PixMix: A Real-Time Approach to High-Quality Diminished Reality

Jan Herling*
Ilmenau University of Technology

Wolfgang Broll†
Ilmenau University of Technology

Figure 1: Even for Diminished Reality on heterogenous backgrounds, our approach produces a coherent video stream in real-time.

Abstract

Diminished Reality (DR) allows to remove objects from a video stream while preseving a frame to frame coherence. Some approaches apply a pseudo-DR, allowing for the removal of objects only, while their background can be observed by a second camera. Most real DR approaches are highly computational expensive, not even allowing for interactive rates and/or apply significant restrictions regarding the uniformity of the background, or allow linear camera movements or even a static camera only.

In this paper we will present a real-time capable Diminished Reality approach for high-quality image manipulation. Our approach achieves a significantly better performance and image quality for almost planar but non-trivial image backgrounds. Our Diminished Reality pipeline provides coherent video streams even for nonlinear camera movements due to the integration of homography based object tracking.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques; I.4.9 [Computing Methodologies]: Image Processing and Computer Vision—Applications

1 Introduction

Image inpainting aka context-aware fill/repair, image completion, or image synthesis allows for filling masked or selected areas of an image by synthesized content, ideally undistinguishable from its environment. Image inpainting has been a research topic for several years (see e.g. [7], [9], [30]), but has only recently become available as a standard feature in popular image manipulation tools. However, the quality of the individual approaches measured by the overall coherence between the filled image part and the surrounding image environment differs significantly. Those approaches providing a good quality even for non-trivial backgrounds are typically rather slow (see e.g. [29] and [6]). Even for images at rather low resolutions such as VGA, most approaches cannot apply proper inpainting in real-time, most of them not even at interactive rates.

Several approaches exist to video inpainting (e.g .[14], [13], [21], [26], [27], [34], [37], [35]) as video inpainting has large potential due to its usage in video post-production (removal of undesired objects) and video frame repair when digitizing vintage movies. Except for our previous work [12] those approaches do not aim for a video frame manipulation at interactive frame rates and typically feature further restrictions such as static cameras, some of them not even ensuring frame to frame coherence. Other approaches for real-time Diminished Reality exist ([38], [11]), but require several synchronized cameras to capture the real background behind objects to be diminished, rather than synthesizing the area of the object to be removed. Further, approaches able to hide tracking markers have been proposed ([28], [16]) superimposing the marker with semi-dynamic textures and applying an alpha blending at the contour of the markers.

In this paper we will introduce a new real-time capable approach for Diminished Reality based on high-quality image inpainting. While inspired by the speed-up of randomized approaches such as Barnes et al. [4] and our first trial [12], we apply a combined pixel based approach rather than a patch-based approach. This allows us to perform high-quality image inpainting in real-time. The real-time capability combined with a frame to frame coherence provides the basis for real-time video manipulations. We additionally apply a homography based approach which enables us to support rotational camera movements in addition to linear camera movements.

This paper is structured as follows: in section 2 we will review the recent related work on image inpainting with a focus on quality and speed. In section 3 we will introduce our approach to image inpainting providing highly sophisticated results in real-time. In section 4 we show how this approach can be used to realize real-time Diminished Reality. We will present our novel real-time object selection and tracking mechanism in detail. We will further introduce our approach to achieve frame-to-frame coherence applying a homography. In section 5 we will discuss limitations and performance issues of our approach before finally concluding and looking

*e-mail:jan.herling@tu-ilmenau.de
†e-mail:wolfgang.broll@tu-ilmenau.de

IEEE International Symposium on Mixed and Augmented Reality 2012
Science and Technology Proceedings
5 - 8 November 2012, Atlanta, Georgia
978-1-4673-4662-7/12/$31.00 ©2012 IEEE

into future work.

2 Related Work

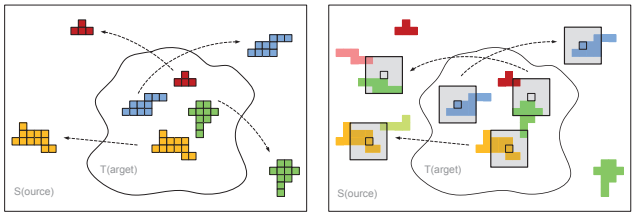
Wexler et al. [35] successfully demonstrated how to remove objects from video sequences. Their approach applied 3D image patches, extending over space and time, using the entire video sequence for patch look-up. While this allows for quite sophisticated results in video manipulation, it cannot be used to manipulate live video streams, where only the current frame and to some extent previous frames are available. Further, while allowing for a user created synthesis mask, their approach, was limited to a static camera and was not able to provide iterative or even real-time capable results. Simakov et al. [29] extended the coherence-based approach of Wexler et al. by a completeness term. Thus, they achieved a bidirectional dissimilarity function also allowing for image reshuffling. However, the approach was too slow for interactive image manipulations. In PatchMatch Barnes et al. [4] resp. in Generalized PatchMatch [5] the cost function introduced by Simakov et al. was used within a randomized patch searching approach. Instead of seeking the optimal image information, randomized iterations are used to find patches close to the optimum. This significantly speeded up the overall process, even allowing for interactive manipulations of images of a reasonable size. By our original approach [12] we already demonstrated a real-time capable approach for object removal from video streams, allowing the manipulation of live videos. We used a randomized approach similar to those of Barnes et al. while applying the similarity measure of Wexler et al. We further introduced means for frame to frame coherence providing the basis for interactive video manipulation. The video synthesis mask was defined by a simple but fast snake contour approach, providing sufficient computational time for the image inpainting. However, in the original approach we achieved the real-time performance mainly by data and sample reductions, e.g. using grayscale images and applying an under sampling inside the target region. The synthesis quality therefore did not reach the quality of the static image approach of e.g. Barnes et al. Pritch et al. [23] proposed an synthesis approach based on a global optimization problem using single pixels and their direct neighborhood rather than patches. They apply a graph labeling approach seeking for an optimal solution over several pyramid layers. However, this approach is several magnitudes too slow for real-time DR. Bugeau et al. [6] define an energy function to be minimized, composed by three terms: self-similarity, diffuse and propagation, and coherence. They propose to compute a correspondence map mainly by the application of pixel intensities inside a patch similar to the approach of Demanet et al. [8]. They achieve sophisticated qualitative results, but do not qualify for real-time use. Takeda et al. [32] apply a homography for removing occluders from video streams. However, their approach is restricted to rotational camera movements and can be used for non real-time post processing only.

