Multi-frame Rate Augmented Reality

Philipp Grasmug*
Supervised by: Dieter Schmalstieg†

Institute of Computer Graphics and Vision
Graz University of Technology
Austria

Abstract

In this work we present a method for improving the visual quality of an augmented reality system. By combining the characteristics of two different sensors, we increase the spatial resolution of a video stream using sub-pixel accurate image registration. Using the optical flow to solve the correspondence problem, we can estimate the depth of the scene, which is further used to compute immersive augmented reality effects. By decoupling the rendering process of the augmented information from the displaying frequency of the system, we can augment the scene using computationally expensive rendering techniques. We utilize image-based rendering to overcome the resulting temporal artifacts. Finally, we evaluated our methods by comparing the achieved quality with conventional augmented reality methods.

Keywords: augmented reality, interactive superresolution, image registration, workload distribution

1 Introduction

Increasing the spatial resolution of images is normally achieved by developing new sensor chips for cameras with a higher pixel density or larger sensors. Both of these improvements have drawbacks in terms of image quality (smaller pixels mean less light, leading to more noise) and efficiency. Also, capturing images at a high resolution (HR) means that a lot of data has to be transferred, which requires a bus with sufficient transfer rate.Alternatively one can combine information from multiple subsequent images of the current scene or arbitrary images from a database to compute a realistic or at least plausible high resolution version of the input image. We combine data from two sensors captures at different temporal and spatial resolution to create a true high resolution output. The hardware setup of our system looks like this: The first sensor provides a video stream in LR, but at a high frame rate (e.g., 30 Hz). A second sensor, which is placed right next to the first one, captures still images in HR at a slow frame rate. The latest high resolution image is registered to the current low resolution video frame to compute the desired output.

The information visualized in augmented reality (AR) applications ranges from simple textual information to computationally expensive visualizations, which can not be computed in real-time. If a high resolution is required. By combining images from different sensors or from a single sensor captured in different spatial resolutions, we can achieve interactive super-resolution of the input video stream. To match the high resolution of the input, we present a strategy for distribution the workload of computationally expensive rendering tasks over several frames using image-based rendering to suppress temporal artifacts. For both tasks we utilize the GPU to speed up the computation. The image registration needed for our super resolution approach can be computed highly efficient on programmable graphics hardware while the use of a GPU for rendering the augmentations is self-explanatory. Figure 1 shows a overview of the main parts of our system which are described in detail in the following sections.

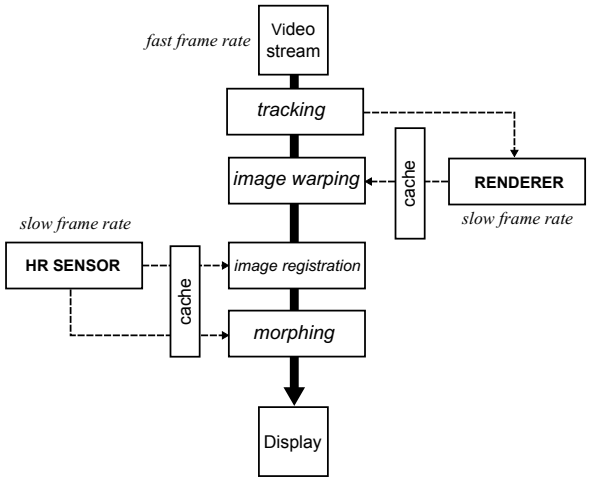

Figure 1: Overview of the main parts of our system.

*grasmug@icg.tugraz.at
†schmalstieg@icg.tugraz.at

2 Previous Work

Improving images in terms of spatial resolution is relevant in many fields and applications. Super-resolution (SR) is an algorithmic approach that combines information from several images or from large image databases into a true or at least a plausible high resolution version of the low resolution (LR) source. Different approaches in the spatial and the frequency domain have been proposed.

