Video Fields: Fusing Multiple Surveillance Videos into a Dynamic Virtual Environment

Ruofei Du,[*] Sujal Bista,[†] and Amitabh Varshney[‡]
Augmentarium, Department of Computer Science and the Institute for Advanced Computer Studies (UMIACS)
University of Maryland, College Park

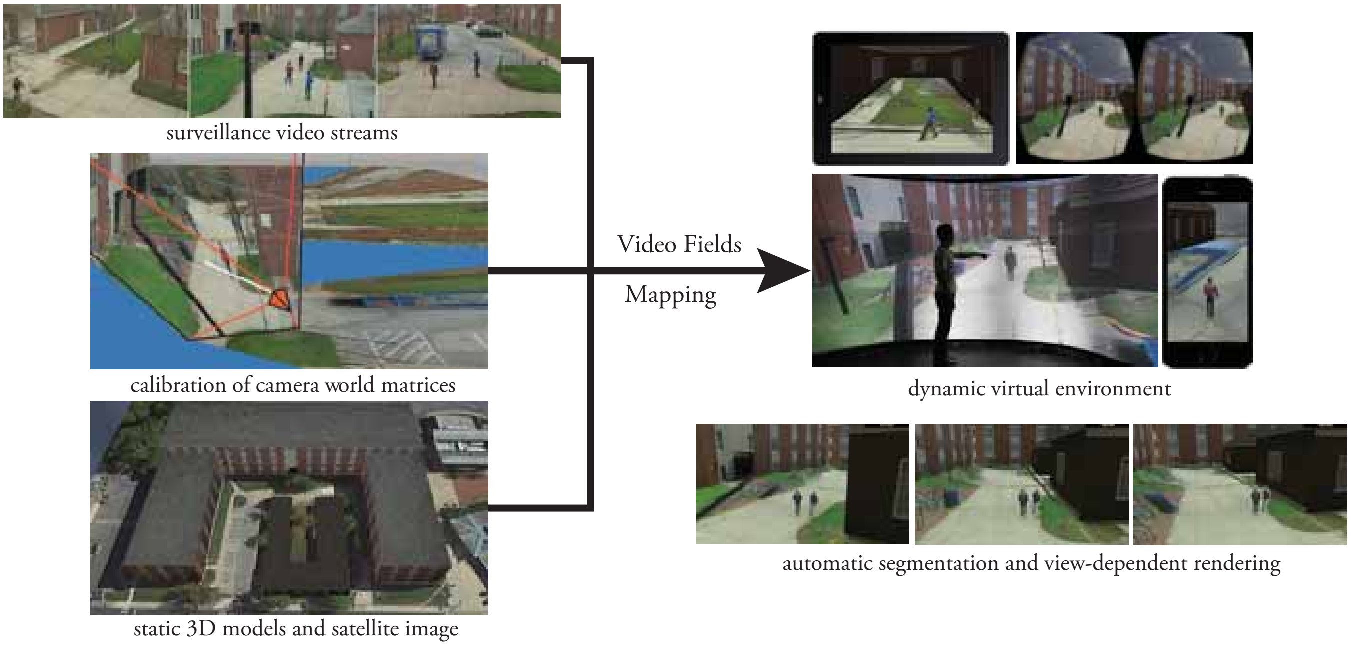

surveillance video streams
calibration of camera world matrices
static 3D models and satellite image
Video Fields
Mapping
dynamic virtual environment
automatic segmentation and view-dependent rendering

Figure 1: *Video Fields system fuses multiple videos, camera-world matrices from a calibration interface, static 3D models, as well as satellite imagery into a novel dynamic virtual environment. Video Fields integrates automatic segmentation of moving entities during the rendering pass and achieves view-dependent rendering in two ways: early pruning and deferred pruning. Video Fields takes advantage of the WebGL and WebVR technology to achieve cross-platform compatibility across smart phones, tablets, desktops, high-resolution tiled curved displays, as well as virtual reality head-mounted displays. See the supplementary video at http://augmentarium.umd.edu and http://video-fields.com.*

Abstract

Surveillance videos are becoming ubiquitous for monitoring and ensuring security. Nevertheless, mentally fusing the data from multiple video streams covering different regions comes with a high cognitive burden. In this paper we introduce, Video Fields, a novel web-based interactive system to create, calibrate, and render dynamic video-based virtual reality scenes in head-mounted displays, as well as high-resolution wide-field-of-view tiled display walls. Video Fields system automatically projects dynamic videos onto user-defined geometries. It allows users to adjust camera parameters, navigate through time, walk around the scene, and see through the buildings. Our system integrates background modeling and automatic segmentation of moving entities with rendering of video fields. We present two methods to render video fields: early pruning and deferred pruning. Experimental results indicate that the early pruning approach is more efficient than the deferred pruning. Nevertheless, the deferred pruning achieves better results through anti-aliasing and bi-linear interpolation. We envision the use of the system and algorithms introduced in Video Fields for immersive surveillance monitoring in virtual environments.

Keywords: virtual reality; mixed-reality; video-based rendering; projection mapping; surveillance video; WebGL; WebVR

Concepts: **•Computing methodologies → Virtual reality;** *Image-based rendering;*

1 Introduction

Surveillance videos play a crucial role in monitoring a variety of activities in shopping centers, airports, train stations, and university campuses. In conventional surveillance interfaces, where multiple cameras are depicted on a display grid, human operators endure a

[*] e-mail: ruofei@cs.umd.edu
[†] e-mail: sujal@umiacs.umd.edu
[‡] e-mail: varshney@umiacs.umd.edu

Web3D '16,, July 22 - 24, 2016, Anaheim , CA, USA
ISBN: 978-1-4503-4428-9/16/07 $15.00
DOI: http://dx.doi.org/10.1145/2945292.2945299

high cognitive burden in fusing and interpreting multiple camera views (Figure 2). In the field of computer vision, researchers have made great strides in processing the surveillance videos for segmenting people, tracking moving entities, as well as classification of human activities. Nevertheless, it remains a challenging task to fuse multiple videos from RGB cameras, without prior depth information, into a dynamic 3D virtual environment. Microsoft Kinect, for example, cannot capture the infrared structured lighting patterns in the sunlight. In this paper, we present our research in fusion of multiple video streams taking advantage of the latest WebGL and WebVR technology.

