MULTI-CAMERA EPIPOLAR PLANE IMAGE FEATURE DETECTION FOR ROBUST VIEW SYNTHESIS

Lode Jorissen\textsuperscript{1}, Patrik Goorts\textsuperscript{1}, Sammy Rogmans\textsuperscript{1}, Gauthier Lafruit\textsuperscript{2}, Philippe Bekaert\textsuperscript{1}

\textsuperscript{1}Hasselt University - tUL - iMinds
Expertise Centre for Digital Media - Wetenschapspark 2 - 3590 Diepenbeek, Belgium

\textsuperscript{2}Université Libre de Bruxelles
LISA department - Av. F.D. Roosevelt 50 CP165/57 - 1050 Brussels, Belgium

ABSTRACT

In this paper, we propose a novel, fully automatic method to obtain accurate view synthesis for soccer games. Existing methods often make assumptions about the scene. This usually requires manual input and introduces artifacts in situations not handled by those assumptions. Our method does not make assumptions about the scene; it solely relies on feature detection and utilizes the structures visible in a 3D light field to limit the search range of traditional view synthesis methods. A visual comparison between a standard plane sweep, a depth-aware plane sweep and our method is provided, showing that our method provides more accurate results in most cases.

Index Terms — Soccer, View Synthesis, Disparity Maps, Epipolar Plane Image, Plane Sweep, Light Fields

1. INTRODUCTION

In recent years there is an increasing interest in applications such as Free Viewpoint Navigation (FVN) and Super Multi-View (SMV) displays. One important part of these applications is the generation of virtual views using a limited set of cameras. Different techniques, such as stereo matching \cite{1, 2}, plane sweep \cite{3} and light field based methods \cite{4, 5} exist for view synthesis.

While these techniques do a good job at generating virtual views in well-textured scenes, they often fail in regions with no or almost no texture. The main reason for this is that these techniques try to match patches between different images. In regions with no or similar textures, this often results in incorrect matches, and thus incorrect depth information which is used to generate the virtual views. This limitation makes FVN and the generation of views for SMV-displays a difficult task in certain scenarios.

One such case is a soccer game. Both FVN and SMV-displays are useful tools for analyzing the in-game events and for showing the match to the public \cite{6}. However, the soccer field, as seen by cameras, consists of a green plane with nearly no texture. Such a scene will undergo the aforementioned problems when using the traditional view synthesis techniques. Work has been done on improving the view synthesis in this scenario. Some techniques, for example, generate billboards of the players and use those to create virtual views \cite{7, 8}. Other techniques, such as the work of Hayasaki and Saito \cite{9, 10} and the depth-aware plane sweep of Goorts et al. \cite{6}, make assumptions about the scene to improve the view synthesis. However, these solutions often require manual input or produce distracting artifacts when some parts of the visible scene are not correctly preprocessed.

To handle these issues, we propose a method that utilizes the line structures in Epipolar Plane Images (EPIs) to create an initial disparity map for each input view, similar to the method of Xu et al. \cite{5}, which will significantly improve the quality of existing methods. We use a descriptor-based approach to detect the structures on the soccer field in a stable way. The resulting disparity maps are used to determine a local disparity search range for the plane sweep algorithm. By doing this we drastically reduce the effective search space and thus reduce the chance of having a mismatch, and therefore improve overall quality.

Our method works well for regions with a small amount of detail. Since we make no special assumptions about the scene, even objects that are not part of the game, such as the public, are correctly interpolated. This also results in a fully automatic method, which does not require manual input. The camera arrangement however is limited to a linear camera setup. This method was tested on recordings from a real-life soccer game.

In the next section we will describe the necessary preprocessing steps. In section 3 the algorithm for estimating the disparity map is explained. Section 4 gives a short overview of the general plane sweep algorithm. Section 5 provides results and a comparison between the traditional plane sweep \cite{3}, the method provided by Goorts et al. \cite{6} and our method.

2. PREPROCESSING

Our algorithm uses EPIs to estimate disparity maps. This requires all cameras to be placed on a line. The images also need to be rectified, i.e. a feature on a scanline in one camera should remain on the same scanline in the other cameras. The algorithm thus needs to know the pose of each camera to accomplish this. A feature-based calibration algorithm, proposed by Goorts et al. \cite{11}, is used to automatically calibrate the camera array. The poses obtained by this calibration are used to rectify the images. Color correction is applied to accommodate for color differences between the cameras. This is accomplished by estimating a color correction matrix that minimizes the differences in color for the matches obtained in the calibration step. The background for each camera is obtained by taking the mean color value for each pixel over multiple frames. All these steps are fully automated.