Algorithms in the frequncy domain exploit the aliasing of LR images to reconstruct a HR image. Huang and Tsai [10] presented an algorithm that built upon the relative motion between the LR images. The method utilizes the aliasing relationship between the continuous Fourier transformation of the HR image and the discrete Fourier transformation of the LR images.

Example-based techniques (also called image hallucination) [6, 5, 1] try to model the relationship between LR and HR images using corresponding patches of LR and HR images that do not necessarily show the same scene. The correspondence between those pairs is learned from a large database and used to synthesize a plausible HR image. Bhat et al. [3] proposed a system similar to ours that uses static images to enhance a video of the same scene. The method computes view-dependent depth data for the images and the video and uses this information to apply a variety of effects including super resolution as an offline post-processing step. Known SR algorithms rely on optimization and/or machine learning which makes them slow and not usable in interactive applications like augmented reality. Our algorithm in contrast is based on optical flow computation and allows for interactive use.

Render caching reuses information from previous frames to speed up the computation of the current one. When the frame rate is high, the changes from a previous frame to the current one are small and therefore, the temporal coherence is high. This information can then directly (e.g., color of a pixel) or indirectly (e.g., intermediate results) be reused.

The term *render cache* was introduced by Walter et al. [17], who used it as a data structure to speed up rendering of otherwise none interactive methods. Yu et al. [18] proposed a GPU implementation of the forward reprojection algorithm, which uses a per pixel disparity vector to compute the new position. Implementing this approach is difficult and can be computationally expensive. Didyk et al. [4] proposed an efficient method, which fits a coarse, regular mesh grid to the cached image. The mesh is aligned to the depth discontinuities of the cache. Each vertex is warped to its new position using the cached image as texture. Holes are avoided by stretching of the mesh and visibility is resolved by fold overs.

The applications for render caching are numerous. Sitthi-Amorn et al. [16] proposed an algorithm for the acceleration of pixel shader computations by directly reusing information from the cache whenever available. A relevant work on spatio-temporal upsampling of renderings was published by Herzog et al. [9]. By combining multiple low resolution renderings, using a modified joint-bilateral filter, a high-resolution image can efficiently be computed. Render caching focuses on reuse of results from previous frames for rasterization based techniques while our workload distribution scheme targets oversampling based methods and exploits their properties. This allows different approaches which cannot directly be applied to rasterization.

Image-based rendering The general idea of image based rendering (IBR) is to derive a novel view from real or synthetic images. The usual way of rendering an image of a synthetic scene is by using one of the standard algorithms like, for example, rasterization, ray tracing or path tracing. Image warping [13, 14] relies on associated per-pixel depth information. This information together with the position and orientation of the camera is used to reproject each pixel into three dimensional space and then into the view of the new desired virtual camera. For synthetic scenes, this is straight forward, since storing per-pixel depth values and camera positions can be easily done during rendering. IBR is utilized in our work to compensate temporal artifacts. We further employ a method, again closely related to oversampling based rendering approaches, to address IBR related issues like resampling and disocclusion artifacts.

3 Upsampling

In this section, we describe our upsamling strategy, which is inspired by previous work in the field of SR. We present an approach that combines HR images and an LR video stream from one ore multiple sensors to compute a SR version of the LR input online. At the point in time at which the still frame is captured, both sensors show a nearly identical images of the scene (with slight differences due to the spatial placement). Over time the images begin to diverge due to the motion of the camera or the scene. To be able to use the information from the high resolution image, we have to correct this divergence. We do this by computing the optical flow between a subsampled version of the latest high resolution image and the current frame of the video stream. The resulting sub-pixel accurate displacement map is used to transform the HR image to match the current video frame. Our algorithm can be outlined by the following steps, where I_{LR} denotes the current video frame and I_{HR} the latest HR image.