3 Image Inpainting

In this section we will present our approach to real-time inpainting.

3.1 Mapping Function

Image inpainting can be defined as a global minimization problem of finding the transformation function $f : T \to S$ producing minimal overall synthesis costs for an arbitrary image I according to a given cost function. The image I is subdivided into the two distinct sets T and S with $I = T \cup S$, $T \cap S = \emptyset$ and $S \neq \emptyset$. All pixels from T(arget) are to be replaced by pixels defined in S(ource). Thus, f defines a mapping between target and source pixels inside an image to be manipulated. Once f has been determined, the final image can be created by replacing all target pixels with source information as defined in the determined mapping. Generally, the result of a manipulated image may be considered as acceptable, if the replaced (synthesized) image content blends in seamlessly with the surrounding image information while it remains free of disturbing artifacts and implausible blurring effects. Further, the new image information should visually fit to the remaining image parts, while image content not existing in the source S must not be used to synthesize the target T (equivalent to the coherence measure of Simakov, et al. [29]). Thus, the transformation function f is based on the following two constraints:



(a) Spatial cost (b) Appearance cost

Figure 2: The two cost constraints of the transformation functions.

- Neighboring pixels defined in T should be mapped to equivalent neighboring pixels in S. This first constraint ensures the structural and spatial preservation of image information (see figure 2(a)).
- The neighborhood appearance of pixels in T should be similar to the neighborhood appearance of their mapped equivalents in S. Thus, a visually coherent result and seamless transitions at the border of the synthesized area are ensured (see figure 2(b)).

The global minimization problem to solve is to find a transformation function f producing the minimal overall cost for an image I and a target region $T \subset I$, with $S = I \setminus T$:

$$\min_f \sum_{p \in T} cost_\alpha(p), \tag{1}$$

while $p = (p_x, p_y)^\top$ is a 2D position and the cost function $cost_\alpha : T \to \mathbb{Q}$ is defined for all elements (pixels) inside the target region.

3.2 Cost Function

As described above, our approach subdivides the overall costs into a part based on the spatial impact and a part based on the appearance impact. This can be represented by the following linear combination:

$$cost_\alpha(p) = \alpha \cdot cost_{spatial}(p) + (1-\alpha) \cdot cost_{appearance}(p) \tag{2}$$

while the control parameter $\alpha \in [0,1]$ allows a balancing between both types of costs. Minimization of the spatial cost impact forces a mapping of neighboring target pixels to neighboring mapping pixels. This is represented for an arbitrary neighborhood N_s by $cost_{spatial} : T \to \mathbb{Q}$:

$$cost_{spatial}(p) = \sum_{\vec{v} \in N_s} d_s[f(p) + \vec{v}, f(p+\vec{v})] \cdot w_s(\vec{v}) \tag{3}$$

with the pixel set/region N_s holding the spatial relative positions of neighboring pixels, any suitable spatial distance function $d_s(\cdot)$ and an individual weight parameter $w_s \in \mathbb{Q}$, while $\sum_{\vec{v} \in N_s} w_s(\vec{v}) \equiv 1$ and $\mathbf{0} \notin N_s$ must hold. Ideally, any neighbor $\vec{v} \in N_s$ of p is mapped to the corresponding neighbor $\vec{v} \in N_s$ of $f(p)$. The spatial cost sums up the spatial distances $d_s(\cdot)$ from this ideal situation for any $\vec{v} \in N_s$ and $p \in T$ (see figure 2(a) and 3). Thus, our approach fundamentally

