An Optical Camera Tracking System for Virtual Sets Applications

Athanasios Drosopoulos, Yiannis Xirouhakis and Anastasios Delopoulos

Computer Science Div., Dept. of Electrical and Computer Eng.,
National Technical University of Athens,
9 Iroon Polytechniou str, GR-15773 Athens, Greece
Email: ndroso@image.ntua.gr

Abstract

During the past few years, there has been an increasing development in the Entertainment industry, thanks mainly to the endorsement of the computer graphics field advances in almost all video productions. In this context, virtual sets have become particularly popular due to the wide range of modern applications requiring composition of synthetic and natural visual information. Such systems have been developed for a variety of film and video productions, from sophisticated visual effects to simple TV programs.

However, existing approaches that go beyond static shots generally require expensive designated equipment or heuristic algorithms and human interaction. Present work provides both theoretical and practical guidelines for camera motion estimation using the late extensions in the traditional blue screen and employing a $3D$ motion estimation algorithm for real-time background updating.

Simulated experiments have been included to illustrate the performance of the proposed method.

1 Introduction

The composition of pre-loaded natural or synthetic sequences with captured natural data has become the state of the art in the Entertainment industry. In particular, interest in this area has raised, especially after the recent guidelines of the Motion Pictures Experts Group on MPEG-4 and MPEG-7 standards and the increasing development in dynamic $3D$ modeling. Virtual sets are a rapidly developing form of video production which is being used every day in news, sports, children's programming and even on the Internet. Broadcasts of this type were characterized as science broadcasts, introducing a new art in the worlds of film, video and television [1].

Virtual sets are based on the traditional blue screen and chroma-keying technologies that make a television meteorologist appear to be standing in front of a weather map, when he is actually standing in front of a blue screen. In this case, foreground action, captured in the studio, is combined off-line (or even on-line) with a given, natural or synthetic, background sequence. The actual background in such applications is the blue screen, a wall painted in a characteristic color that is not included in the foreground scene. In this way, after immediate segmentation of the foreground action from the blue screen background, the latter is substituted by the given pre-captured sequence. The recent trend in virtual sets, is the extension of the the single wall to a whole blue room. The result is that the objects or people that are actually being shot, appear to exist inside a $3D$ environment that has been pre-loaded.

However, what is, until recently, observed in such productions is the lack of camera motion. Such a task would require an equivalent

change of the background view in order to end up with a scene that looks natural. For example, if the camera was focusing in one of the actors and the background view applied to the scene still remained the same, the result would be far from the desired one. Virtual productions that go beyond static shots (or preplanned animations) involve some very advanced real and virtual camera matching procedures along with dynamic $3D$ modeling, image compositing, lighting and texturing capabilities and high speed rendering of animation.

Present work is focusing on constructing a blue screen such that the camera's relative position and orientation is determined on the basis of the frame captured. Then, by analyzing the information in the frames captured by the camera and comparing it to previous ones (or to a reference frame) $3D$ camera motion is extracted. Using the extracted information, we are then able to change the background view to obtain the same result, as that would be if the same camera was filming it. The proposed algorithm introduces a novel approach in the blue screen construction based on primitive polynomials [7] and the application of simple edge detection and $3D$ motion estimation [2, 3] for the extraction of camera motion. Relevant experimental results are included.

2 Scenario Description

As mentioned above, in a virtual set application, it is critical that camera motion is accurately estimated in real time. In this way, foreground natural action is correctly combined with the pre-loaded background. A very interesting case, where these results are applicable, appears when instead of a natural background, a synthetic $3D$ world is available with a virtual camera moving around it. Then, the virtual camera is subjected to the obtained $3D$ studio camera motion, and the background is refreshed dynamically through rendering of the synthetic scene.

In order to surpass the constraint of the stationary camera, there are mainly two camera tracking approaches employed. The first and most common one is based on the use of camera-mounted sensors which capture panning, tilting, zooming and $3D$ positioning data through time. This method leads to accurate results, however its application is limited, due to the fact that the designated equipment is very expensive and the constraints imposed on the camera's motion speed.

