Visual Reconstruction using Geometry Guided Photo Consistency

Jan-Friso Evers-Senne, Andreas Niemann, Reinhard Koch

Institute for Computer Science, Christian-Albrechts University
24103 Kiel, Germany
{evers, aniemann, koch}@mip.informatik.uni-kiel.de

Abstract

In this paper we present an approach to generate novel views from multiple images without known geometry at interactive framerates. Starting with a set of calibrated images, coarse depth information is computed for the images using a plane-sweep algorithm. During interactive rendering, the geometry approximation is used to guide the photo consistent reconstruction based on a tiled plane-sweep refinement. This results in significant speed-up and enhanced image quality at the same time. The approach is designed to use todays graphics processing unit and can fuse color information from up to 8 views which allows to handle occlusions and fill holes. The focus of the reconstruction is the visual consistency, not precise geometry. While geometric reconstruction is often hard to achieve for homogeneous image regions, visual reconstruction is easier to achieve and sufficient for view interpolation.

1 Introduction

View interpolation is one key issue for free viewpoint TV. Given a set of cameras observing an arbitrary scene, the user should be able to watch that scene from any view point, not only from the camera positions. Image interpolation techniques avoid the need of explicit models or complete reconstruction. Many of these so called *Image Based Rendering* (IBR) systems have been developed in the last years, [5].

The basic idea for our approach is to precompute depth maps for all real views without the need of high accuracy, and to use those depth maps to guide the view interpolation, to reduce visible artefacts, and to enhance the performance. For view interpolation it suffices to generate a new view which is photo consistent with the scene, the correct depth is not the primary goal. Computing best possible depth maps is a problem on its own and typically takes much computational time. But computing an approximate depth can be done much faster and helps to improve the view interpolation considerably.

2 Related Work

Novel view generation is a well known problem in computer vision and computer graphics. It is also referred as *Image Based Rendering*. In [5] view generation is generalised as reconstruction and resampling of the plenoptic function. Different approaches have been published to achieve this. They can be classified based on the amount of scene geometry which is required and on the allowed movements of the virtual camera. Shum et al [8] classified image based rendering into three categories: Using no geometry (Light-Field), using explicit geometry (view dependent texture mapping) and using implicit geometry. Implicit geometry means that geometry information is extracted from the images itself, either as a preprocessing step or during interpolation.

Levoy and Hanrahan [6] introduced the Light-Field: Image Based Rendering without known geometry from a dense grid of cameras. This system was enhanced in many directions later on. While the scene geometry for the lightfield was assumed to be a plane, Gortler et. al. [4] started to use an approximative 3D shape for the Lumigraph.

View interpolation using implicit geometry has been presented in [3]. Starting with calibrated images, depth maps are computed offline and new views are generated by texture warping with a view dependent surface. The quality of the generated images heavily depends on the quality of the depth maps which have to be accurate and must be densely filled.

Yang et. al. [10] presented a system for view

interpolation using no geometry. Image interpolation is performed using a plane-sweep algorithm by finding photo consensus between surrounding real views. In [11] they extend the system to multi resolution depth estimation. Our approach is similar to their work, in fact it combines some aspects of depth estimation and consensus based view generation.

A view interpolation system with high image quality was introduced by Zitnick et. al. [12]. Much effort is taken to precompute accurate depth maps and separate regions arround depth discontinuities into a special layer.

Recently, Nozick et. al. [7] developed a real-time plane-sweep view interpolation system which can handle occlusions using a statistical method and does not require any preprocessing. This suits it well for dynamic scenes and free viewpoint video.

Our approach is a combination of those systems. It precomputes depth maps but in a fast and approximative manner. It does not rely on the accuracy of the depth maps alone but uses them to guide the visual reconstruction.

3 Standard Plane-Sweep

Before describing our modifications, we first introduce the plane-sweep algorithm used for depth estimation and reconstruction [1].

3.1 Reconstruction

Starting with two real views *left* and *right* defined by their projection matrices P_l and P_r and associated images I_l and I_r, the goal is to generate a new view for P_v, the projection of the virtual camera. Figure 1 depicts this situation.

The projection matrix P projects the homogeneous 3D point X into a 2D image point x:

$$z \cdot x = PX = K[R^T | - R^T C]X \qquad (1)$$

with z the projective depth. Following the pinhole camera model P can be decomposed in the intrinsic camera parameter matrix K, the rotation R and the position of the camera center C.