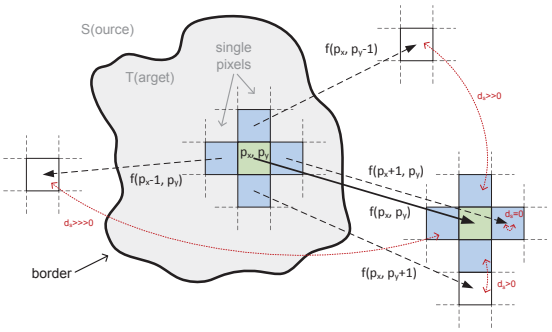


Figure 3: Spatial cost calculated by neighboring mappings depicted for a four-neighborhood. The mapping $f(p_x+1, p_y)$ is ideal and therefore the local spatial cost $d_s(\cdot)$ is zero. The mapping $f(p_x, p_y+1)$ is quite good and therefore $d_s(\cdot)$ is almost zero. However, the mappings for $f(p_x, p_y-1)$ and $f(p_x-1, p_y)$ are far away from the ideal positions resulting in a high $d_s(\cdot)$.

differs from previous pixel- and patch-based approaches such as [8], [7], [30], [23], [4] or [6] applying appearance similarity costs. In contrast to those approaches, our spatial cost function allows for a significant faster convergence while reducing image blurring and geometrical artifacts. This novel cost constraint can be seen as an elastic spring optimization automatically minimizing neighboring mapping offsets. While N_s allows for any kind of neighborhood, a common symmetric neighborhood is defined by:

$$N_s(\delta_s) = \{\vec{v} | \forall \vec{v} \in (\mathbb{Z} \times \mathbb{Z}) : 0 < |\vec{v}| \leq \delta_s\} \quad (4)$$

where $\delta_s \in \mathbb{R}$ specifies the radius of the neighborhood.

The impact of the appearance cost measure is represented by $cost_{appearance} : T \to \mathbb{Q}$:

$$cost_{appearance}(p) = \sum_{\vec{v} \in N_a} d_a[i(p+\vec{v}), i(f(p)+\vec{v})] \cdot w_a(p+\vec{v}) \quad (5)$$

with N_a being an individual neighborhood equivalent to N_s holding the relative positions of neighboring pixels. $i(p)$ holds the image pixel value of image I, and d_a specifies a pixel intensity distance measure. The application of an appearance cost can be found in several related works like the approach of Efros [10], Demanet et al. [8] or Barnes et al. [4]. However, our additional weight function w_a allows for weighting appearance distances individually according to external constraints like e.g. the synthesis border between source and target pixels. Thus, appearance costs close to the synthesis border may have higher impact to the overall cost to avoid undesired border effects such as edges or visual discontinuities. Tests with different neighborhood sets revealed that N_a provides the best results regarding the trade-off between accuracy and performance when represented by a small image patch centered at p. Further, circular neighborhood sets required far more processing time while providing only a negligible visual improvement. A patch size of 5x5 pixels proved to provide sufficient details regarding the visual content while still allowing for fast computation.

In our approach we use the sum of squared differences (SSD) for the appearance distance d_a as it provided a suitable trade-off regarding performance and quality when compared to other measures such as the sum of absolute differences (SAD) and the zero-mean SSD. Also, tests showed that the spatial distance function is well approximated by the squared distance clamped to τ_s:

$$d_s(p_0, p_1) = \min(|p_0 - p_1|^2, \tau_s) \quad (6)$$

The upper border τ_s crops the spatial cost to a maximal influence as the cost for mappings with an already large distance of e.g. 200 and 2000 pixels should be the same. The cropping is comparable to the Tukey robust M-estimator [31] where the error remains constant if it exceeds a specified threshold. We apply a symmetric neighborhood $N_s(\delta_s)$ with $\delta_s = 1$ defining a four-neighborhood or $\delta_s = \sqrt{2}$ defining an eight-neighborhood while the importance weighing for those small neighborhoods is set to a uniform weight $w_s(\vec{v}) = \frac{1}{|N_s|}$. Tests further showed that the l^2 norm provides better results but obviously requires significantly more computation time.

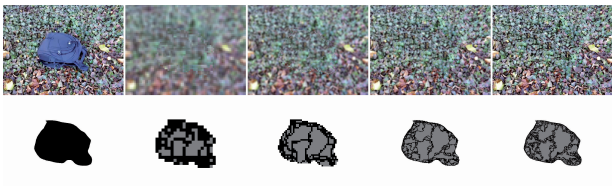


Figure 4: Iterative layer refinement: Original image and mask (left column), intermediate results of each layer (upper row) and corresponding neighboring blocks (bottom row).

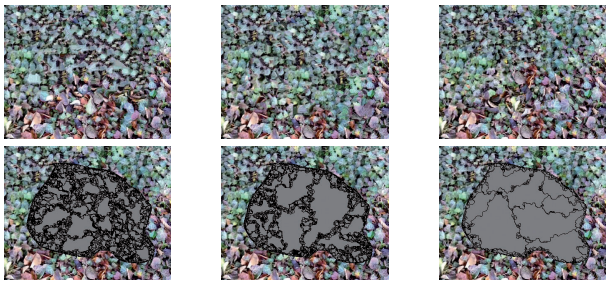
3.3 Iterative Refinement and Propagation