Figure 2: *This photograph shows conventional surveillance interface where multiple monitors are placed in front of the operators. One of the greatest challenges for the users is to mentally fuse and interpret moving entities from multiple camera views.*

With recent advances in virtual reality (VR) headsets and web-based interactive graphics technology, there is an increasing demand for real-time photo-realistic dynamic scene generation. As an exploratory work, we investigate the following research questions:

1. Can we efficiently generate dynamic scenes from surveillance videos for VR applications?
2. Can we use a web-based interface to allow human operators to calibrate the cameras intuitively?
3. Can we use the-state-of-the-art web technologies to achieve interactive video-based rendering?
4. Can we render moving entities as 3D objects in virtual environments?

In this paper, we present our early solutions to create dynamic virtual reality from surveillance videos using web technologies. This paper makes following contributions to the Web3D, graphics, and virtual worlds' communities:

1. conception, architecture, and implementation of Video Fields, a mixed-reality system that fuses multiple surveillance videos into an immersive virtual environment,
2. integrating automatic segmentation of moving entities in the Video Fields rendering system,
3. presenting two novel methods to fuse multiple videos into a dynamic virtual environment: *early pruning* and *deferred pruning* of geometries,
4. achieving cross-platform compatibility across a range of clients including smart phones, tablets, desktops, high-resolution large-area wide field-of-view tiled display walls, as well as head-mounted displays.

2 Related Work

We have built our Video Fields system upon previous work in fusing from multiple photographs and videos, modeling and visualization of surveillance videos, as well as spatio-temporal video navigation and analysis.

2.1 Fusing Multiple Static Photographs

The idea of fusing multiple digital photographs using computer vision techniques seems intuitive, considering the spatial and visual features of underlying images. Prior research has addressed crucial problems and implemented novel systems to fuse multiple digital photographs. For example, the University of Washington's Photo Tourism[1] project [Snavely et al. 2006] and Microsoft's PhotoSynth[2] project [Sinha et al. 2009] use local scale-invariant features [Lowe 1999] to estimate corresponding points between images and reconstruct 3D point clouds of architectural tourist landmarks, thus providing a unique visualization for seamless browsing of multiple photographs. Patro *et al.* [2010] have described their Social Snapshot system, which reconstructs a textured and colored mesh from a loosely-ordered photo collection rather than the sparse or dense-point reconstruction used in PhotoSynth and Photo Tourism. Recently, Autodesk's 123D Catch[3] enables ordinary consumers to scan static 3D objects by using phones or tablets to capture multiple photographs and upload to the cloud servers.

In our Video Fields system, instead of fusing from multiple photographs, we focus on a greater challenge: fusing multiple videos. There are two significant challenges in fusing videos: systematically handling (a) dynamic textures and (b) dynamic meshes or point clouds. Dynamic textures demand an efficient segmentation approach for real-time rendering, while dynamic meshes or point clouds require efficient pruning techniques to improve the rendering performance.

2.2 Fusing Multiple Dynamic Videos

Many methods have been proposed to address the problem of 4D reconstruction from dynamic videos using multi-view reconstruction algorithms. We give a brief overview of recent research in fusing content from standard color (RGB) video cameras as well as video cameras that capture color with depth (RGBD).

Furukawa *et al.* [2008] designed and implemented a markerless motion capture system from synchronized video streams acquired by calibrated cameras. While compelling, it took two minutes on a dual Xeon 3.2 GHz workstation to process a single frame. De *et al.* [2008] proposed a system that reconstructs space-time coherent geometry with motion and textural surface appearance of actors performing complex and rapid moves. However, this also suffers from slow processing speed (approximately 10 minutes per frame), largely due to challenges in stereo matching and optimization. Since then, a number of advances have been made in dealing with video constraints and rendering quality [Cagniart et al. 2010; Vlasic et al. 2008; Xu et al. 2011; Casas et al. 2013], but rendering dynamic scenes in real-time from RGB video streams has remained a challenge. In our approach, we did not employ structure-from-motion algorithms for fusing multiple surveillance videos due to the demands imposed by real-time surveillance monitoring. Instead, we use projection-mapping and pruning algorithms to render the mixed reality scene in real time.

[1] Photo Tourism: http://phototour.cs.washington.edu

[2] PhotoSynth: https://photosynth.net

[3] 123D Catch: http://www.123dapp.com/catch

With recent advances in consumer-level depth sensors, a number of dynamic reconstruction systems use Microsoft Kinect or Intel Real Sense to generate dynamic geometries from point clouds. KinectFusion by Newcombe *et al.* [2011] and Izadi *et al.* [2011] was the first system that tracks and fuses point clouds into dense meshes using a single depth sensor. However, the initial version of KinectFusion could not handle dynamic scenes. The systems developed by Ye *et al.* [2014] and Zhang *et al.* [2014] are able to reconstruct non-rigid motion for articulated objects, such as human bodies and animals. Further advances by Newcombe *et al.* [2015] and Xu *et al.* [2015] have achieved more robust dynamic 3D reconstruction from a single Kinect sensor by using warp-field or subspaces for the surface deformation. Both techniques warp a reference volume non-rigidly to each new input frame. Nevertheless, these systems rely on a volumetric model that is used for model fitting, which is limited in accommodating fast movement and major changes in the shapes. In addition, the reconstructed scenes still suffer from the occlusion issues since the data comes from a single depth sensor.