To obtain greater camera motion flexibility and lower cost, interest in the field has directed to a second, more hardware-independent, approach. Camera motion detection is performed on the basis of the captured information, allowing the camera to be hand-held and moved around the set. This approach surpasses the limitations imposed by systems using camera-mounted equipment, that demand post-shooting tasks, such as system set up and calibration. The most common solution, in this case, relies on the detection of key-points placed on the blue screen. Unfortunately, such a scheme requires motion tracking which itself is proved to be inaccurate, as well as a big deal of human interaction (for example when the camera toggles between speakers in the foreground or when key-points are occluded).

Another employed method, which seems more of a combination of the two aforementioned approaches, can be found in the $BBC's$ $Free-d$ system [8]. In this case, there is a small camera attached to the studio camera body, pointing upwards at targets on the lighting grid. The targets are circular barcodes, positioned at different heights, and as long as the camera can see at least four of them, its position can be located correctly.

In this work, we take advantage of the fact that each captured frame contains an already large amount of information about the background, that could provide additional information apart from just being used for foreground segmentation. The main idea is to store additional information in the blue screen for camera motion estimation purposes, without affecting its segmentation property. The latter is proved to be achieved without the use

of any special equipment or lighting. In this way, it suffices to estimate the $3D$ transformation, to which the visually apparent portion of the blue screen was subjected, in order to define the change in the $3D$ position of the camera. Similar ideas for the construction of the blue screen were used by ORAD [9] in their system released, up to our knowledge, in 1998. In that work, the blue screen was re-engineered using two tones of blue in order to create uniquely identifiable patterns. However, as it is reported in [9], the resulting blue screen was produced in terms of mixed pattern recognition algorithms and heuristics. One of the main contributions of this work is the establishment of theoretical guidelines for the construction of a two-tone blue screen, with the use of primitive polynomials and shift register sequences.

3 Blue Screen Construction

In our approach, the blue screen is constructed using two relatively close levels of blue color. The main contribution of this section is the subsequent theoretical approach in the construction of the blue screen binary map.

After dividing the rectangular wall (blue screen) in squares (or rectangles) of equal size, each block is 'painted' using darker or lighter blue. In the following, 1 and 0 will denote the dark and the light tone respectively. In this sense, if the blue screen contains $N \times M$ blocks, it is expressed in terms of an $N \times M$ matrix $\mathbf{B}$ with 1s and 0s in the appropriate positions.

In each frame, the camera captures a small portion of the wall, which corresponds to the respective portion Π of matrix $\mathbf{B}$. Π is a collection of elements of $\mathbf{B}$, generally not corresponding to some submatrix of $\mathbf{B}$. Let $\mathbf{S}$ denote the maximum submatrix of $\mathbf{B}$, which is included in Π. At this point, the question is whether one can determine uniquely the submatrix of $\mathbf{B}$ that is equal to $\mathbf{S}$. It will be shown, that $\mathbf{B}$ can be stuffed with 1s and 0s in a way that all the resulting $\mathbf{S}$s, of predefined dimensions, are unique in $\mathbf{B}$. In fact, one way to achieve the latter is through the utilization of primitive polynomials theory.

Let $\mathbf{p}$ be an order k polynomial, so that $p(x) = a_k x^k + a_{k-1} x^{k-1} + \cdots + a_0$ where $a_i \epsilon \{0, 1\}$. Let $R(p)$ be the corresponding simple shift register with $k-1$ positions, for which there is feedback connection in the i-th position, if and only if $a_{i+1} = 1$ [7]. If $\mathbf{p}$ is a primitive polynomial, then the sequence produced by $R(p)$ is maximal. In other words, the register goes through all the $2^k - 1$ states generated by k digits (having excluded the zero state), and thus the output sequence s is of length $2^k - 1$. It can then be seen that any given sequence of length k can be found only once in s.

Assume now that the minimum observed submatrix $\mathbf{S}$ of $\mathbf{B}$ is of length $n \times m$. We construct the $N \times M$ matrix $\mathbf{B}$ in terms of two primitive polynomials p_n and p_m of order n and m respectively, as

$$\mathbf{B} = p_n \times p_m \tag{1}$$

In this case, $N = 2^n - 1$ and $M = 2^m - 1$, and each $n \times m$ submatrix $\mathbf{S}$ of $\mathbf{B}$ is unique. Thus, by simple observation of the visible blue screen part in a frame, we can determine where the latter is located in the entire blue screen.