Having only one 2D image point x and P, the 3D point X can not be reconstructed due to the missing depth z. But having a pair of images and projections and assuming the known correspondence of the 2D

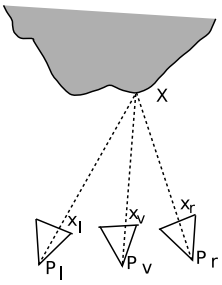


Figure 1: The image point x_v has to be interpolated from the observations x_r and x_l of the point X on the surface of the object.

image points x_l and x_r, the point X can be triangulated. The major problem is to find this correspondence. Many different algorithm has been proposed to solve the correspondence problem. Nearly all of them use a *matching function* to compare points or regions in images based on their intensity values. The underlying assumption here is that any point X projects into every image I_i with the same intensity and color (lambertian surface).

3.2 Correspondence Search with Plane-Sweep

The plane-sweep algorithm is designed to find correspondences between two images by evaluating the matching function for each pixel in a given interval and finally chosing the minimum and its associated 3D point as the correspondence. Evaluation and sampling of the matching function of all pixels is done by placing a plane Π_i in 3D space and projecting all image points of the both views onto this plane and computing the difference between the intensity from the left and the right view. This is equivalent to back projecting a 2D x_l image point from one view to the 3D point X_z on the plane Π_i:

$$X_z = C + z\frac{RK^{-1}x_l}{||RK^{-1}x_l||} \text{ with} X_z \in \Pi_i \qquad (2)$$

Projecting X_z into the second view with (1) yields a potential correspondence in x_r. For all potential correspondences the matching d_i is computed and stored together with z, the projective distance of X_z from the camera.

The algorithm proceeds by placing a new plane Π_{i+1} with a given distance coplanar to the Π_i and

again computing the matching. By iterating over n planes, for each pixel the matching d_i and the associated z is a sample of the matching function $d(z)$. The minimum $\min(d(z))$ can be assumed to be the best correspondence for the given views and the evaluated planes. The quality of the matching heavly depends on the number and positions of the planes. When dealing with discretised images the planes have to placed such that the minimal and maximal pixel disparity is matched and the spaces between planes should match the disparity steps.

Figure 2 show a schematic of the correspondence search for two views. Using projective texture mapping, a plane-sweep algorithm is well suited to be implemeted on graphics hardware [10].

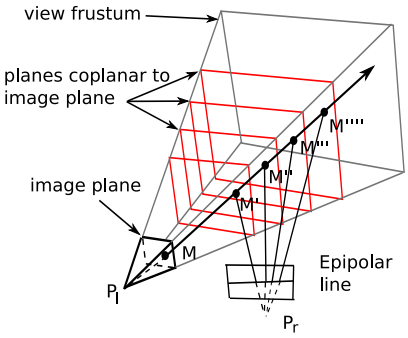


Figure 2: Correspondence search along epipolar line and with plane-sweep.

The approach is easily extended to compare more then two images simultanously.

3.3 Matching Function

Like most algorithms for correspondence computation, the plane-sweep algorithm compares intensity values of images. This can be done in various flavours. The most simple comparison is the absolute difference between two values, one from each image. It can be shown that this matching function is not robust against noise in the images, variations in exposure and especially non-lambertian surface properties. More advanced matching function take a region around the x_r and x_l into account. The sum of squared differences (SSD) penalises larger differences and is computed over a correlation window of size $[2w+1, 2h+1]$ around x_l and x_r:

$$d_{\text{SSD}} = \sum_{j=-h}^{h} \sum_{i=-w}^{w} (I_r(x_l+(i,j)^T) - I_l(x_l+(i,j)^T))^2 \quad (3)$$

More complex matching functions are known but these are expensive to be computed and thus can not be used for the purpose of realtime matching.

3.4 Problems

Using d_{p} as matching function often results in mismatches due to multiple local minima. An example is shown in figure 3. X_1 and X_3 have a similar appearance in both images leading to a match on plane E_2 which occludes the match in E_1 from the point X_2.

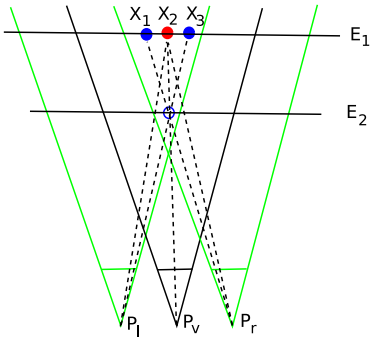


Figure 3: A mismatch from a local minimum on E_2 occludes the match in E_1 from the red point.