Finding the optimal transformation function f is realized by starting with a rather rough guess of f, followed by a series of iterative refinements steps. At each iteration, the mapping for each target pixel is sought to be improved. Randomly, individual source positions are tested according to the local cost function and accepted whenever the local cost can be reduced. The improved matching is then propagated to neighboring positions in T. This approach is similar to that proposed by Barnes, et al. [4] and the approach we applied in our previous work [12]. However, as these two approaches apply the dissimilarity measure of Wexler, et al. [35] or Simakov, et al. [29] respectively, each refinement needs a target information update of an entire image patch, requiring the application of the individual contribution from each patch followed by a normalization. Instead, our approach updates a single pixel only and avoids expensive normalizations. The iterative refinement is applied on an image pyramid starting with a reduced resolution layer and increasing the image size until the original resolution has been reached. The applied image pyramid allows for covering visual structures with individual frequency, speeds up the mapping convergence and significantly reduces the chance that the algorithm gets trapped by some local minimum. Figure 4 shows some intermediate results during an inpainting process visualizing the spatial cost and the corresponding intermediate inpainting result. Further, a comparison between weak, medium and strong spatial weighting is depicted in figure 5 showing the characteristic joint image blocks after the final iteration.

Barnes et al. [4] applied an information propagation improving the overall process significantly. However, originally the propagation idea in the context of image inpainting had been proposed by Ashikhmin [2] and [8]. Our work applies a comparable propagation to benefit from the significant speedup opportunity. However, instead of propagating the position of entire image patches, our approach forwards single pixel mapping positions only.

4 Real-time Diminished Reality

The significant performance improvement compared to previous approaches allows us to apply the image manipulation techniques to video streams in real-time. However, in addition to the aspects to be considered in image manipulation, manipulations of video streams additionally require a frame to frame coherence. The simplest video stream manipulation would be a static image synthesis in the first frame followed by an interpolation of this artificial image content in



(a) Weak weighting (b) Moderate weighting (c) Strong weighting

Figure 5: Neighboring block comparison for individual spatial weighting. The top row shows the final result, the bottom row provides the spatial costs. The stronger the weighting, the larger the joint blocks (zero spatial cost: gray; non-zero spatial cost: black).

subsequent video frames. However, pure content interpolation cannot handle dynamic effects in a video stream like changing light conditions and moving camera positions sufficiently. Therefore, our approach creates a synthesis result based on the visual information of the recent frame to handle dynamic elements while using an already synthesized key frame as a reference model.

4.1 Object Selection

Interactive object selection for static image inpainting is a rather trivial problem as the synthesis mask may be defined by the user applying common selection techniques such as polygon or lasso tools. The closer and more accurately the object to be removed is selected (including e.g. areas shadowed by the object), the better the final synthesis result. Object selection and image segmentation for static frames has been in focus of research for several decades. The most sophisticated approaches include combinations of graph, mean-shift, scribble and painting based techniques like the work of Ning et al. [20], Rother et al. [24], Levin et al. [18], Arbelez et al. [1] or the progressive paint selection of Liu et al. [19]. However, although their segmentation results of arbitrary objects may achieve a high accuracy; these approaches do not provide real-time performance, often requiring several iterative adjustments by the user and never include image areas shadowed by the objects to be removed. Existing selection and segmentation approaches for video streams are even slower like the approaches of Tong et al. [33] and Bai et al. [3].

Thus, in our previous work [12] we applied a simple active contour algorithm [15] allowing to detect and track objects providing a noticeable boundary with an almost homogenous background. This approach allowed users to roughly select previously unknown objects while tracking them during the successive video frames. Although our previous approach benefited from the usability and the high selection and tracking performance, the boundary and homogenous background constraints limited the number of application areas significantly.

Therefore, we combined the rather rough selection technique of our previous approach with a more complex segmentation approach allowing for non homogenous backgrounds and stronger blurred object contours. As recent state-of-the-art segmentation approaches are not real-time capable, we developed a segmentation and tracking algorithm based on the following three assumption:

- Objects to be removed are entirely visible in the video frame and do not intersect with the frame boundaries in successive frames.
- The image area to be removed is directly enclosed by image content visibly different from the undesired area.
- The enclosing image content itself may be textured and may change along the object boundary.

Figure 6 depicts that these three conditions hold for several individual environments. Also, the figure shows that the image areas covered by the direct neighborhood of the rough user selection are visually different from the content of the objects to be selected. Therefore, we designed an approach detecting image content visually not matching with the characteristics of the rough selection.

In detail, the selection approach is based on several fingerprints $U = \{U_{p_1}, U_{p_2}, \ldots, U_{p_n}\}$ storing the appearance of the frame at n equally distributed positions $p_1, p_2, \ldots, p_n$ spread on the roughly defined object contour. Fingerprints U_{p_i} are determined by a function $\phi(p_i) : I \to \mathbb{Q}^m$ defining the most important visual characteristics of each point. Each fingerprint U_i is composed of m individual components defining disjoint fingerprint characteristics, and thus $U_i = \{u_1, u_2, ..., u_m\}$.