The choice of the parameters n, m depends both on the particular studio and the minimum portion Π of the blue screen that is expected to be visually apparent to the camera. The latter does not imply that the choice of the blue screen is application dependent, since the minimum visible portion can be determined by considering the worst cases.

The first consideration is the real-world blue screen dimensions. Assume that an $H_w \times W_w$ space is available (where H,W denote height and width) and the minimum portion visible by the camera is estimated to be $H_c \times W_c$. Let $H_b \times W_b$ be the unknown size

of each block. In this case,

$$N = \frac{H_w}{H_b},\ M = \frac{W_w}{W_b},\ n = \frac{H_c}{H_b},\ m = \frac{W_c}{W_b} \quad (2)$$

At the same time, $N = 2^n - 1$ and $M = 2^m - 1$, and solutions for the two unknowns H_b, W_b are obtained. Generally, the H_b, W_b are chosen well smaller than their limit values in order to ensure that a sufficient portion of the blue screen is always visible. On the other hand, H_b, W_b are also limited by the camera resolution, so that blocks are visually distinguished.

On the other hand, it can be seen that all rows of each **S** matrix are equal or zero, and the non-zero rows correspond to a state of the order-m register. Similarly all non-zero columns correspond to a state of the order-n register. In this sense, it suffices that one row and one column of **S** are visible, in order to determine to which submatrix of **B** it corresponds. This fact adds an error-detection property to the algorithm. If a block's color is misinterpreted as being 1 instead of 0 due to poor lighting and other shooting conditions, then the algorithm has the potential of detecting it and in the majority of cases correct it. In fact, we are currently investigating the prospect of implementing error codes in the blue screen construction using non-maximal sequences.

4 Key-points Extraction

As a next step, the information needed to estimate the camera's motion is extracted from each captured frame. At first, foreground objects are segmented from the blue screen background in terms of chroma-keying techniques. The latter is performed extending the existing segmentation technique to include the two shades of blue utilized. Afterwards, an edge detection scheme with a common gradient operator (e.g. Sobel) is applied to the background segment.

A subsequent line detection technique is utilized in order to locate those lines that consist the transformed sub-grid of the blue screen, that is visible to the camera. The Hough transform is used to locate the most probable to exist lines in the frame, based on the detected edges, apart from those that belong to the boundaries between the background and the foreground. We then calculate the cutting points between those lines and determine the transformed portion Π of the blue screen.

Still though, this portion may consist of large parts painted in one of the two tones of blue, in which we can not immediately detect the number of quadrangles included. In order to surpass this difficulty, the minimum quadrangle included in the image is detected. On the basis of the extracted block, we are then able to re-create the whole transformed part of the blue screen grid that is included in the frame captured, including the part that may be covered from the foreground. Any of the $n \times m$ **S** submatrices contained in the captured region Π can then be immediately determined on the basis of the pixels' colour in each detected block. This provides us with the information needed to decide which part of the blue screen is being captured by the camera.

Along the lines of the algorithm, correspondence is established between the extracted key-points (the cutting points that consist the selected **S**) and the actual points on the blue screen (see Subsection 5.2). These point correspondences are next fed to the $3D$ motion estimation algorithm to obtain camera motion.

5 Estimation of camera motion

As mentioned in Section 2, the application of a $3D$ camera motion estimation technique is required, so that the background natural/synthetic scene is respectively updated. In the ideal case, the background should update in such an accurate and smooth manner, as if it was not a distinct visual object but was part of the real-world camera capture. On the other hand, since generally in a

real-time transmission the human-eye attention is drawn by the foreground objects, small artifacts due to errors in camera motion estimation are acceptable.

As it will be explained in the sequel, the algorithms presented in [2, 3, 5] were considered in the proposed approach.

5.1 Initialization Steps

On the basis of the construction scenario, prior knowledge is available for:

- the camera initial $3D$ position and CCD geometric characteristics,
- the $3D$ blue screen position and area,
- the block lengths resulting from the partitioning of the blue screen,
- and the blue screen binary map, resulting to the particular block coloring.

Naturally, these parameters are given as input to the system, as an initialization step.