1. Compute the optical flow from I_{LR} to I_{HR} and vice versa. The computation is done at the resolution of I_{LR}.

2. Compute the confidence for every pixel.

3. Upscale the flow field using bilinear interpolation.

4. For each pixel, lookup the color of I_{HR} using the computed flow field from I_{LR} to I_{HR}.

5. Blend I_{HR} with I_{LR} image according to the confidence map.

The result of the optical flow computation is crucial for the quality of the SR and mainly depends on the disparity between the pair of images, which is influenced by spatial placement of the sensors, camera motion and motion in the scene. Another associated problem is occlusion and disocclusion of objects in the scene. Due to the change in viewpoints over time, areas of the scene get revealed which are not visible in the latest HR image. This is obviously an issue for the image registration step, especially if the scene has a high depth complexity since objects are more likely to occlude each other. Therefore, we have to expect a certain error in the optical flow computation.

3.1 Confidence Map

Figure 2: Visualization of the confidence map. Red shading denote regions with low confidence like, for example, the area around the bottle, which has been disoccluded due to the movement of the camera.

In order to be able to compute an artifact-free upscaled version of the video stream, we have to account for the mentioned problems. In detail, this means that we have to detect occlusions and regions, where the computation of the flow field yields erroneous results. Common metrics like *endpoint difference* [15] and *angular difference* [2] are not suitable for our case, since they are not reference free. To establish this property, we modeled our metric based on a simple observation. If we compute the optical flow from image I_1 to I_2 and vice versa, the flow vectors should approximately be the same with inverted sign.

$$uv_{fw}(p) \approx -uv_{bw}(p + uv_{fw}(p)) \tag{1}$$

Equation 1 formulates this observation, where uv is the 2-component flow vector with respect to the image position $p = (x, y)$. Based on this equation, we derived a metric which tells us whether the flow vector at a certain position is correct or not.

$$conf = 1 - \lambda\left(\frac{|uv_{fw}(p) + uv_{bw}(p + uv_{fw}(p))|}{\alpha}\right) \tag{2}$$

$$\lambda(x) = \begin{cases} x & \text{if } x \leq 1 \\ 0 & \text{else} \end{cases} \tag{3}$$

We call this metric *confidence* and compute a map for the whole flow field (Figure 2) as defined in Equation 2, where α is a weighting factor that defines how strongly the difference is penalized. The fiducial marker in the shown image is later used to augment the scene but not for the SR approach. The formula is based on the distance between the two flow vectors interpreted as points and ideally is zero. The confidence map is recomputed every frame. This also means that we not only have to compute the flow once, but twice each frame, to be able to compute this quality metric.

To finally create the HR output stream we blend the LR image with the warped HR image weighted by the confidence map. Since the flow field is usually good in high frequency regions of the image, we can preserve those important details. Low frequency regions tend to yield worse results in terms of optical flow computation. Using information from the LR frame in those areas means only a negligible loss of information. In the case of disocclusion, no information is available for this region, which means blending with the LR image is the only meaningful solution.

3.2 Depth Estimation

With the camera extrinsics given from the tracking (fiducial markers in our case) and the intrinsics from the calibration we further need correspondences to be able to reconstruct depth by triangulation. Using the optical flow, we can compute pixelwise point correspondences. Having all this information at hand, we now can calculate a depth estimation of the scene. Using the current video frame and an older video frame with a certain delay as key frames for the estimation, we can compute the depth with the algorithm outlined in the following:

1. Compute a ray from the center of projection into the scene for both views. This ray is given by $w = c + \lambda Q^{-1} m$, where $m = [u, v, 1]^T$ is a point on the image plane and Q is defined in equation 4. The center of projection is given by $c = -Q^{-1}\tilde{q}$.

$$P = A[R|t] = \left[\begin{array}{c|c} \mathbf{q_1} & q_{14} \\ \mathbf{q_2} & q_{24} \\ \mathbf{q_3} & q_{34} \end{array}\right] = [Q|\tilde{q}] \tag{4}$$

   P_1, P_2 as well as c_1, c_2 are obtained from associated extrinsics $[R|t]$ and intrinsics A of the chosen key frames.

2. Intersect the rays. Since rays usually do not intersect at a point in $\mathscr{R}^3$, we need to computed the shortest

line between the rays. If the length of that line is below a certain threshold, it is treated as intersection.