3. DISPARITY INITIALIZATION

In this section we describe the algorithm that estimates the initial disparity maps of the input view. This map will be used to define a local disparity range in which view synthesis algorithms should search for optimal correspondences between frames. The
algorithm makes use of the line structures observed in Epipolar Plane Images (EPIs) [12]. An EPI is a 2D-slice (i.e., an image) of a 3D cube, representing a 3D light field, which was generated by stacking the rectified images of a linear camera array on top of each other. The EPI can be thought of as copying the scanline $y$ from every rectified input image, at a position $p$ along the linear array, to row $p$ in the EPI image. These images contain line structures which represent the position and color of points in space. We will call these structures EPI-lines. For each EPI, the algorithm detects features in each scanline and matches them between different scanlines. These matches are used to create a sparse set of lines, which are considered to be the EPI-lines. These lines are then used to create disparity EPIs (DEPIs) which can be used to construct disparity maps. The extracted backgrounds from the preprocessing step are used to obtain a more accurate estimate. Figure 1 gives an overview of the algorithm.

3.1. Feature Detection

The first step is to detect features that can be used for detecting EPI-lines in each EPI. To reduce noise, a horizontal Gaussian filter can optionally be applied to the scanlines of the EPI. Next, the horizontal gradient $D(x) = |I(x) - I(x - 1)|$ of each scanline $I$ of the EPI is determined. The local extrema in the gradient are considered to be good features for matching and line detection.

3.2. Matching

After feature detection, a descriptor for each feature is calculated. Each descriptor describes the surrounding of its feature in the rectified input image. They are used to match each feature to features on the other scanlines in the EPI. We used the U-SURF descriptor since the features undergo only a small rotation [13]. We empirically observed that U-SURF provides more accurate results than normal SURF or other similar descriptors in our setup.

We limited the matching to features that are on the same scanline in the different images. This can be thought of as matching features between the different rows of an EPI. At the end of this step each feature $F$ has a set $S$ of similar features in the different rows of the EPI. Each set may still contain some outliers.

3.3. Finding Lines

Using the matches $S$ for each feature $F$ we estimate a line using the following steps:

1. Determine $d = \frac{X(F) - X(Cam(F))}{Cam(F)}$ for $F$ and each $F_i \in S$, where $X(F)$ is the position of $F$ along its scanline and $Cam(F)$ represents its camera position. Hence, $d$ is an estimate of the corresponding feature disparity.
2. Group the features in $S$ with a similar $d$ and select the group $G$ with the most features. Also add $F$ to $G$.
3. If $G$ has at least $n$ features, where $n$ is a threshold that limits the inclusion of outliers:
   a. Fit a line through the features in $G$.
   b. Remove features in $G$ that are too far away from the line. (i.e., whose distance to the line is more than $s$ pixels, where $s$ represents the maximum distance to the line). If less than $n$ points remain: ignore this feature.
   c. Refit the line using the remaining features in $G$.
   d. Store features that are close to the line in a set $H$. (i.e., whose distance to the line is less than $s$ pixels).
   e. Refit the line using the features in $H$. A disparity $d = \frac{1}{m}$ is assigned to the line, where $m$ represents the slope of the line.

The result is a set of lines for each EPI that will be used to construct a corresponding disparity EPI. $n$ is a threshold that makes sure that only features with a decent number of matches are passed on to the next step. This limits the inclusion of outliers. To prevent having duplicate lines we keep track of the features that are assigned to a line. These features are no longer used in the above steps. In case that two lines intersect, the line which corresponds to the point farthest away from the camera setup is split at the intersection and only the segment that has the most features in $G$ is kept. The other line stays intact. Lines that are clearly incorrect (e.g., lines with an invalid slope) are removed from the final set.

3.4. Creating Disparity Maps

Before creating the disparity maps, we first create disparity EPIs (DEPIs) using the estimated lines. The DEPIs have the same size as the EPIs. The disparity of each pixel in the DEPI is determined by selecting the lines immediately left and right to the pixel. The disparity of the line that represents the point farthest away from the cameras is assigned to the pixel. After creating all DEPIs, the disparity maps can be generated by moving the scanlines from the DEPI to their corresponding scanlines in the disparity maps. This is the inverse of how the EPIs were created. A median filter is applied to reduce the influence of possible outliers.