The initial camera focal length F must also be given, or equivalently a camera calibration post-processing step is required. Without loss of generality, let us assume that $F = 1$, in order to be compatible with the employed $3D$ motion estimation algorithms. Assume also that the lens center is located on the world origin $(0, 0, 0)$, in cartesian coordinates, and the barycentric coordinates for the CCD and the blue screen are $(0, 0, -1)$ and $(0, 0, 1)$ respectively in the reference scene (both CCD and blue screen are supposed to be initially parallel to the xy plane).

Then, with respect to the perspective projection model, when a point $\mathbf{p} = (x, y, z)$ projects onto $\mathbf{P} = (X, Y)$:

$$X = F\frac{x}{z} \text{ and } Y = F\frac{y}{z}. \qquad (3)$$

In this sense, having prior knowledge of the initialization parameters, the projected $2D$ reference scene is estimated in terms of its projected features, for example $\mathbf{P}_k^f = (X_k^f, Y_k^f)$ for the k-th employed point on the f-th available frame (f=0 for the reference frame). If supposedly (X_k^f, Y_k^f) were available for a sufficient number of points on the f-th frame, then $3D$ camera motion would be extracted using a $3D$ motion estimation algorithm.

For simplicity, from this point on, we will refer to the equivalent reverse problem, i.e. the case of a static camera and a moving plane (blue screen).

5.2 3D Motion Recovery

When the camera moves, the square blocks of the blue screen project to quadrangular ones. Naturally, all blocks are subjected to the same transformation. In this sense, it can be seen that the four transformed vertices of any extracted block contain sufficient information in order to estimate the relative motion between the camera and the blue screen [6]. The latter can be achieved through the utilization of a $3D$ motion and structure extraction algorithm [2]. Such algorithms require that the appropriate point correspondences are available for all employed points.

In fact, in our scenario, point correspondences could be estimated only in the case of relatively small camera movement using a motion estimation, feature tracking or local correlation scheme. However, such a scheme, apart from the considerable computational cost it engages, would fail mainly due to:

- fast camera movement,
- partial occlusion of the blue screen,
- or even total change in the visible blue screen part.

The latter is maybe the bottleneck in the estimation of camera motion, however, it can be seen that this constitutes a common case rather than an exception; consider for example the case where the two heroes converse, with the camera switching repeatedly between them. In fact, the major advantage of following the proposed in Section 3 methodology in the blue screen construction is the immediate extraction of key-point correspondences. One additional advantage can be seen, considering that straight lines are more accurately detected in comparison to single points.

Since, as described in Subsection 5.1, the projected CCD points are available both in

cartesian and pixel coordinates in the reference scene/frame, it suffices that correspondence is established with their extracted $2D$ counterparts in the f-th frame. However, the latter are available in pixel coordinates w.r.t. the particular frame. On the other hand, the portion of the blue screen in the reference frame, to which the visual portion in frame f corresponds, has been detected. In other words, correspondence is already established, as the extracted key-points are vertices of the blue screen portion at hand. The extracted point correspondences can then be fed to a $3D$ motion estimation algorithm. For this purpose, in this work, we employ a simple such algorithm for the case of planar surfaces [2] under perspective projection. The well-known employed algorithm estimates motion and structure for a planar surface moving relative to a camera.

Let $\mathbf{A}$ be the matrix defined by the following equations, which describe the movement of point $\mathbf{p}$ to $\mathbf{p}'$ in $3D$ space,

$$\mathbf{p}' = \mathbf{R}\mathbf{p} + \mathbf{T} = \mathbf{A}\mathbf{p} \ , \tag{4}$$

$$\mathbf{A} = \mathbf{R} + \mathbf{T} \ [a \ b \ c] \ , \tag{5}$$

where $\mathbf{R}$, $\mathbf{T}$ denote the rotation matrix and translation vector respectively and $[a \ b \ c]$ defines a $3D$ plane in the sense that $[a \, b \, c] \ \mathbf{p} = 1$.