3. From the intersection point (or the starting point of the shortest line between both rays), we can retrieve the reconstructed depth of the scene in that particular point.

Figure 3: The top image shows the computed depth map for the reference image below.

The selection of the pair of input images is essential for the depth estimation. To obtain good results, the images must have a appropriate stereo baseline. The result further depends on the correctness of the found correspondences. We need to compute the optical flow between the latest HR image and the current LR image every frame for our super-resolution algorithm. Therefore, we already have correspondences, which could be used for the depth information. Whenever the HR image is refreshed, it is nearly the same as the current image. In this case, the stereo baseline vanishes, and we cannot use it for depth estimation. Therefore, we cache the latest N frames of the LR video stream and use one out of it for the stereo matching. The image from the cache is selected either with a fixed frame distance or using an angular threshold. For both cases this method fails, if the camera movement stops. To resolve this issue, a keyframe based approach should be used in the future. Figure 3 shows the resulting dense depth map. While our SR approach is applicable to dynamic scenes too, the depth estimation is limited to static scenes.

4 Workload Distribution

In this section, we describe a workload distribution scheme that allows to compute expensive effects in real time by decoupling the rendering process. In contrast to render caching we focus on oversampling based approaches like path tracing which allows us to employ a different strategy.

To decouple the rendering process from the displaying frequency of the system, we discuss two strategies: First the computation can be *time sliced*. Only a subset of all samples per pixel is computed each frame. The final image is available after a number of frames depending on the size of the subset. A similar technique is to *spatially slice* the image by computing a sub-region of the rendering each frame with the full number of samples. Since the objects in the scene are most likely not distributed uniformly, this method can result in an unsteady frame rate. With both strategies, the final image is available after a distinct number of frames depending on the splitting criterion and on the desired total number of samples. Figure 4 illustrates both strategies.

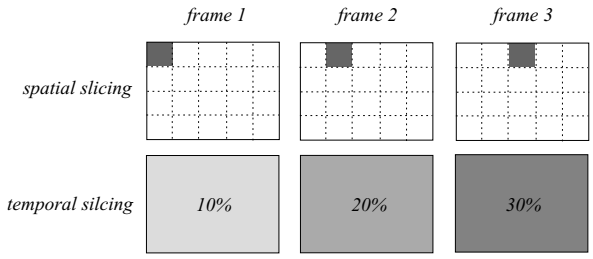

Figure 4: Spatial versus temporal slicing. In the upper row, the shaded rectangle denotes the region which is computed at frame n. In the lower row, the shading denotes the percentages of samples per pixel which have been computed to this point.

For our experiments, we implemented a path tracer to augment the scene. This algorithm is just an example of the rendering methods that can be used. Since image-based rendering, in our case image warping, only relies on per-pixel color and depth information, essentially any rendering technique can be used. The workload distribution strategy has to be chosen accordingly.

4.1 Image-based Rendering

To overcome the difference in frame rate, which results from decoupling the rendering process, we utilize image-based rendering techniques. Per pixel depth information

can easily be stored during the rendering process. In combination with the matrices used for rendering and the current camera matrix, we can re-project every pixel to derive a novel view of the scene. Some issues related to this approach have to be addressed in order to produce satisfying results. The first issue is one we have already encountered with super-resolution of the video stream. Since image warping uses a static image from a different viewpoint than the current, disocclusion is again a topic. A common way to address this issue is to cache or even render different views in a preprocessing step [8]. In addition to the latest rendered image, an appropriate stored view can be used to fill disoccluded areas. In contrast, we employ hole filing strategies to account for this issue.

Not only disocclusion, but also occlusion is a problem, which needs to be handled. If parts of the rendered object occlude other parts due to the change in camera position, different pixels are warped to the same location. This results in a *depth fighting* like behavior. Another problem that comes with this approach is sampling related. If the camera in the derived view is closer to the scene than in the original one, the image is sampled at a higher rate, leading to cracks in the novel view.