3.5. Using Background Information

The background images of the soccer field can be used to correctly fill in small regions, such as the space between the arms and the body of the players or regions between two players. In this case the disparity maps are first estimated for the background. Next the foreground objects (e.g., the players and ball) are extracted from the input images using background segmentation. EPI-lines are extracted and grouped for these foreground objects. The disparity maps for those objects are then created using the EPI-lines.
assign to that object. Disparities are only estimated for pixels which represent foreground objects, and which are surrounded by two lines. In the final step, the disparity maps from the background and foreground objects are merged.

3.6. Search Range

The resulting disparity maps are used to determine a local disparity range in which an interpolation algorithm, such as plane sweep, will search for a local minimum cost. This search range is determined by applying a minimum and maximum filter (erosion and dilation) on the disparity maps to determine a local minimum and maximum value. The minimum and maximum values may be scaled to add a margin to the search space. This creates two disparity maps for each camera that represent the minimum and maximum search range for each pixel. These disparity maps are passed on to a plane sweep algorithm for view synthesis.

4. PLANE SWEEP

Plane sweep is a view synthesis algorithm that first deprojects the input images onto a virtual plane, located at a distance $D_i$ from the virtual camera. The resulting images are then projected onto the virtual camera’s image plane. The color consistency for each pixel is determined for different depths $D_i$, and the best color consistency for each pixel is saved [3, 6].

The disparity maps calculated in our algorithm are used to limit the range of $D_i$ for each pixel. This is accomplished by taking the minimum and maximum disparity in the neighborhood of the pixel and scaling them with a factor. By doing this we significantly reduce the number of mismatches in color consistency, which results in the generation of more accurate virtual views.

5. RESULTS

Our algorithm was tested on recordings of a real-life soccer game. The sequence was captured with 8 cameras placed on a line. There was approximately 1m distance between the adjacent cameras. For this setup we chose the threshold $n$ to be 4, i.e. at least 4 points should match a line. The maximum distance $s$ between the point and its line was set to 2 pixels. The size of the median filter was set to $7 \times 7$, while the size of the erosion and dilation kernels was set to $23 \times 23$. Figure 2 gives a close-up of the disparity maps obtained with these settings.

We will now make a comparison of the results obtained with a standard plane sweep (SPS) [3], the depth aware plane sweep (DAPS) provided by Goorts et al. [6] and our proposed method. In this experiment we synthesized the fourth view and compared it to the corresponding original image. The fourth camera was not used as input for the view synthesis. The SPS uses a fixed search range for the whole image. To increase the quality of its output all 7 cameras were used. Also a neighborhood of $9 \times 9$ was used to calculate the color consistency. DAPS on the other hand fits a plane through the soccer field and uses it as the depth for the background. The players are segmented from the background and they are interpolated using plane sweep. Our method, as mentioned earlier, uses a local disparity range estimated from the EPIs and passes it on to a plane sweep algorithm. Both DAPS and our method use only three cameras and only one pixel was used to determine the color consistency. Contrary to DAPS, our method does not use segmentation during the plane sweep.

For each algorithm a virtual view was created at the position of the reference image (Figure 3). The rectified images were used by each implementation. Table 1 shows the close-ups for the areas marked in Figure 3.

As can be seen in the first two close-ups the SPS method almost completely removes the soccer players from the field. The DAPS method on the other hand suffers significantly less from these artifacts, but incorrect interpolation still regularly occurs as observed in the second close-up: the region between the arm and the leg is filled up incorrectly. Our EPI based method on the other hand results in no significant artifacts.

When looking at the public in the third close-up, we see that SPS interpolates the view correctly for the most part, but artifacts are noticeable. DAPS on the other hand deforms the stage and its public. The reason for this is that the stage is not taken into account by the assumptions made, i.e. the stage is considered to lay in the same plane as the soccer field. This also results in the removal of objects and people, such as the person with the orange coat at the top of the stairs. Our method clearly doesn’t suffer from any of these shortcomings.

6. CONCLUSION

By visually comparing the results of different view interpolation methods we can state that our method results in a more accurate view synthesis than the traditional plane sweep and the Depth-aware plane sweep. Our method has close to no visible artifacts and tends to keep the details of the soccer field intact. Contrary to methods that utilize the scene of a soccer game, our method is able...
to handle objects outside of the game correctly. These results were achieved by utilizing the structures in a 3D light field to find a local disparity search range for the plane sweep algorithm. Contrary to most soccer specific interpolation algorithms, our method does not require manual input. However, the camera arrangement is currently limited to a linear setup.

7. REFERENCES