Recently, Dou *et al.* [2016] have proposed the first system, *Holoportation*, to capture dynamic 3D performance in real-time by using multiple RGBD cameras. The *Holoportation* system makes no prior assumption, such as any pre-defined templates, regarding the target scene and is highly robust to sudden motion and large changes in meshes. Nevertheless, all of these depth-sensor-based approaches are unable to work in outdoor scenarios, such as surveillance monitoring, because the sunlight interferes with the structured lighting patterns used by the current generation of depth cameras. In Video Fields system, we use just the RGB cameras to achieve photo-realistic 3D scene rendering in real-time, for *both* outdoor and indoor scenarios.

2.3 Modeling and Visualization of Surveillance Videos

Modeling and visualization of moving entities such as humans, vehicles, and even unmanned aerial vehicles have triggered significant research in both computer vision and computer graphics fields. Kanade *et al.* [1998] were the first to introduce the idea of importing people from surveillance videos into a 3D scene in the DARPA Image Understanding Workshop. Sankaranarayanan *et al.* [2009] have taken the first steps in rendering moving pedestrians in surveillance videos as virtual 3D human models. However, this work did not employ dynamic texture mapping onto the human models, which makes it difficult to distinguish one moving character from another. The Video Surveillance and Monitoring (VSAM) system, invented by Hall *et al.* [2002], first demonstrated the feasibility of fusing live outdoor videos with maps and top-view aerial images. Instead of projection mapping, their system uses a four-point registration procedure to simplify the projection and does not correct the perspective from arbitrary viewpoints. Neumann *et al.* [2004] presented the Augmented Virtual Environment (AVE), which used video-based projection mapping onto 3D models to create a mixed-reality environment. Nonetheless, AVE does not remove background pixels when rendering moving entities and only supports a single billboard in the rendering pass. Recently, Evans *et al.* [2015] presented a web-based visualization of multimodal data including LIDAR data, static image reconstruction, as well as witness video. This is an impressive advance that overcomes many challenges of data transfer limitations and rendering power. In contrast to their approach of rendering the entire video as a billboard, in our approach we use video-based projection mapping.

There are three distinct contributions of our Video Fields system compared with the previous systems. First, Video Fields system presents two new approaches to interactively segment and render the moving entities in real time. Second, our system provides a unique web-based interface to simplify the calibration and modeling processes, which usually requires professional domain knowledge and significant time. Finally, our system is the first web-based system that fuses and renders multiple videos in a virtual reality headset or a tiled display.

2.4 Spatio-temporal Video Navigation and Analysis

Besides 3D video-based rendering techniques, prior research has also devised novel interfaces for spatio-temporal video navigation and analysis. For example, Girgensohn *et al.* [2007a] invented a real-time, indoor video surveillance system with twenty cameras called Dynamic Object Tracking System (DOTS). DOTS was able to display nearby video feeds based on the referenced view, providing spatial information for easier navigation and analysis. As reported in a formal user study [Girgensohn et al. 2007b], DOTS was more helpful in tracking tasks than camera display arrays. DOTS also introduced a 3D viewer that visualizes moving entities as billboards in a 3D context. Their system also showed that embedding people in a 3D viewer is useful for video surveillance. In our approach, we build upon this work to merge multiple videos with the latest advances in web technology and use transparency and new pruning techniques to achieve superior fusion and rendering.

Wang *et al.* [2007] have explored combining videos with 3D spatial layouts. Their pioneering work has identified user tasks in video surveillance that are likely to benefit with a spatial-temporal context. They proved that embedding videos in a 3D context can improve task performance for video-model relationship tasks. We build upon their research by including video projection with appropriate segmentation and making our results be available on a variety of platforms using WebGL and WebVR technology. We believe that users can benefit more from an immersive visualization experience of multiple fused videos in a virtual environment.

3 System Design

Video Fields system consists of a camera world calibration interface, a back-end server to process and stream the videos, and a web-based rendering system. The flow chart of the system is shown in Figure. 3.

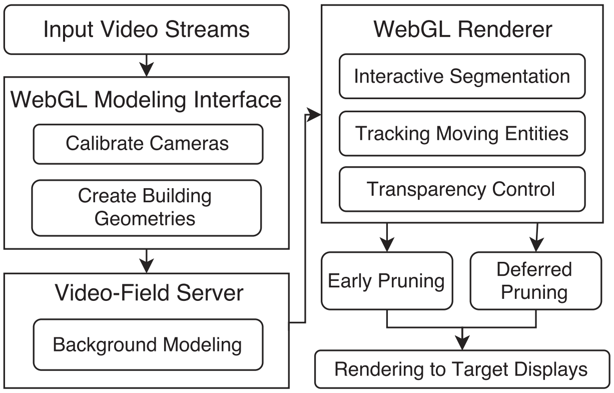


Figure 3: *This figure shows the work flow of Video Fields. Our system imports video streams as video textures in WebGL. In the WebGL camera world calibration interface, the user may create the ground, calibrate the cameras, and add initial geometries. The videos are then sent to the Video Fields server for generating a background model.*

3.1 Camera-World Calibration Interface

The camera-world calibration interface for Video Fields has been built based on WebGL and WebVR. We use the open-source library `Three.js`[4] to create the interface and render the mixed-reality scene. We have designed a four-stage work flow for Video Fields. First, users import their videos into the system. During this stage, the users can also alter the video time-line to crop and synchronize the videos manually, if needed. Then users can define the ground projection plane for the projection. Second, users adjust the position and the rotating quaternion for each camera in the videos as if they were orienting flashlights onto the ground projection plane. We have used the transform controller in the `Three.js` library to provide such an interface. Third, users can drag geometric primitives for constructing buildings for the 3D world. The user can watch real-time rendering results interactively as they are dragging the geometric primitives. Finally, users can use any number of display devices, including the VR headsets, to experience the world that they have just created. Using the estimated position and size of each geometry, we can attach textures to the user-defined building geometry. We can also use a sky sphere to enhance the visual immersion. The position, scaling, and rotation of the cameras (as camera world matrices $\boldsymbol{C}$) and the 3D models are exported and saved in the JSON format.