The straightforward way for implementing image warping would be to multiply each pixel and its depth position by the inverse projection and model matrix used for rendering and then again by the projection and model matrix of the current view, like given in equation 5, where P is the projection matrix, C_{ref} is the camera matrix used for rendering, $C_{current}$ is the camera matrix of the desired view and p is the position of the pixels given in homogeneous coordinates.

$$p' = P \cdot C_{current} \cdot C_{ref}^{-1} \cdot P^{-1} \cdot p \tag{5}$$

The problem of this approach is that it is likely that different pixel are projected onto the same output position. Since we use parallel warping on the GPU, this results in undefined behavior. We implemented the image warping in the following way to resolve the described problems: In a first step, only the depth value of every pixel is warped into the desired view. We resolve the described race condition by using atomic operations to store only the depth values closest to the camera. This is similar to the z-buffer algorithm. In the next step, the depth and position of each pixel is used to look up the color in the original image. This is done by performing the operation given in equation 5, in the reverse direction. Using this multi-stage warping, we can efficiently eliminate resampling artefacts and occlusion.

4.2 Hole Filling

In the final step, we aim to fill small holes and cracks which result from the image warping. The simple approach is to interpolate holes from neighboring pixels iteratively or using push-pull interpolation [12].

Another approach is to re-render the disoccluded areas to fill the holes an cracks. To detect disoccluded regions, we first render only the silhouette of the model from the current viewpoint. Second, we warp the image and subtract the result from the silhouette which gives us the disoccluded region. Now, the pathtracer is instructed to recompute only the identified area with a small number of samples in order to keep the computational expense low. Thus, we can efficiently fill all holes with the disadvantage of discontinuities between the original rendering and the re-rendered area due to the small sample count. Figure 5 shows a comparison of the two hole filling methods. The re-rendering produces good results in any case, while the computational expense is again related to the size of the disoccluded area and is higher compared to interpolation. The interpolation works well as long as the holes are small.

5 Results

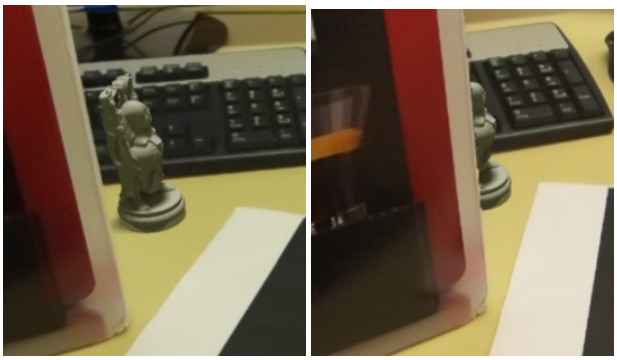

Figure 6: Occlusion of a virtual object (Buddha statue) by the real world using the estimated depth information.

We used three different scenarios to evaluate the SR method. Two of the data sets show office scenes, while one shows an outdoor scene. All scenes have different complexity in terms of depth, depth range and texture. As metric for similarity, Hdr-vdp-2 is used [11] to show that the results of our algorithm are similar to the reference image and free of artifacts. To account for image sharpness, the reference free LPC-SI [7] metric is used. The input video stream was captured at 640x480 pixels (0,3 megapixels) with 30 frames per second (fps) while the HR still images had a resolution of 2304x1728 (4 megapixels) pixels captured with 8 fps. The system was evaluated on an Intel Core i5 (3.4 GHz) with 8 GBs of memory and a Geforce GTX 680. For capturing the test data, a Canon IXUS 240 HS was used. Figure 7 shows average scores for Hdr-vdp-2 (similarity to the ground truth in percent) and LPC-SI (higher score means more sharpness) of all three scenes. It can be seen that the sharpness of our method is close to the ground truth and way above what bilinear