The n fingerprints are compared to the entire image content inside the rough user-defined contour and tested upon similarity. A pixel p inside the rough contour is considered as an undesired pixel if:

$$\sum_{k=1}^{|U|} d_\phi(\phi(p), U_k) \geq |U| \cdot \gamma \tag{7}$$

where γ defines the amount of necessary fingerprint dissimilarities to consider the pixel p as undesired and d_ϕ measures the dissimilarity between two fingerprints U_k and U_j by:

$$d_\phi(U_k, U_j) = \begin{cases} 0, & \tilde{d}_\phi(u_{k_i}, u_{j_i}) \leq v_i, \quad u_{k_i} \in U_k, u_{j_i} \in U_j, \forall i \in [1, m] \\ 1, & \text{else} \end{cases} \tag{8}$$

while $\tilde{d}_\phi$ measures the component-wise distance between the corresponding components of two individual fingerprints. The distances are rated according to reference thresholds v_i.

The cost measure $\tilde{d}_\phi$ is defined as the one dimensional Euclidean distance:

$$\tilde{d}_\phi(u_{k_i}, u_{j_i}) = |u_{k_i} - u_{j_i}| \tag{9}$$

while the corresponding thresholds v_i are adapted to the deviation of U. However, instead of using the entire deviation of all fingerprints U, we separate U into disjoint fingerprint clusters $C = \{C_1, C_2, \ldots C_b\}$ with $C_1 \cup C_2 \cup \ldots \cup C_b = U$ with $b \ll n$ and take the maximal deviation from all b clusters as reference thresholds. Thus, each v_i is calculated by:

$$v_i = \max_{i \in [1,b]} \sqrt{\mathbb{E}[{C_i}^2] - (\mathbb{E}[C_i])^2} \tag{10}$$

Without clustering, an application of the deviation would be useless for background images represented by more than one visual characteristic. Figure 7a) depicts a selection situation with individual background areas illustrating the necessity of data clustering.

Finally, the cluster calculation has to be investigated. A wide variety of clustering algorithms exist providing individual clustering accuracies and run-time complexities as hierarchical, k-means, or distribution-based clustering approaches. However, concerning the real-time necessity we decided to apply a very simple but for our needs adequate clustering algorithm:

1. First, one fingerprint is selected randomly and defined as the center of the first cluster.
2. Afterwards, the remaining fingerprints are assigned to the new cluster if their distance is close enough to the cluster's center.
3. If still fingerprints are left, again one of them is selected randomly, defining the center of a new cluster. The algorithm then continues with 2.

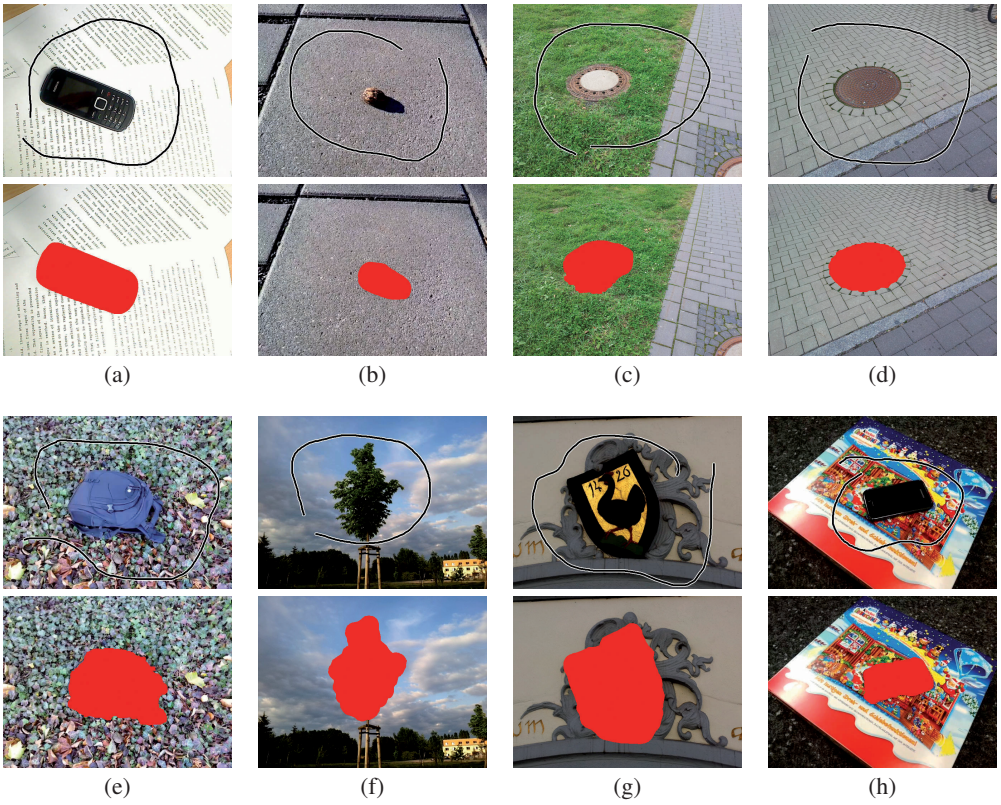


Figure 6: Real-time selection results of our fingerprint segmentation approach for individual objects and backgrounds; top row: original with rough user-defined selection, bottom row: selection result. a) selection with a noisy background, b) object with large shadow with a homogenous background, c) two-colored object and almost homogenous but two-colored background, d) three background regions, e) highly but regular textured background, f) tree with frayed borders, g) non homogenous object, h) highly individual textured background.

