4. Precision of the tracking module.

6. Conclusion
This article looks at a model that is suitable for the purpose of 3D reconstruction. We describe an improved motion-tracking algorithm needed to create an augmented reality application. A tracking algorithm based on significant points and descriptors without the use of optical flow is used. The tracking module is written in C++. Experiments have been carried out of the precision of the system using the camera motion-tracking module created. Particular attention has been paid to the process of initialization and re-initialization for universal applicability.
References

[1] Hartley A and Zisserman R 2000 Multiple view geometry in computer vision Cambridge University Press

[2] Carmigniani J, Furht B, Anisetti M, Ceravolo P, Damiani E and Ivkovic M 2011 Augmented reality technologies, systems and applications Multimed. Tools Appl. 51 1 pp 341–377

[3] Mathworks Inc. 2016 Estimate Geometric Parameters of a Single Camera available at: http://mathworks.com/help/vision/ref/cameracalibrator-app.html

[4] Fuentes-Pacheco J, Ruiz-Ascencio J and Rendón-Mancha J M 2015 Visual simultaneous localization and mapping: A survey Artif. Intell. Rev. 43 1 pp 55–81

[5] Bekele M K, Pierdicca R, Frontoni E, Malinverni E S and Gain J 2008 A Survey of Augmented, Virtual, and Mixed Reality for Cultural Heritage J. Comput. Cult. Herit. 11 pp 1–36

[6] Devernay F E and Faugeras Q 2001 Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured environments Mach. Vis. Appl. 13 1 pp 14–24

[7] Bradski G 2015 OpenCV: Camera calibration With OpenCV, available at: http://docs.opencv.org/3.1.0/d4/d94/tutorial_camera_calibration.html

[8] Fischler M A and Bolles R C 1981 Random sample consensus Commun. ACM 24 6 pp 381–395

[9] Dementhon D F and Davis L S 1995 Model-based object pose in 25 lines of code Int. J. Comput. Vis. 15 1–2 pp 123–141

[10] Bleser G, Pastarmov Y and Stricker D 2005 Real-time 3d camera tracking for industrial augmented reality applications WSCG 2005 conference proceedings pp 47-54

[11] Lowe D G 2004 Distinctive image features from scale-invariant keypoints Int. J. Comput. Vis. 60 2 pp 91–110