3.2 Background Modeling

The motivation of background modeling is to provide a background texture for each camera to identify moving entities in the rendering stage and to reduce the network bandwidth requirements when streaming videos from the web-server. With a robust estimation of the background, only the pixels corresponding to the moving entities will need to be streamed for rendering and not the rest of the video for every frame.

In order to estimate robust background images, we take advantage of Gaussian Mixture Models (GMM) for background modeling. Compared with the mean filter or the Kalman filter, GMM is more adaptive with different lighting conditions, repetitive motions of scene elements, as well as moving entities in slow motion [Stauffer and Grimson 1999].

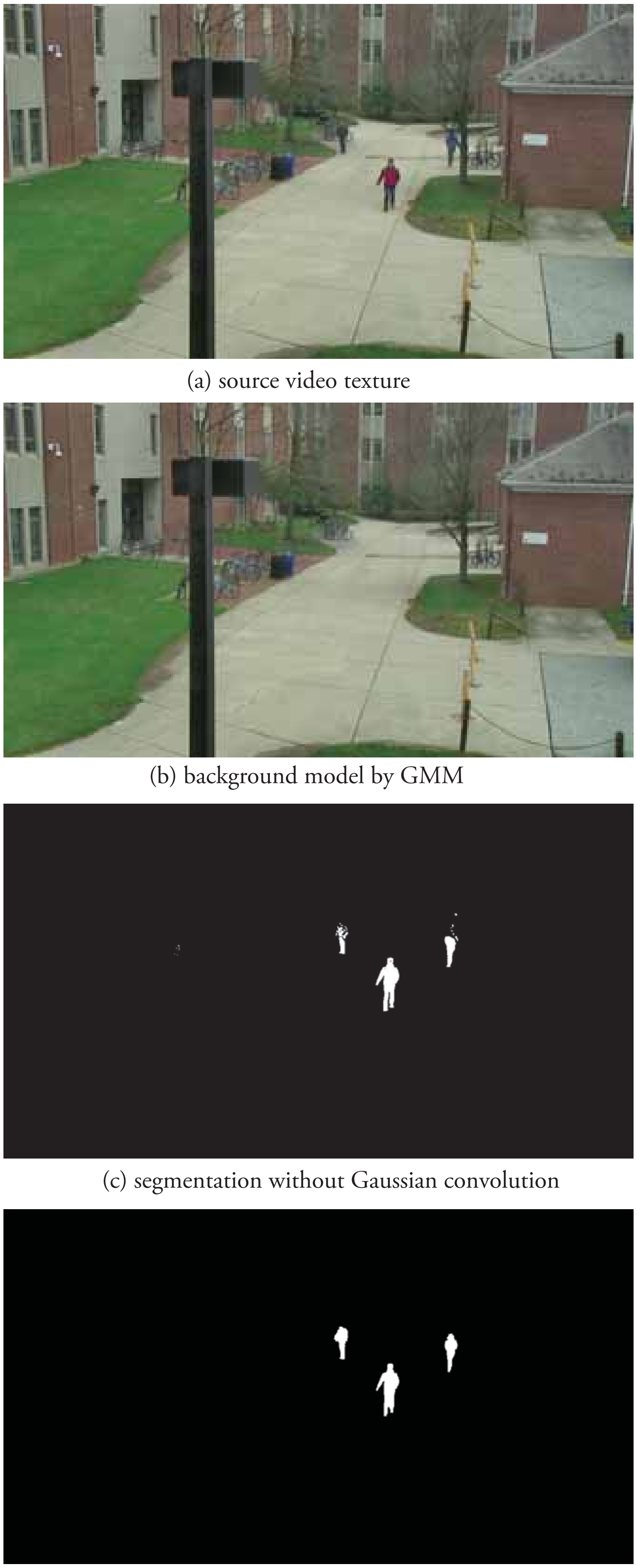
Given the video texture $\boldsymbol{T}$, for each pixel $\boldsymbol{t}_{uv}$ at the texture coordinates (u, v), we model the background value of the pixels as a mixture of N Gaussian distributions $\mathscr{P}_{uv}$. For each frame at time $i \in \{1, \ldots, M\}$, we denote $\boldsymbol{T}(u, v)_i$ as the set of all previous pixels at $\boldsymbol{t}_{uv}$, $\mathscr{P}(\boldsymbol{T}(u, v)_i)$ as the probability of the background color of the pixel at (u, v):

$$\begin{aligned} \boldsymbol{T}(u,v)_i &= \{\boldsymbol{T}(u,v,j), 1 \leq j \leq i\} \\ \mathscr{P}(\boldsymbol{T}(u,v)_i) &= \sum_{j=1}^{N} \mathscr{N}(\boldsymbol{T}(u,v)_i | \mu_{ij}, \Sigma_{ij}) \cdot \omega_{ij} \end{aligned} \tag{1}$$

where N is the total number of Gaussian distributions. We set $N \leftarrow 3$ in our implementation and $\mathscr{N}$ is the Gaussian probability density function:

$$\mathscr{N}(\boldsymbol{T}(u,v)_i | \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}}} \cdot \frac{1}{\Sigma^{\frac{1}{2}}} \cdot e^{-\frac{1}{2}(\boldsymbol{T}(u,v)_i - \mu_{ij})^T \sigma_{ij}^{-1} (\boldsymbol{T}(u,v)_i - \mu_{ij})} \tag{2}$$

where Σ is the covariance matrix. We assume that the red, green, and blue channels are independent and have the same variances.



(a) source video texture

(b) background model by GMM

(c) segmentation without Gaussian convolution

(d) segmentation with Gaussian convolution

Figure 4: *(a) shows a reference frame of the video texture, (b) shows the background model learnt by the Gaussian Mixture Models approach, (c) shows segmentation without Gaussian convolution, and (d) shows the segmentation with Gaussian convolution*

[4] Three.js: http://threejs.org/

Thus, the GMM can be trained at a lower cost of computing the covariance matrix. To update the Gaussian distributions, we used an online K-means algorithm to learn the weights ω_{ij}:

$$\omega_{ij} \leftarrow (1-\alpha)\omega_{i(j-1)} + \alpha\mathcal{M}_{ij} \tag{3}$$

where α is the learning rate and $\mathcal{M}_{ij}$ indicates whether the model is matched to the current pixel. An example of the computed background model is shown in Figure 4(b).