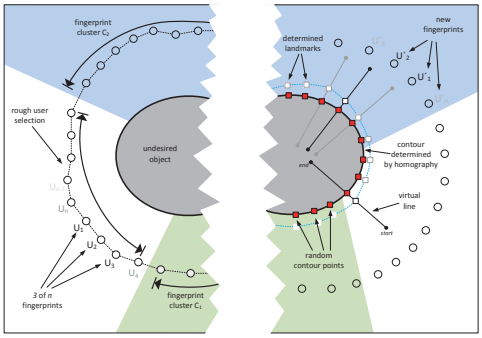


Figure 7: Object selection and tracking scheme. a) Object segmentation by the applications of fingerprints distributed along the rough user selection. The fingerprints are clustered into disjoint sets. b) Adjustment of the contour received by the homography; red squares: randomly selected contour points with perpendicular virtual lines; white squares: resulting landmarks; blue dashed line: resulting accurate and final contour.

This algorithm is repeated several times and the clustering result with lowest maximal deviation within all clusters is finally accepted. Obviously the clustering method is very efficient and we found that the result is accurate enough for our needs. Another benefit is that the number of clusters do not have to be defined in advance as e.g. for k-means algorithms. Thus, the more variations within the fingerprints, the higher the number of final clusters. The rough user-defined selection in figure 6a) produced one fingerprint cluster, while the selection in 6h) ended with 17 clusters.

We defined $\gamma = 0.95$ to detect undesired pixels if at least 95% of the fingerprints reject the corresponding pixel. As data base we use the color image with three channels. Several individual fingerprint functions $\phi(\cdot)$ have been evaluated and we found that a simple Gaussian filter with a large kernel provides sufficient results. However, due to performance reasons we apply a modified mean filter with one large and one small kernel size in combination with an integral image. This optimized mean filter needs eight integral image lookups only and performs much faster than a Gaussian filter while providing sufficient good results.

Further, the entire fingerprint segmentation is combined with a multiresolution approach reducing the computational effort. The approach starts on the coarsest image resolution and forwards the result to the next finer layer. On this finer layer only the border pixels are investigated according to their fingerprint dissimilarity. Finally, a dilation filter is applied to removed possible tiny gaps. All pixels considered as undesired build the binary synthesis mask. A corresponding synthesis contour directly enclosing this mask is determined.

In subsection 5.4 we provide performance results showing the segmentation and tracking time in detail. According to the available processing time, the performance and accuracy of the approach can be tailored easily by adding or removing contour fingerprints or visual characteristics f_i (e.g. color and texture channels). Obviously, the more information is used to determine the dissimilarities, the better the selection result.

We found that the application of fingerprints with three elements, one for each image color channel, is a good compromise between

segmentation accuracy and application speed. However, if accuracy is more important than computational time or if the computational hardware is powerful enough, we use a fourth fingerprint element by default. The fourth fingerprint element represents a simple textureness property determined by averaging the Scharr response for the grascale image information.

4.2 Object Tracking

Once the object's contour is found, the determined object contour has to be tracked in successive video frames. Therefore, we apply a two phase contour tracking approach. In the first phase, a homography based contour tracking is used, while in the second phase the new contour is refined and adjusted regarding to the undesired object area.

Typically, the contour points found will be rather planar. Thus, the relation of a set of tracked contour points between two consecutive frames may be described by a homography [31]. For cases with non-planar backgrounds the motion between two frames can be expected to be very small. Thus, even for these situations, determining the homography may provide a good approximation of the contour movement. For homography determination, only the strongest contour points are tracked between two video frames by using a pyramid based motion detection. A Harris corner vote distinguishes between contour points with good and bad motion tracking properties. Finally, within several RANSAC iterations a reliable homography is determined [31]. However, determination of the homography may fail if only an insufficient number of contour points can be tracked reliably. This may happen if the point motion detection is inaccurate due to an almost homogenous image content.

The second phase for contour tracking is necessary to adjust the new contour positions (determined by the homography) more precisely to the real object contour. If no adjustment would be applied, a solely homography based tracking would accumulate small drift errors over time. Therefore, the fingerprint dissimilarity is used again. This time, the reference fingerprints are automatically defined outside the new contour (equally distributed) in the current frame. However, in order to ensure the performance required, the contour adjustment is performed for a randomly selected subset of the new contour points only. For each random point a virtual line perpendicular to the new contour starting outside and ending inside the new contour is defined. For each pixel, covered by this virtual line, the fingerprint dissimilarity is determined. Therefore, to avoid the influence of fingerprints at the opposite side of the contour, the new fingerprints in the direct neighborhood are investigated only. The first pixel on the virtual line identified as undesired and followed by at least two successive undesired pixels defines a contour landmark for the virtual line. Finally, all contour points extracted in the first tracking phase are adjusted according to the contour landmarks in their neighborhood (see figure 7b). The adjusted positions define the final and accurate contour.

We wish to emphasize the aspect that the homography is calculated in the first tracking phase, while the final synthesis mask is determined after the second phase. Therefore, the homography is based on strong correspondences between contour points, while the mask is based on the actual object border. This separation is important as the homography determined will later be used to ensure a visual coherence of subsequent frames for non-linear camera movements. In contrast to the active snake approach of our previous approach [12] the improved object tracking provides unique correspondences for contour points between successive video frames.