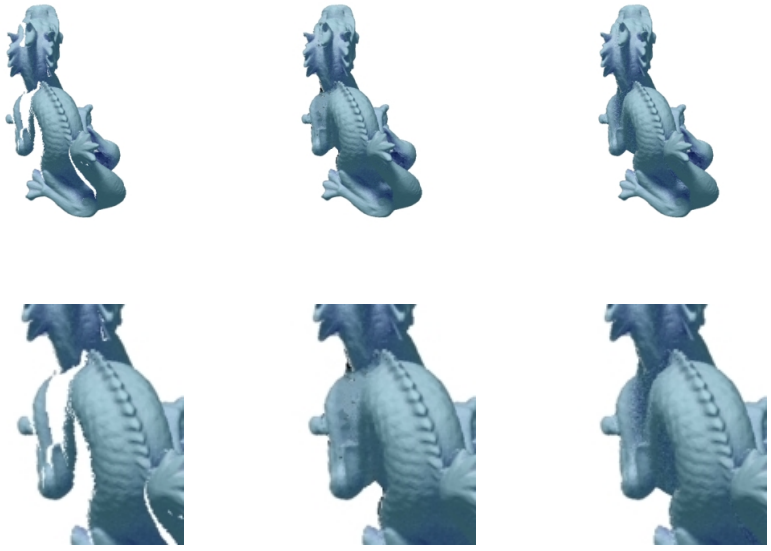

Figure 5: Comparison of the two hole filling methods for different warping angles. The first column shows the warped image without hole filling, the second with color interpolation and the third with re-rendering of disoccluded areas. The bottom row gives a detail view.

interpolation gives us. The online SR resolution runs at 7 fps (average) on the reference system.

Using the estimated depth, we can now let real objects in the scene occlude augmented virtual objects to generate a more immersive AR experience. This is done using an CUDA kernel, which compares the depth of the virtual scene with the estimated depth of the real scene and blends the images accordingly. Figure 6 shows an exemplary result. The depth estimation runs at 12 fps on average. Since the optical flow gives good sub-pixel accurate correspondences, the resulting depth map features sharp depth discontinuities, which is crucial for artifact-free occlusion.

Limitations The super-resolution part of our work is applicable on static and dynamic scenes. It has to be kept in mind that fast motion of both camera and objects in the scene leads to bad image registration results. Due to our flow quality measure, the algorithm does not fail but rather falls back to the LR video stream. In case of fast motion the fall back might only be hardly noticeable due to motion blur. In the case of slow motion results show that high frequency details can be preserved nicely by our algorithm. While image-based rendering in general is a very versatile technique, a limit is encountered, when it comes to view dependent effects. Effects like specular lighting, refraction and parallax depend on the position of the viewer relative to the scene.With our technique those effects cannot be simulated, since all information is baked into the rendering. Therefore, the quality of the result drops dramatically, if view dependent effects are simulated.

6 Conclusion

We developed a system capable of improving the overall quality of an augmented reality setup. Quality in this case means on the one hand the spatial resolution of the video stream and on the other hand the visual quality of the augmented information. Still images captured at a slow frequency are used to improve the spatial resolution of the low resolution video stream. By registering the high resolution still image with sub-pixel accuracy, we can warp it to fit the current video frame. Furthermore, we designed a reference-free metric for the quality of the optical flow that lets us decide whether the image registration was successful or not. Based on this metric, we blend the high resolution image with the video stream. As a result, we get a method that can upsample videos from 640x480px to 2304x1728ms in under 200ms. The algorithm in the worst case falls back to the quality of the LR video stream, introducing only small artifacts in extreme cases. The evaluation shows that this approach yields good results in a variety of scenarios. Using the optical flow to solve the correspondence problem, we can compute a depth map of the scene, which can further be used to create immersive AR effects.