Mismatches and local minima can be reduced by using different matching functions and larger correlation windows. But correlation windows bring in problems on their own at depth discontinuities and perspective distortion between two cameras can disturb the matching. Figure 4 shows an example where the appearence of a region $M_1M_2M_3$ changes between P_1 and P_2 due to different perspectives.

The combination of a small correlation window and a hierarchical matching using a resolution pyramid can help to solve this. Starting from the original image a new pyramid level is constructed by low-pass filtering and subsampling the image. Each evaluation of the matching is done for every pyramid level and the results are sumed up. This takes a region around the potential correspondence into

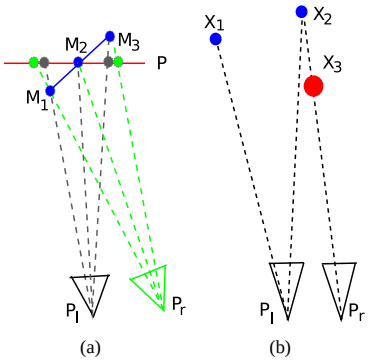


Figure 4: (a) Perspective distortions of points $M_1M_2M_3$. In P_1 the projections are closer together than in P_2. (b) Occlusion leading to a mismatch.

account without the need for complex correlation computations. As proposed in [11] this can be done on current graphics hardware by using MipMap textures as resolution pyramid. One problem of MipMap is, that those are only an approximation auf low-pass filtered images. MipMap levels are generated by averaging four adjacent pixel leading to boundary problems between two groups of four pixels.

Another problem results from occlusions. Points seen by a camera P_1 and occluded in camera P_2 can not be matched properly (figure 4). This can only be resolved by using additional views for matching.

4 Extended Plane-Sweep for View Interpolation

The described standard plane-sweep algorithm can be used to interpolate images by accepting the best correspondence per pixel and combining the color information from both images at the corresponding points. No geometry is required, but mismatches result in visible artefacts. To ensure an appropriate reconstruction, the number of planes has to match all possible disparities.

On the other hand, depth information is computed by the position of planes and can be transformed into depth maps or other representations suitable for rendering. But the problems remain the same and depth maps can not be precomputed for any possible view.

Our view interpolation approach combines depth estimation and plane-sweep view generation to achieve two major goals:

1. increase the quality and precision of the matching,
2. render with interactive frame rate.

4.1 Overview

In a preprocessing phase, for each real view a depth map is computed using all available real view for matching. This is described in detail in section 4.2. The depth maps serve as starting points and search range limitation for realtime correspondence search during rendering. Each depth map is adaptively subdivided into a quad-tree based structure with regions of consistent depth [2].

For each new view to generate, the nearest real view is selected as reference view and 3 or 7 neighbour views are selected as support views. Thus interpolation is done either from 4 or from 8 views.

Based on the depth map and the quad-tree of the reference view, tiles are created. For each tile, a depth interval is determined and this tile is used as a limiting plane for a plane-sweep algorithm to compute the pixel values of the new image.

By tiling the plane and performing partitial sweeps, the number of matching evaluations is reduced dramatically and the sampling density for each sweep can be increased at low extra costs. At the same time, mismatches due to local minima are avoided.

4.2 Depth Estimation

The initial depth estimation is a preprocessing step. This has the advantage that a more complex matching algorithm can be used and the images of all real views can be taken into account. The main focus here is the completeness, not the absolute precision and reliability of the depth of each pixel. The algorithm is similar to that proposed in [11] and to compute a depth map for the real view I_r with P_r works as follows ([9]).

- The virtual camera is set to match P_r.
- Chose k other views as support views.
- For each support view $s, 0 \leq s < k$
  - Images I_r and I_s are selected as textures.
  - Projection matrices P_r and P_s are set to generate texture coordinates.
  - For each of N planes

- ∗ The matches between I_r and I_s are computed by rendering the plane. Use d_{SSD} with 3x3 window as matching function.
- ∗ For each pixel: Compare matching with best match so far. If current match is better then best match, update best match with current match. Remember plane number of best match per pixel.

– Store 2D array M_s with plane numbers from best matches for this P_s

- Per Pixel: Process all $M_s, 0 \leq s < k$ and find the most probable depth by evaluating the distribution of best matching planes in space.