3.3 Segmentation of Moving Entities

After learning the background model for each video, we achieve real-time interactive segmentation of moving entities during the fragment-shader rendering pass by taking advantage of the many-core computing on the GPU. To alleviate noise and smooth the boundaries, given input video texture $\boldsymbol{T}$ and its corresponding background model $\boldsymbol{B}$, we convolve $\boldsymbol{T}$ and $\boldsymbol{B}$ with a Gaussian kernel $\mathscr{G}$ at scale σ:

$$\boldsymbol{T}' \leftarrow \mathscr{G}(\sigma) \otimes \boldsymbol{T}, \boldsymbol{B}' \leftarrow \mathscr{G}(\sigma) \otimes \boldsymbol{B} \tag{4}$$

We segment the moving entities by the thresholding function δ:

$$\boldsymbol{F} \leftarrow \delta(|\boldsymbol{I}' - \boldsymbol{B}'|) \tag{5}$$

A comparative example showing the advantage of using the Gaussian convolution is shown in Figure 4 (c) and (d) using the same threshold of 0.08. This parameter is passed to the fragment shader as a GLSL floating-point number, thus enabling the user to interactively alter the thresholding function in the WebGL browser. After obtaining the foreground $\boldsymbol{F}$, we also calculate the set of bounding rectangles $\boldsymbol{R}$ of moving entities in $\boldsymbol{F}$.

3.4 Video Fields Mapping

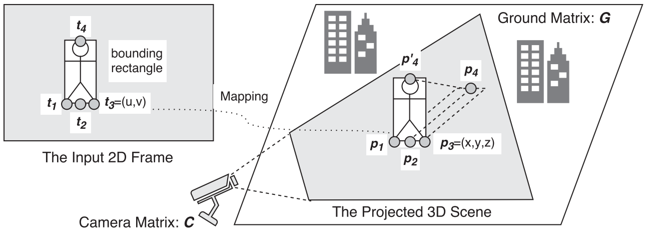


Figure 5: *This figure shows the the overview of the Video Fields mapping by projecting 2D imagery from the streaming videos onto a 3D scene; $t_1, \ldots, t_4$ indicate the key points of the foreground bounding boxes in the input 2D frames and $p_1, \ldots, p_4$ indicate the corresponding projected points.*

To map video fields onto the geometries in the 3D virtual environment, we need to establish bidirectional mapping between the texture space and the 3D world space. Here, we use the ground model as an example of an arbitrary geometry in the 3D scene. The challenges we address here are as follows:

- Given a vertex on the ground model, we need to calculate the corresponding pixel in the texture space. This is important for visualizing the color of the ground model.
- Given a pixel in the texture space, we need to calculate the corresponding vertex on the ground to project that pixel. This is important for projecting a 2D segmentation of a moving entity to the 3D world.

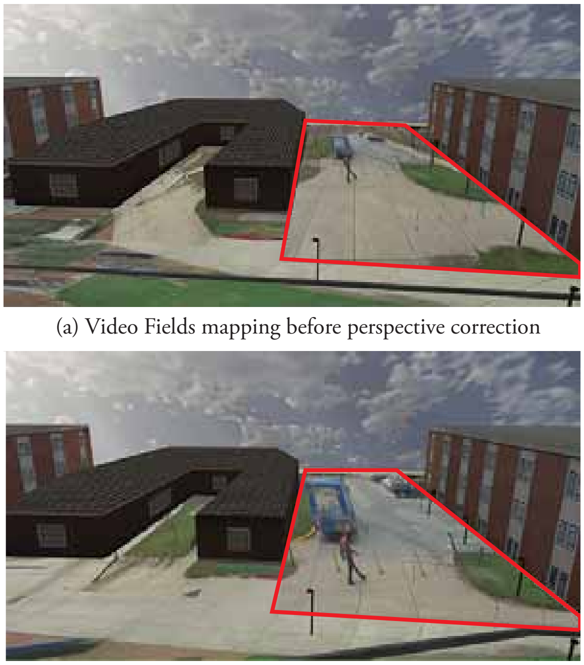
The first challenge could be solved by projection mapping, but we need to correct for the inherent perspective in the acquired video. Given the camera-world matrix $\boldsymbol{C}$, which is obtained from the WebGL-based camera-world calibration interface, and the model matrix of the ground $\boldsymbol{G}$. For each vertex $\boldsymbol{p}_{xyz}$ on the ground, we first use the camera-world matrix and the ground-model matrix to convert its coordinates to the homogeneous coordinates in the camera space:

$$\hat{\boldsymbol{p}}_{xyzw} \leftarrow \boldsymbol{C} \cdot \boldsymbol{G} \cdot (\boldsymbol{p}_{xyz}, 1.0) \tag{6}$$

To obtain the texture coordinates $\boldsymbol{t}_{uv}$, we need to carry out perspective correction:

$$\boldsymbol{t}_{uv} \leftarrow \left(\frac{\hat{\boldsymbol{p}}_x + \hat{\boldsymbol{p}}_w}{2\hat{\boldsymbol{p}}_w}, \frac{\hat{\boldsymbol{p}}_y + \hat{\boldsymbol{p}}_w}{2\hat{\boldsymbol{p}}_w} \right) \tag{7}$$

Figure 6 shows the results before and after the perspective correction.