The computational complexity of physically correct rendering algorithms like pathtracing is high, which does not allow for interactive rendering, especially if the spatial resolution is also high. We described approaches to decouple the frequency of the rendering process from the displaying frequency by distributing the workload over multiple frames. The resulting difference in frame rate is then addressed using image warping. The results show that this strategy can be applied to move complex rendering algo-

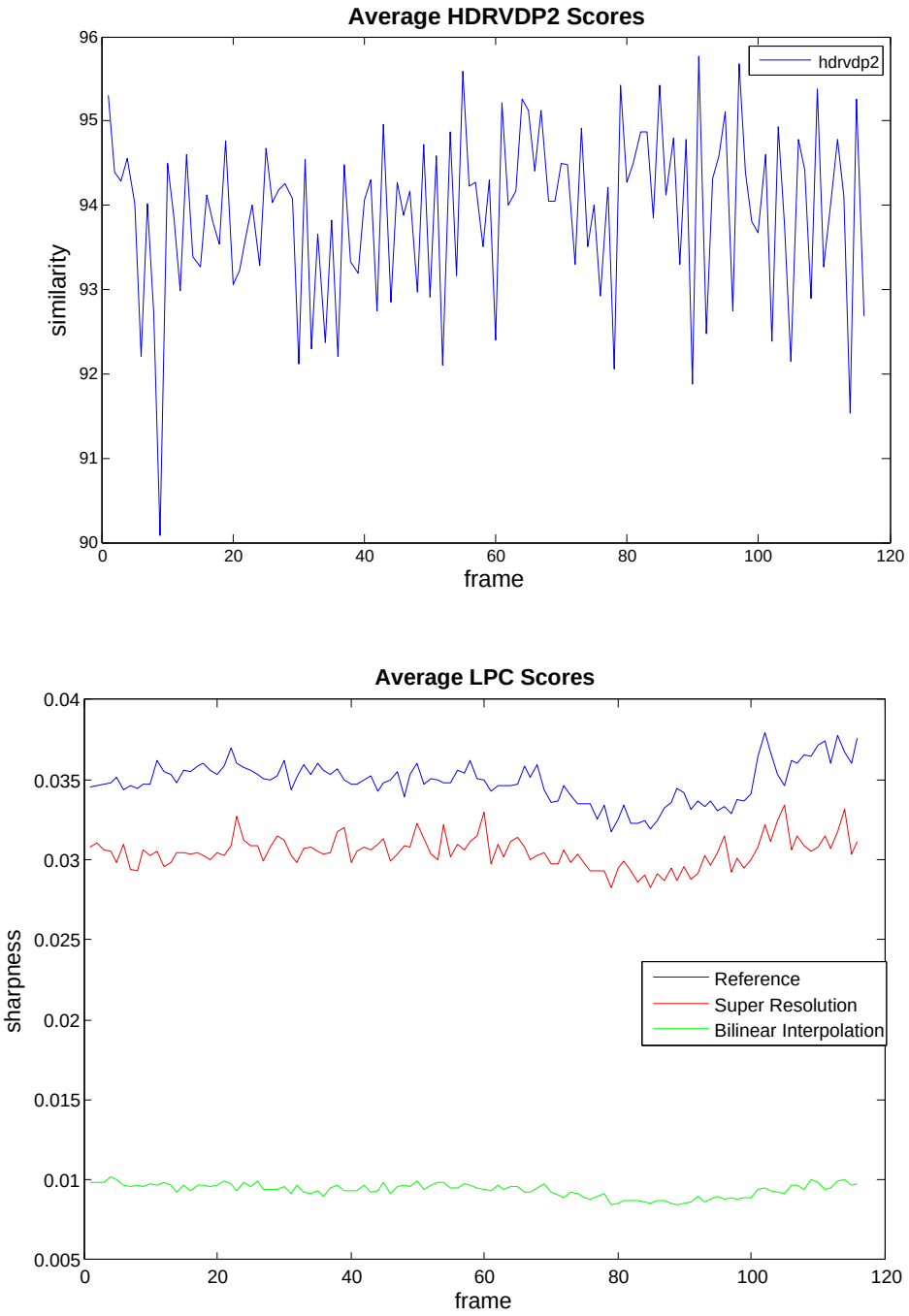

Figure 7: Average results for Hdr-vdp-2 (similarity) and LPC-SI (sharpness). The upper figure shows that our result is close to the ground truth in terms of structural similarity. The lower figure displays that the output of our SR method produces a sharp high resolution version of the input data close to the ground truth and superior to a bilinear interpolated version.

rithms towards interactivity without a noticeable loss of quality.