Practically, for any $2D$ point correspondence $(X, Y) \rightarrow (X', Y')$, the elements a_i (so-called pure parameters) of matrix $\mathbf{A}$ are computed by

$$\begin{bmatrix} X & Y & 1 & 0 & 0 & 0 & -XX' & -YX' \\ 0 & 0 & 0 & X & Y & 1 & -XY' & -YY' \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_8 \end{bmatrix} = 0 \tag{6}$$

It can be shown that only eight out of the nine elements of $\mathbf{A}$ are independent. In this sense, given four point correspondences, four such pairs of equations are obtained and $\mathbf{A}$ is computed. Then, as it is shown in [2], the rotation and translation matrices are computed by means of a singular value decomposition of $\mathbf{A}$.

In the case of unknown $3D$ structure, translation and structure can be determined within a scalar ambiguity, as it can be seen from eq. (5). In our case, this ambiguity is withdrawn, as the initial relative camera and blue screen position is given by construction.

5.3 Accuracy Improvement

It must be noted here that for considerably noisy measurements resulting to inaccurate point correspondences, the employed algorithm [2] generally yields unsatisfactory motion estimates. For this reason, a more complicated algorithm, such as the one proposed in [3], is more appropriate. The particular method recovers motion parameters for the case of planar surfaces under perspective projection on the basis of noisy input correspondences.

In addition, it can be seen that the perspective model, depending on camera motion, can degenerate to the orthographic one. This is for example the case when the camera is located considerably far from the blue screen, so that the blue screen depth is less than 10 percent of its distance from the camera [6]. By 'blue screen depth' we denote the distance in z coordinates between the closest and the most distant blue screen points from the CCD. On the other hand, there is an upper limit on the distance between the camera and the blue screen, imposed by the fact that the camera field of view is restricted inside the blue screen (or else the studio environment would be present in the sequence). Since, in our experiments, it was found possible that the '10 percent constraint' is violated, the algorithm presented in [5] was deemed as most appropriate in such cases.

6 Simulations

The proposed approach has been experimentally verified over several synthetic sequences. Using an appropriate software package, synthetic $3D$ scenes were produced, consisting of foreground 'objects' and a background blue screen designed along the lines of the aforementioned method. The synthetic camera was

then subjected to a sequence of random moves and the resulting $3D$ scenes were rendered to produce image sequences.

One such scenario is illustrated in Figure 1. The particular blue screen was constructed considering the minimal visual subpart size as $n \times m = 5 \times 6$, resulting to a binary map of size $N \times M = 31 \times 63$. A frame of the produced sequence is illustrated in Figure 2 after rejecting foreground information. The outputs of the edge detection (Sobel) and the line detection (Hough) algorithms are depicted in Figures 3 and 4 respectively. In Figure 5 the cutting points and the minimum block vertices ('*' dot) have been extracted. Finally, the grid extension, in order to derive the rectangular visual subpart of the blue screen binary matrix, is depicted in Figure 6. The obtained pattern is illustrated in Figure 7.

The algorithm's efficiency was verified over a large number of synthetic sequences. In all cases, the error in the rotation angles obtained was less than 1^o, resulting to satisfactory background adaptation.

7 Conclusions

In the present work, a novel algorithm is presented for the estimation of camera motion in virtual sets applications. Initially, a blue screen construction scheme is proposed on the basis of two close blue levels and the primitive polynomials theory. Then, by analyzing the information in the frames captured by the camera and comparing it to a reference frame, point correspondences are established. Finally, $3D$ camera motion is extracted on the basis of a $3D$ motion estimation algorithm. Using the extracted information, one is then able to change the background view (blue screen) to obtain the same result, as that would be if the same camera was filming it.

Currently we are investigating the prospect of incorporating error codes in the blue screen construction using non-maximal sequences. At the same time, several algorithms using lines as input features for $3D$ motion extraction are being tested in our scenario.

References

[1] S. Gibbs, C. Arapis, C. Breiteneder, V. Lalioti, S. Mostafawy and J. Speier, "Virtual studios: an overview," *IEEE Multimedia,* vol.5, no.1, pp. 18-35, Jan. 1998.

[2] R.Y. Tsai and T.S. Huang and W.L. Zhu, "Estimating 3-D motion parameters of a rigid planar patch II: Singular value decomposition," *IEEE Trans. Acoust. Speech Sign. Proc.,* vol.30, no.4, pp.525-534, 1982; and vol.ASSP-31, no.2, p.514, 1983.