(a) Video Fields mapping before perspective correction

(b) Video Fields mapping after perspective correction

Figure 6: *This figure shows the results before and after the perspective correction. The texture in (a) has the same perspective as the original video, but is not projected correctly in the 3D scene.*

The second challenge, to convert a 2D point to 3D, is non-trivial because projection from 3D to 2D is irreversible. To do this, we first compute a dense grid contained within the ground projection of the video. For each 3D vertex of this grid, we compute the corresponding 2D coordinates in the video texture using equations 4 and 5. We store the results in a hash function $\mathscr{H}$. Since we are using a dense grid, all points in the video texture can be mapped to a 3D vertex on the ground. Finally, we store the results in a hash function:

$$\mathscr{H}: \boldsymbol{t}_{uv} \longmapsto \boldsymbol{p}_{xyz}. \tag{8}$$

Once calculated, this hash map can be stored on the server side and be used for mapping 2D texture points back to the 3D world. For each vertex $\boldsymbol{p}$ on the ground, we also calculate the angle between the camera ray and the ground surface $\theta_{\boldsymbol{p}}$.

3.5 Early Pruning for Rendering Moving Entities

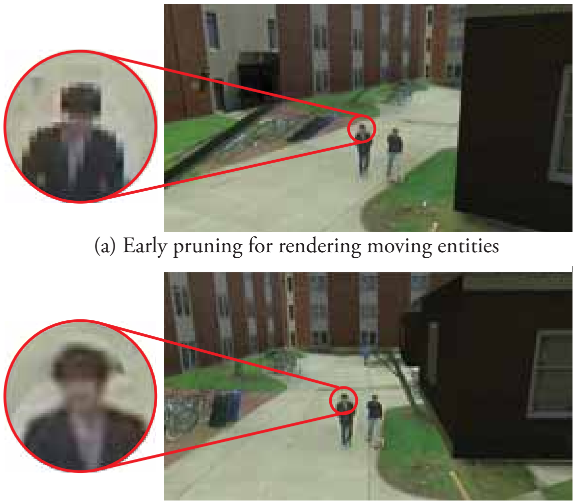
To render moving entities in the 3D world, we remove the background pixels from the video texture and correct the projection so that the moving entities are vertical on the ground model. In the *Early Pruning* approach, we discard the pixels that do not belong to the foreground as soon as the foreground is identified after the Gaussian convolution and thresholding by equations 4 and 5. Then the foreground pixels are transformed into a 3D point cloud and projected to the 3D world space. We use point clouds to implement the early pruning technique and optimize the rendering performance of Video Fields since they are extremely efficient to render. The detailed algorithm is described in Algorithm 1. View-dependent results of our visualization technique are shown in Figure 4.

3.6 Deferred Pruning for Rendering Moving Entities

Though pruning the background pixels at an early stage is useful for our videos where most pixels belong to the background, we have also developed the deferred pruning approach for better anti-aliasing, bi-linear sampling and faster visibility testing. In this approach, we dynamically project videos on moving billboards. The background subtraction is completed in the fragment shader of each billboard. After the world matrix of a billboard is determined, we test the visibility of the billboard. We then render foreground pixels onto the billboard and discard the background pixels. We describe the details in Algorithm 2.

3.7 Visibility Testing and Opacity Modulation

One of the biggest advantages for visualizing videos in an immersive 3D virtual environment is that the system is able to adjust the opacity of every object, thus allowing the users to “see-through buildings”. The users can therefore observe the video-recorded activities from cameras that would otherwise not be viewable from a user’s given vantage point. We achieve this by doing a visibility test using ray-casting from the current camera to every moving entity in the scene. If a moving entity is found to be occluded, we can modulate the opacity of that occluding object and render the moving entities. Figure 8 shows an example before and after the visibility test and opacity modulation.



(a) Early pruning for rendering moving entities

(b) Deferred pruning for rendering moving entities

Figure 7: *This figure shows the segmentation of moving entities, view-dependent rendering, as well as zoom-in comparison between the early pruning algorithm and the deferred pruning algorithm.*

4 Experiments and Evaluation

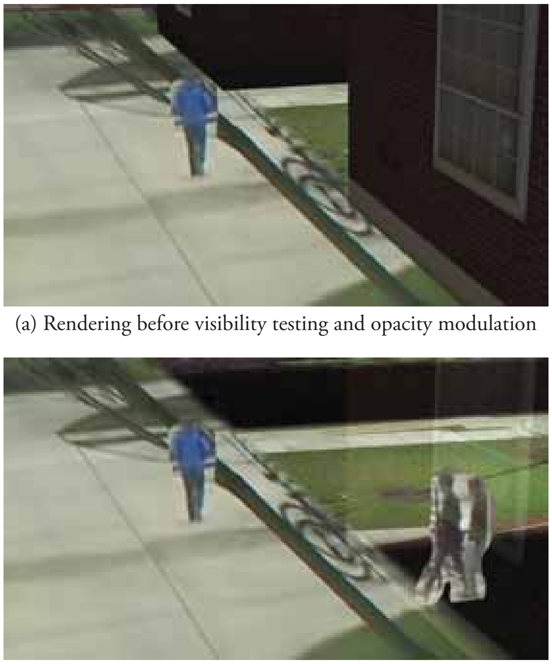
In our experiment, we recorded three 10-minute video clips with the resolution of 1280 × 720 pixels. We tested both the early- and deferred-pruning algorithms in the following three settings. The first two tests were conducted on a desktop workstation with a NVIDIA Quadro K6000 graphics card running Windows 8.1 on Google Chromium 48.0.2544.0 with WebVR enabled. We tested on a regular desktop display with a resolution of 2560 × 1440 as