Future Work The sub-pixel accurate image registration step is achieved by computing the optical flow between the still frame and the current video frame. This has to be done twice each frame, since we need the forward and the backward flow to be able to compute our quality metric. This step is the most time consuming part of the algorithm. Since we know the camera pose from tracking, the flow computation can be simplified to a one dimensional problem by using epipolar geometry. This simplification would not only speed up the computation, but also might give better results in terms of quality, since the search space is significantly smaller. Furthermore, we aim to move the super-resolution algorithm of our work to mobile devices. Especially for this case, the image registration has to be simplified to achieve a satisfying performance.

References

[1] Simon Baker and Takeo Kanade. Hallucinating faces. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 83–88. IEEE, 2000.

[2] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[3] Pravin Bhat, C Lawrence Zitnick, Noah Snavely, Aseem Agarwala, Maneesh Agrawala, Michael Cohen, Brian Curless, and Sing Bing Kang. Using photographs to enhance videos of a static scene. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 327–338. Eurographics Association, 2007.

[4] Piotr Didyk, Elmar Eisemann, Tobias Ritschel, Karol Myszkowski, and Hans-Peter Seidel. Perceptually-motivated real-time temporal upsampling of 3d content for high-refresh-rate displays. In *Computer Graphics Forum*, volume 29, pages 713–722. Wiley Online Library, 2010.

[5] William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *Computer Graphics and Applications, IEEE*, 22(2):56–65, 2002.

[6] William T Freeman, Egon C Pasztor, and Owen T Carmichael. Learning low-level vision. *International journal of computer vision*, 40(1):25–47, 2000.

[7] Rania Hassen, Zhou Wang, and Magdy Salama. No-reference image sharpness assessment based on local phase coherence measurement. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 2434–2437. IEEE, 2010.

[8] Stefan Hauswiesner, Denis Kalkofen, and Dieter Schmalstieg. Multi-frame rate volume rendering. In *Proceedings of the 10th Eurographics conference on Parallel Graphics and Visualization*, pages 19–26. Eurographics Association, 2010.

[9] Robert Herzog, Elmar Eisemann, Karol Myszkowski, and H-P Seidel. Spatio-temporal upsampling on the gpu. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pages 91–98. ACM, 2010.

[10] T. S. Huang and R. Y. Tsay. Multiple frame image restoration and registration. In *Advances in Computer Vision and Image Processing*, volume 1, pages 317–339, Greenwich, 1984.

[11] Rafal Mantiuk, Kil Joong Kim, Allan G. Rempel, and Wolfgang Heidrich. Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions. *ACM Trans. Graph.*, 30(4):40:1–40:14, July 2011.

[12] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Efficient point-based rendering using image reconstruction. In *SPBG*, pages 101–108, 2007.

[13] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 39–46. ACM, 1995.

[14] Leonard McMillan Jr. *An image-based approach to three-dimensional computer graphics*. PhD thesis, Citeseer, 1997.

[15] M. Otte and H.-H. Nagel. Optical flow estimation: Advances and comparisons. In Jan-Olof Eklundh, editor, *Computer Vision ECCV '94*, volume 800 of *Lecture Notes in Computer Science*, pages 49–60. Springer Berlin Heidelberg, 1994.

[16] Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V Sander, and Diego Nehab. An improved shading cache for modern gpus. In *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, pages 95–101. Eurographics Association, 2008.

[17] Bruce Walter, George Drettakis, and Steven Parker. Interactive rendering using the render cache. In *Rendering techniques 99*, pages 19–30. Springer, 1999.

[18] Xuan Yu, Rui Wang, and Jingyi Yu. Real-time depth of field rendering via dynamic light field generation and filtering. In *Computer Graphics Forum*, volume 29, pages 2099–2107. Wiley Online Library, 2010.