4.3 Diminished Reality Pipeline

First, the user selects those image parts to be replaced. According to this selection, a precise contour is calculated and a static image synthesis is invoked. The synthesized image is provided to the viewer and stored as the key frame I_K. Then the homography and the contour are determined as described before. The key frame, the current binary synthesis mask M_n, and the calculated homography define a reference model I_R for each new video frame I_n: Current frame pixels $p \in I_n$ are copied directly for all desired pixels (with $M_n(p) = 1$), while undesired pixels (with $M_n(p) = 0$) are replaced by interpolated information from the key frame I_K. This interpolation is defined by a concatenation of homographies $(\ldots \cdot H_{n-2} \cdot H_{n-1} \cdot H_n)$ determined since the key frame has been changed.

Further, as the illumination conditions may change between the reference frame and the current frame, our approach allows for compensation of ambient lighting changes. Therefore, the inpainting contour points of the current frame are transformed into the corresponding points in the key frame I_K by application of the gathered homographies. For all pairs of contour points the color differences between the current frame and the reference frame are stored individually for each image channel. Afterwards, a sparse grid is defined covering the entire inpainting mask of the current frame and the nodes of the grid receive approximated color differences by interpolation of the gathered contour differences. This grid then is used to modify the reference frame according to the illumination changes between the current and the key frame. An application of the Poisson equation [22] may provide more accurate results. However, we found that our approach allows for compensating the most important lighting changes while processing in less than 1ms which is several magnitudes faster than e.g. an Poisson related approach.

Afterwards, the synthesis module replaces all undesired mask pixels (with $M_n(p) = 0$) of the current frame with pixels of the remaining image data while creating a visual result almost identical to the given reference model I_R. Therefore, as described for the static inpainting approach, a current mapping f is determined having minimal overall costs for the current frame. The mapping found for the previous frame is adjusted as the initial guess to improve the quality and to speed up the convergence (see subsection 4.4). In contrast to the static image synthesis, the appearance costs are extended and additionally measured between the synthesis result and the reference model (see subsection 4.5). This extension ensures that a sufficient coherence between successive video frames will be received. Then, the synthesized image content is blended with the original frame at the synthesis border to avoid mask border effects like semi-sharp transitions. Tests revealed that a blend border of 1-3 pixels is typically sufficient. Depending on the age of the key frames, it might be replaced by the final synthesis result. However, the number of key frame replacements has to be balanced carefully to avoid visual drifts during the synthesis Afterwards, the synthesis pipeline restarts with the new video frame.

4.4 Mapping Forwarding

In the case of a given mapping function for the previous synthesis, the number of iterations to find a transformation for the new frame can be reduced significantly. A mapping initialization for each mask pixel is determined from the corresponding pixel in the previous image by application of the associated mapping. Therefore, the already determined homography is applied as depicted in figure 8. An arbitrary mask pixel p in frame n is translated to the corresponding pixel p' in frame $n-1$ by:

$$p' = Hp \tag{11}$$

with known Homography H. Further, the previous mapping in frame $n-1$ at point p' is given by:

$$m' = f^{n-1}(p') \tag{12}$$

Thus, a sufficiently precise prediction of this mapping for the current frame n can be calculated by applying the inverse homography:

$$\hat{m} = H^{-1}m' \tag{13}$$

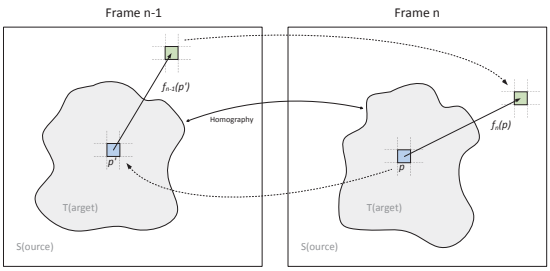


Figure 8: Mapping forwarding with determined homography.

Thus, the initial mapping prediction of an arbitrary mask point in frame n is given by:

$$\hat{f}^n(p) = H^{-1}(f^{n-1}(Hp)) \tag{14}$$

In general an initial guess of $\hat{f}^n$ is defined by:

$$\hat{f}^n = H^{-1} f^{n-1} H \tag{15}$$

Please note that the application of the inverse homography commonly will not result in a precise mapping for the current frame. This is due to the determination of the homography from the object contour, providing exact estimations for totally planar contours only while non-planar contours will introduce a certain error in the calculation of the homography. However, as the change in content between two successive video frames is low, this inaccuracy is negligible as it will be adjusted in the subsequent cost minimization iterations.