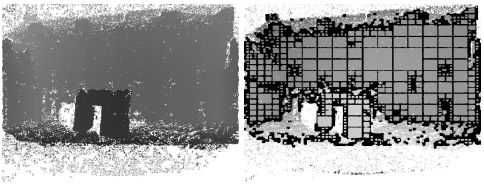


Figure 5: Example of a computed depth map and the tiling (Castle scene).

After depth estimation, all depth maps are divided into regions of similar depth by a quad-tree, starting with a region size of 64x64 pixels. In each region, the minimal (z_{min}) and maximal (z_{max}) depth is computed and the difference is compared to the global difference in depth z_δ:

$$z_{\text{max}} - z_{\text{min}} < \alpha z_\delta \tag{4}$$

The factor α is typically chosen to be $1/10$. If this evaluates to true, the tile is accepted, otherwise it is subdivided into 4 tiles. The recursive refinement stops when the size of a tile is 2x2. If even this 2x2 tile has too much variation in depth it is rejected and depth information for this region is discarded. In figure 5 the estimated depth map and the tiling is shown. The original image of the castle is shown in figure 7.For each tile the search range in depth for the following reconstruction is determined by $[z_{\text{min}} - z_\epsilon, z_{\text{max}} + z_\epsilon]$ with z_ϵ a small offset to compensate for depth estimation errors (figure 6).

4.3 View Generation

New views are generated by a piece-wise plane-sweep algorithm guided by the previous depth estimation. For this purpose, one reference view and 3 or 7 support views are selected.

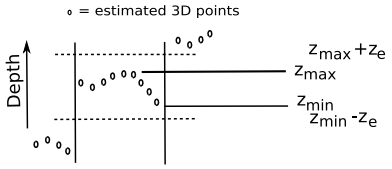


Figure 6: Search range in a region with similar depth.

The first step in view generation is the ranking of all real views according to their distance and angle to the new virtual view to interpolate. Then the best ranked real view is chosen to be the reference view and the 3 or 7 next best views are chosen as support views. All selected views are set as textures for projective texture mapping.

Based on the tiled depth map of the reference view, for each tile a quad is constructed and placed in 3D space coplanar to the virtual camera. The distance to the virtual camera is chosen given $z_{\text{min}} - z_\epsilon$. Then this quads is successively moved towards $z_{\text{max}} + z_\epsilon$ in αN steps of $(z_{\text{max}} - z_{\text{min}} + 2z_\epsilon)/(\alpha N)$ size.

The matching function used here is a 1x1 d_{SSD} without MipMap. Due to the limited search range of the tile sweep, severe mismatches are avoided. The d_{SSD} can be computed efficiently on the GPU using the *dot()* operation which computes the dot product of two vectors.

For every plane and every pixel in the new view, the matching from 4 or 8 views has to be merged and an intensity or color values has to be computed. When using 4 real views, 3 matches are computed. From these 3 matches the best two are chosen and the third is discarded to cope with occlusions. The remaining two matches and their RGB values are blended according to the distance of the associated cameras to the virtual camera. This ensures that the color changes smoothly over time when moving the virtual camera from one real view towards another one. At the same time, the removal of one view reduces the effect of views with partial occlusions significantly.

When using 8 real view, 7 matches are computed and mismatches are removed by a statistical approach. From 7 matches, the mean and the variance is calculated. All matches with distances to the mean greater than a given threshold are removed. This is repeated until only two matches are left or

no more matches are above the threshold. The final result is then blended as describe above. Details about this method can be found in [7].

4.4 Advantages of our Approach

The described view generation approach brings some advantages. In the preprocessing phase, the plane-sweep is performed for the view of the real camera which is chosen as the reference view for matching. This results in more stable matching compared to a virtual view not located at a real view. In addition, no time contraints apply and all available real views can be incorporated to fill holes and handle occlusions. A matching function d_{SSD} with 3x3 correlation window and no MipMap has shown to give the best results.

In the online phase, interactive frame rates should be reached. The matching has to be done with a small correlation window (1x1) and the number of planes which can be evaluated is limited. The precomputed depth maps allows adaptive limiting of the final sweep and therefore the search range, without impact on the precision and matching quality. For example, given a scene is properly reconstructed by 200 planes in a traditional sweep, it suffices to uses 20 different levels per tile.

Local minima of the matching function can only influence the view if they occur within the search range. Limiting the search range reduces the probability of local minima considerably leading to more robust matching and less visible errors.

5 Results

The described view generation approach has been implemented using C++ and Cg, a shader language from Nvidia. All results have been produced on a 3.2 GHz PC with a NVidia Geforce 6600 GT.