ALGORITHM 1: Early Pruning for Rendering Moving Entities

```
Input: foreground $\boldsymbol{F}$ and the set of bounding rectangles $R$ of
       moving entities
Output: a 3D point cloud $P$ visualizing the moving entities
Initialize a set of points for the video visualization. (Run once);
For each pixel $\boldsymbol{t}$ inside the bounding box, calculate the intersection
  point $\boldsymbol{t}_\perp$ between its perpendicular line and $\boldsymbol{t}_1\boldsymbol{t}_3$;
for each pixel $\boldsymbol{t}$ from the video do
    if $\boldsymbol{t} \notin \boldsymbol{F}$ then
        discard $\boldsymbol{t}$ and continue;
    set the color of the pixel: $c \leftarrow \mathbf{texture2D}(\boldsymbol{F}, \boldsymbol{t})$;
    look up the corresponding projected points in the 3D scene:
      $\boldsymbol{p} \leftarrow \mathscr{H}(\boldsymbol{t}), \boldsymbol{p}_\perp = \mathscr{H}(\boldsymbol{t}_\perp)$;
    update the $z$ coordinate of the 3D point:
      $\boldsymbol{p}_z \leftarrow |\boldsymbol{p} - \boldsymbol{p}_\perp| \cdot \tan(\theta_{\boldsymbol{p}})$ ;
    use the $x, y$ coordinates of $\boldsymbol{t}_\perp$ to place the point vertically:
      $\boldsymbol{p}_{xy} \leftarrow \boldsymbol{t}_{uv}$ ;
    render the point $\boldsymbol{p}$;
```

ALGORITHM 2: Deferred Pruning for Rendering Moving Entities

```
Input: foreground $\boldsymbol{F}$ and the set of bounding rectangles $\boldsymbol{R}$ of
       moving entities
Output: a set of billboards rendering the moving entities
Initialize a set of billboards to display moving objects. (Run once);
for each detected bounding box $\boldsymbol{r}$ in $\boldsymbol{R}$ do
    calculate the bottom-left, bottom-middle, bottom-right and
      top-middle points $\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{t}_3, \boldsymbol{t}_4$ in $\boldsymbol{r}$, as illustrated in Fig. 5;
    look up the corresponding projected points in the 3D scene:
      $\boldsymbol{p}_i \leftarrow \mathscr{H}(\boldsymbol{t}_i), i \in \{1, 2, 3, 4\}$.;
    calculate the width of the billboard in the 3D space:
      $w \leftarrow |\boldsymbol{p}_3 - \boldsymbol{p}_1|, h \leftarrow |\boldsymbol{p}_4 - \boldsymbol{p}_2| \cdot \tan(\theta_{\boldsymbol{p}_4})$.;
    Reposition a billboard to the position $\frac{\boldsymbol{p}_1 + \boldsymbol{p}_3}{2}$ with width and
      height $w$ and $h$;
    In the fragment shader of the billboard, sample the color from
      $I$ as described in Equation. 6 and 7, but replace $G$ with the
      current billboard's model matrix; discard pixels which does
      not belong to the foreground $F$;
```



(a) Rendering before visibility testing and opacity modulation

(b) Rendering after visibility testing and opacity modulation

Figure 8: *This figure shows the rendering results before and after the visibility test and opacity adjustment. Note that the two people behind the building are correctly rendered through the semi-transparent meshes.*

well as Oculus Rift DK2 head-mounted display with a resolution of 950 × 1080 for each eye. The last test was conducted in an immersive curved screen environment with 15 projectors driven by 4 NVIDIA Quadro K6000 graphics cards. The rendering resolution was 6000 × 3000 pixels. The results were rendered with the same software setup. The experimental results are shown below:

Table 1: *The experimental results of early pruning and deferred pruning for different display devices and resolutions.*

Render Algorithm	Resolution	WebVR	Framerate
Early Pruning	2560 × 1440	No	60.0 fps
	2 × 960 × 1080	Yes	55.2 fps
	6000 × 3000	No	48.6 fps
Deferred Pruning	2560 × 1440	No	60.0 fps
	2 × 960 × 1080	Yes	41.5 fps
	6000 × 3000	No	32.4 fps

From the table above, both deferred pruning and early pruning achieve interactive rates in desktop settings. However, for stereo rendering using WebVR-enabled browser and high-resolution rendering, deferred pruning suffers from lower frame rate. This is because deferred pruning carries out the texture sampling after the vertex transformation for each billboard. However, the advantage of the deferred pruning approach is that it achieves better anti-aliasing results than early pruning as shown in Figure. 7. On the other hand, the early pruning approach samples the colors at an early stage and discards unnecessary background pixels in the fragment shader, making it a faster approach than the deferred pruning.

Please visit the websites http://augmentarium.umd.edu and http://video-fields.com for supplementary materials related to this paper.

5 Conclusions and Future Work

In this paper, we have described a web-based rendering system that fuses multiple surveillance videos to create a dynamic virtual environment.

Our approach leverages the recent advances in web-based rendering to design and implement the Video Fields system to provide a more immersive and easier-to-use dynamic virtual environment for visualizing multiple videos in their appropriate spatial context. We have compared two new ways of fusing moving entities into the 3D world: early pruning and deferred pruning. We found that each has its relative advantages. The choice of which technique to use will depend on the characteristics of the environment being recorded as well as the preferred display device.

In future, we plan to scale up our approach to handle hundreds of surveillance videos spread over a wider area. To do so effectively, we plan to use techniques from image and mesh saliency [Lee et al. 2005; Kim and Varshney 2008; Kim et al. 2010; Ip and Varshney 2011]. As we broaden the scale and scope of our work, more efficient distributed systems and parallel computing algorithms will be necessary to achieve interactive rendering rates. We also plan to explore the integration of our efforts with scalable, distributed, and parallel web-services platforms such as Amazon's S3.

Acknowledgements

We wish to thank *Tim Gray*, *Xuetong Sun*, *Hsueh-Chien Cheng* and *Eric Krokos* from the Institue for Advanced Computer Studies (UMIACS) at the University of Maryland, College Park for their help in acquiring the surveillance videos, *Sai Yuan* from the Department of Animal Science for helping with the vocal track of the demo. We thank the anonymous reviewers from the ACM SIGGRAPH Web3D community for their meticulous reading of our manuscript, as well as their insightful suggestions and comments.

This work has been supported in part by the NSF Grants 09-59979, 14-29404, 15-64212, the State of Maryland's MPower initiative, and the NVIDIA CUDA Center of Excellence. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of the research sponsors.