[3] J. Weng, T.S. Huang and N. Ahuja, "Motion and structure from point correspondences with error estimation: Planar surfaces," *IEEE Trans. Sign. Proc.,* vol.39, no12, pp.2691-2717, Dec. 1991.

[4] T. S. Huang and A. N. Netravali, "Motion and structure from feature correspondences: A review," *Proc.IEEE,* vol. 82, pp. 252-269, Feb. 1994.

[5] A. Delopoulos and Y. Xirouhakis, "Robust Estimation of Motion and Shape based on Orthographic Projections of Rigid Objects," *IMDSP98 (IEEE),* Alpbach Austria, July 1998.

[6] M.Tekalp, "Digital Video Processing," *Prentice Hall,* 1995.

[7] S.W. Golomb, "Shift register sequences," *Holden-Day,* 1967.

[8] *http://www.bbc.co.uk/rd/projects/virtual*

[9] *http://www.orad.co.il*

Figure 1: Synthetic scenario

Figure 2: Captured frame background

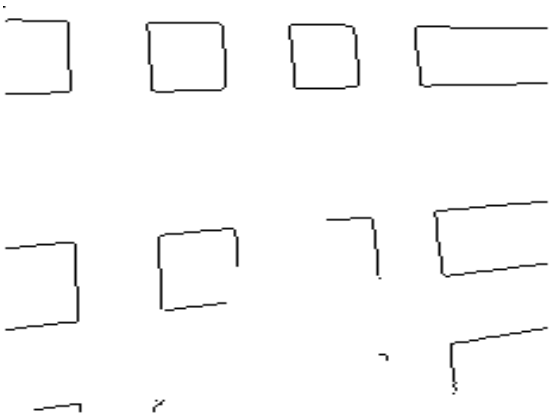

Figure 3: Edge detection results

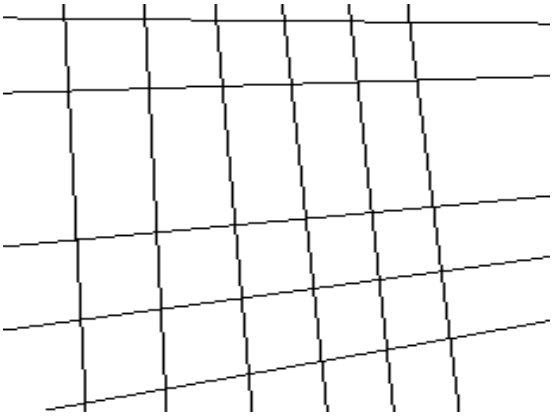

Figure 4: Line detection results

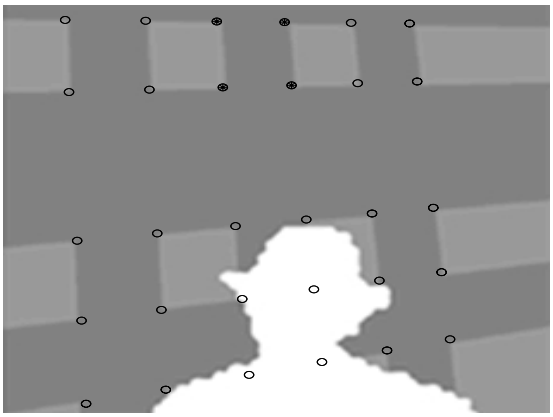

Figure 5: Extracted intersections

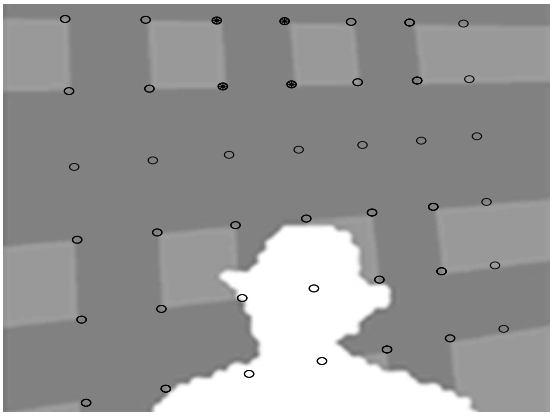

Figure 6: Extended grid

Figure 7: Extracted pattern