We compared our method to those from Yang [10] and Nozik [7] using 4 real views and the following configurations which proved to give the best results so far:

short name	correlation window	method from	resolution level
3Y1	3x3	Yang	1
1Y2	1x1	Yang	2
3N1	3x3	Nozick	1

To quantify the quality of the reconstruction, the number of pixel errors per image are counted. A pixel error is counted if the difference compared to the ground truth image of one color channel or of the intensity value exceeds 10. Regions which have only been seen by one real view or which have not been seen can not be reconstructed properly. These regions are subtracted from the results. Figure 7 shows the visibility labeling for a view of the castle scene.

5.1 Synthetic Scene

This implementation has been tested on a variety of scenes. The castle scene consists of 200 images of a 3D model from 4 horizontal scans in different heights. Synthetic data allows to analyse the algorithm without interference from noise, calibration errors, lens distortion or exposure changes. The castle scene consists of a textured background showing a castle, a textured ground plane and a textured arch occluding parts of the background. All parts are 3D models from a reconstruction, therefore the geometry does not consist of perfect planes. Four views of this scene have been taken along with the depth maps. Figure 7 shows an image of the sequence and the visibility in the four views.

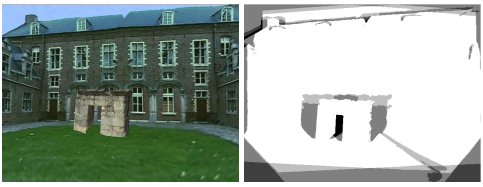


Figure 7: Left: Original image of the castle scene. Right:Visibility in real views as seen from the novel view (White=4,greys1-3,balck=0).

Experiments showed, that for the castle scene 200 planes are sufficient for view generation. Using only 50 or 100 planes results in more errors, but more then 200 planes does not improve the image quality any more.

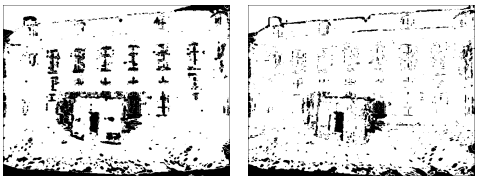
In table 1 the relative numbers of pixel errors are given. In image regions seen by only 2 real views our method produces 38% false pixels which is only slightly better then the other methods around 45%. But in regions seen by 3 cameras, the error drops from 25% to 13%. For regions observed by all 4 chosen views, we can reduce the number of errors

to 4%. Of all regions seen by at least 2 cameras (denoted as full image), we can reconstruct with an error rate of 7%. The next best method is 3Y1 with 13% pixel error. But 3Y1 needs more then 2 second to achieve this result while our method only needs 166ms yielding 6 frames per second.

The distribution of the pixel errors is shown in figure 8. All pixels which differ from the reference view by at least 10 levels are marked black. The center of the arch and some borders can not be reconstructed.

method	3Y1	1Y2	3N1	Our
seen by 2 views	46%	46%	44%	38%
seen by 3 views	25%	25%	25%	13%
seen by 4 views	8%	16%	11%	4%
full image	13%	18%	14%	7%
time [ms]	2150	401	1360	166

Table 1: Pixel errors of different reconstruction methods based on the plane-sweep approach and the computation time.



(a) 3Y1 (b) Our depth guided method

Figure 8: Pixel errors distribution in comparison (white=good, black=bad).

As an example of how the search range limitation affects the reconstruction, the matching function for one pixel is plotted over 200 planes in figure 9. There are two local minima clearly visible and 3Y1 as well as 1Y2 and 3N1 chose the minimum around plane 119. By limiting the search range to $[50, 66]$ our method choses the minimum around plane 51, which is the right one.

Without the depth guided matching, best results can be achieved with a 3x3 correlation window. A matching with 1x1 requires a hierarchical matching or otherwise produces too many mismatches. Computing the matching function in a 3x3 region is expensive in terms of GPU cycles. It requires 9 times more texture look-ups than single-pixel

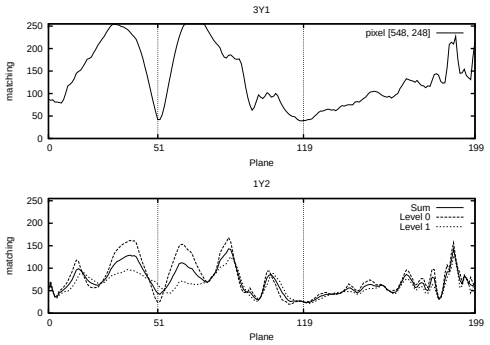


Figure 9: Matching function for one pixel. In 1Y2 the matching for MipMap level 0, 1 and the sum is shown.

matching leading to longer computation times. Using our depth guided matching, different matching functions produce only slightly different results. It suffices to compute single-pixel matching without MipMaps. Combined with the reduced number of depth evaluations this results in the performance of 10-15 fps.