References

Cagniart, C., Boyer, E., and Ilic, S. 2010. Probabilistic Deformable Surface Tracking from Multiple Videos. In *ECCV'10 Proceedings of the 11th European Conference on Computer Vision: Part IV*. Springer, 326–339.

Casas, D., Tejera, M., Guillemaut, J.-Y., and Hilton, A. 2013. Interactive Animation of 4D Performance Capture. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 19*, 5, 762–773.

De Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., and Thrun, S. 2008. Performance Capture from Sparse Multi-View Video. *ACM Transactions on Graphics (TOG) 27*, 3, 98.

Dou, M., Khamis, S., Degtyarev, Y., Davidson, P., Fanello, S., Kowdle, A., Escolano, S. O., Rhemann, C., Kim, D., Taylor, J., Kohli, P., Tankovich, V., and Izadi, S. 2016. Fusion4D: Real-time Performance Capture of Challenging Scenes. In *ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*, ACM.

EVANS, A., AGENJO, J., AND BLAT, J. 2015. Hybrid Visualisation of Digital Production Big Data. In *Proceedings of the 20th International ACM Conference on 3D Web Technology (Web3D)*, ACM, 69–72.

FURUKAWA, Y., AND PONCE, J. 2008. Dense 3D Motion Capture from Synchronized Video Streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 1–8.

GIRGENSOHN, A., KIMBER, D., VAUGHAN, J., YANG, T., SHIPMAN, F., TURNER, T., RIEFFEL, E., WILCOX, L., CHEN, F., AND DUNNIGAN, T. 2007. DOTS: Support for Effective Video Surveillance. In *Proceedings of the 15th International Conference on Multimedia*, ACM, 423–432.

GIRGENSOHN, A., SHIPMAN, F., TURNER, T., AND WILCOX, L. 2007. Effects of Presenting Geographic Context on Tracking Activity Between Cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1167–1176.

HALL, B., AND TRIVEDI, M. 2002. A Novel Graphical Interface and Context Aware Map for Incident Detection and Monitoring. In *9th World Congress on Intelligent Transport Systems*, ITS, 1–9.

IP, C. Y., AND VARSHNEY, A. 2011. Saliency-Assisted Navigation of Very Large Landscape Images. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 17*, 12, 1737–1746.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., ET AL. 2011. KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST)*, ACM, 559–568.

KANADE, T., COLLINS, R., LIPTON, A., BURT, P., AND WIXSON, L. 1998. Advances in Cooperative Multi-Sensor Video Surveillance. In *DARPA Image Understanding Workshop*, vol. 1, DARPA, 3–24.

KIM, Y., AND VARSHNEY, A. 2008. Persuading Visual Attention Through Geometry. *IEEE Transactions on Visualization and Computer Graphics 14*, 4, 772–782.

KIM, Y., VARSHNEY, A., JACOBS, D. W., AND GUIMBRETIÈRE, F. 2010. Mesh Saliency and Human Eye Fixations. *ACM Transactions on Applied Perception (TAP) 7*, 2, 12.

LEE, C. H., VARSHNEY, A., AND JACOBS, D. W. 2005. Mesh Saliency. In *ACM Transactions on Graphics (TOG)*, vol. 24, ACM, 659–666.

LOWE, D. G. 1999. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV)*, vol. 2, IEEE, 1150–1157.

NEUMANN, U., YOU, S., HU, J., JIANG, B., AND SEBE, I. O. 2004. Visualizing Reality in an Augmented Virtual Environment. *Presence: Teleoperators and Virtual Environments 13*, 2, 222–233.

NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. 2011. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, IEEE, 127–136.

NEWCOMBE, R. A., FOX, D., AND SEITZ, S. M. 2015. DynamicFusion: Reconstruction and Tracking of Non-Rigid Scenes in Real-Time. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 343–352.

PATRO, R., IP, C. Y., BISTA, S., AND VARSHNEY, A. 2010. Social Snapshot: A System for Temporally Coupled Social Photography. *IEEE Computer Graphics and Applications*, 1, 74–84.

SANKARANARAYANAN, A. C., PATRO, R., TURAGA, P., VARSHNEY, A., AND CHELLAPPA, R. 2009. Modeling and Visualization of Human Activities for Multicamera Networks. *EURASIP Journal on Image and Video Processing 2009*, 259860.

SINHA, S. N., STEEDLY, D., AND SZELISKI, R. 2009. Piecewise Planar Stereo for Image-Based Rendering. In *Proceedings of the 12th International Conference on Computer Vision (ICCV)*, IEEE, 1881–1888.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo Tourism: Exploring Photo Collections in 3D. *ACM Transactions on Graphics (TOG) 25*, 3, 835–846.

STAUFFER, C., AND GRIMSON, W. E. L. 1999. Adaptive Background Mixture Models for Real-Time Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, IEEE, 246–252.

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated Mesh Animation from Multi-view Silhouettes. *ACM Transactions on Graphics (TOG) 27*, 3, 97.

WANG, Y., KRUM, D. M., COELHO, E. M., BOWMAN, D., ET AL. 2007. Contextualized Videos: Combining Videos with Environment Models to Support Situational Understanding. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 13*, 6, 1568–1575.

XU, F., LIU, Y., STOLL, C., TOMPKIN, J., BHARAJ, G., DAI, Q., SEIDEL, H.-P., KAUTZ, J., AND THEOBALT, C. 2011. Video-Based Characters: Creating New Human Performances from a Multi-View Video Database. *ACM Transactions on Graphics (TOG) 30*, 4, 32.

XU, W., SALZMANN, M., WANG, Y., AND LIU, Y. 2015. Deformable 3d fusion: From partial dynamic 3d observations to complete 4d models. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2183–2191.

YE, M., AND YANG, R. 2014. Real-Time Simultaneous Pose and Shape Estimation for Articulated Objects Using a Single Depth Camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2345–2352.

ZHANG, Q., FU, B., YE, M., AND YANG, R. 2014. Quality Dynamic Human Body Modeling Using a Single Low-Cost Depth Camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 676–683.