5.2 Real Scene

To verify the usefulness of our method we reconstructed a more complex scene captured with a TV-camera in four different heights. An image of this studio scene and a depth map is shown in figure 10. The precomputation of the depth maps typically takes around 3-4 seconds per view, depending on the number of support views and the number of planes to use. For the studio scene we chose 500 planes and 4 support views to compute depth maps for all real views.

Figure 10: The studio scene, original image and depth map. In some regions the depth estimation failed

As seen in figure 10 the depth reconstruction does

not give sufficent stable result for the complete image. Most problems occur in homogenous regions. During the construction of the quad-tree the algorithm decides not to create search range limitations for these regions due to the large variance in depth. For the visual reconstruction these areas have to be sweeped applying the complete search range. But due to their homogenous appearance, mismatches from local minima remain invisible. In figure 10 the visual quality of the reconstructed image is better then the error image suggests.

(a) interpolated image (b) pixel error 12%

Figure 11: Result of our method for the studio scene. See also in the colored appendix.

The details about the reconstruction of the studio seen can be found in table 2. Again it showed that without depth guided matching best results can be achieved with a 3x3 correlation window leading to non-interactive frame rates. With our method we can increase the quality and maintain interactive rates even if for some image regions where no proper depth is available.

method	3Y1	1Y2	3N1	Our
full image	16%	20%	17%	12%
time [ms]	1280	262	820	175

Table 2: Results of different reconstruction methods based on the plane-sweep approach.

6 Conclusion

We presented an efficient view interpolation approach which incorporates depth information and photo consistency. By computing approximative depth maps offline, these are used to limit the search space for the plane-sweep view generation. This reduces mismatches and enhances the performance considerably.

Acknowledgements

We want to thank the IST project *MATRIS* (nr. IST-002013) for the studio scene.

References

[1] Collins, R., *"A space-sweep approach to true multi-image matching"*Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pp. 358–363.

[2] Evers-Senne, J.F., Koch, R. *"Image Based Rendering from Handheld Cameras using Quad Primitives"*Proceedings of VMV 2003, Munich, Germany

[3] Evers-Senne, J.-F.and Koch, R., *"Image Based Interactive Rendering with View Dependent Geometry"* Proceedings Eurographics 2003, Granada, Spain, 573-582

[4] Gortler, S. J., Grzeszczuk, R., Szeliski, R. and Cohen, M. *"The Lumigraph"*, Computer Graphics Vol. 30, 1996, 43–54

[5] Koch, R. and Evers-Senne, J-F. *"View Synthesis and Rendering Methods"* in: "3D Video Communications", Wiley, 2005

[6] Levoy, M. and Hanrahan, P. *"Light Field Rendering"*, Computer Graphics Vol. 30, 1996, 31–42

[7] Nozick, V., Michelin, S. and Arques D.", *"Real-time Plane-Sweep with local startegy"*, Proceedings of WSCG 2006, Plzen, Czech Republic

[8] Shum, H.-Y. and Kang, S. B., *"A Review of Image-based Rendering Techniques"*, Proceedings Visual Communications and Image Processing, 2000, 2-13.

[9] Woetzel, J. and Koch, R., *"Multi-camera real-time depth estimation with discontinuity handling on PC graphics hardware"*, Proceedings ICPR 2004

[10] Yang, R., Welch, G. and Bishop, G. , *" Real-Time Consensus-Based Scene Reconstruction using Commodity Graphics Hardware"*, Proceedings of Pacific Graphics, 2002, Beijing

[11] Yang, R. and Pollefeys, M., *"Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware*, CVPR 2003, Madison (WI), USA

[12] Zitnick, L., Kang, S.B., Uyttendale, M., Winder, S. and Szeliske, R., *"High-quality video view interpolation using a layered representation"* Proceedings of ACM SIGGRAPH 2004, 600–608

(a) Original (b) Interpolated

Figure 12: Images of the castle scene

(a) Original (b) Interpolated

Figure 13: Images of the studio scene