4.5 Stream Coherence with Extended Appearance Cost

When compared to the static image synthesis, a coherent video inpainting must additionally follow the appearance of the reference model. Thus, the appearance cost from (5) is extended to measure not only the cost between the synthesized data and the remaining image but also the cost between the synthesized data and the reference model. While the standard appearance cost ensures coherence inside the frame itself, the additional appearance cost ensures visual coherence between successive synthesized frames. Therefore, the appearance cost is extended by the additional cost term $cost'_{appearance} : T \to \mathbb{Q}$:

$$cost'_{appearance}(p) = \sum_{\vec{v} \in N_a} d'_a[i(p+\vec{v}), r(p+\vec{v})] \cdot w'_a(p+\vec{v}) \tag{16}$$

with $r(p)$ holds the image pixel value of the reference model and its own distance function d'_a and weight function w'_a. The distance measure as well as the weight function may be identical to d_a and w_a from (5) respectively. However, if quality is more important than performance, a zero-mean SSD instead of a simple SSD might produce better results because changing light conditions between the current frame and the reference model will be better supported.

5 Discussion

In this section we will discuss the limitations of our approach, review its overall complexity, and look into performance issues.

5.1 Limitations

Although our real-time capable Diminished Reality system is able to provide a high quality result as well as a coherent video stream, it has some limitations which have to be discussed. We found that the fingerprint based object selection approach is a significant improvement compared to our previous approach. However, the object selection may still fail due to the appearance complexity. The more individual characteristics the object and the background have and the smaller the difference between the object and the background characteristics, the more difficult is a reliable selection. Figure 9 shows two examples which our real-time selection approach is not able to handle in a sufficient way. The first example misses significant visual differences between the fingerprints of the rough user-defined contour and the visual characteristics of the stone. Therefore, a wrong area is selected not intended by the user. In contrast, the selection in the second example fails at a small part of the object only. The area of the car's hatchback is not selected as the rough contour fingerprints scattered at the gray sky are almost identical to the visible characteristics of the hatchback due to the reflection of the sky.

(a) (b) (c)

Figure 9: Real-time selection results of the fingerprint segmentation approach. a) Original images with rough user-defined selections, b) the stone cannot uniquely be separated from the environment, c) the algorithm fails to identify parts of the hatchback.

(a) (b) (c)

Figure 10: Video inpainting of a volumetric object. a) Original video frame, b) inpainting result after a few seconds, c) video frame after the camera has moved around the volumetric object.

The presented Diminished Reality system is able to provide a coherent high quality result for a hand-held camera with dynamic translations and rotations. Due to the homography determination of the object's contour the object must be static and should have an almost planar background. However, if the camera is rather static or its movements are limited to very small amounts, a coherent video stream can be provided even for non-planar backgrounds. While our approach supports real-time Diminished Reality applications with a camera moving around a rather static object to be removed, it does not yet cover scenarios, where the object moves around or where both, camera and object are dynamic. Further, our approach currently does not sufficiently cover situations where a previously unknown background becomes visible later on. These situations may occur if an object with a significant volumetric expansion (perpendicular to the surface) in combination with dynamic camera movements has to be removed. A previously synthesized image area is then replaced by the real image content. In order to realize a coherent view, additional fading mechanisms will be required. However, as a real-time Diminished Reality system does not have any information about future video frames (in contrast to removing objects from a pre-recorded video), a trivial solution for this problem does not exist (see figure 10).

5.2 Complexity Considerations

In the following the complexity of our algorithm will be investigated. As an heuristic optimization approach is applied during the

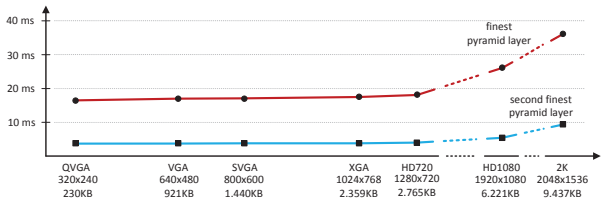


Figure 11: Performance comparison of an inpainting image with individual frame dimensions for the two finest pyramid layers. In each test the inpainting mask has the constant size of 10,000 pixels.

pixel mapping, our approach is much faster than other pixel-based approaches like e.g. that of Pritch et al. [23]. For each pixel to be removed our algorithm tries to find a better matching pixel by several random seeks. This optimization iteration is repeated several times for each pyramid layer. Thus, each pyramid layer with n pixels to be removed has a run-time complexity of:

$$\mathcal{O}(ik \cdot n) \tag{17}$$

while i is the number of improvement iterations and k is the number of random seeks. If i and k are identical on each pyramid layer, the entire complexity over all layers l is given by:

$$\mathcal{O}\left(\sum_{t=1}^{l} ik \cdot n \left(\frac{1}{s}\right)^{t-1}\right) \tag{18}$$

with the pyramid layer shrinking factor s. As the shrinking factor s is larger than 1 (a factor of $s = 4$ means that the horizontal and vertical frame dimensions are bisected at each coarser layer), the geometric series converges to $\frac{s}{s-1}$. Thus, the upper bound of the algorithm complexity is

$$\mathcal{O}\left(ik \cdot \frac{s}{s-1} \cdot n\right) \tag{19}$$

Finally, as the number of iterations i, the random seeking repetitions k and the shrinking factor s are constants, the overall complexity can be estimated by $\mathcal{O}(n)$.

Figure 11 depicts the optimization performance of our algorithm applied to the same image with individual frame resolutions. The image completion mask has a constant size of $10,000$ pixels in each inpainting execution. The performance stays constant up to an image resolution of 1280×720. However, for images with higher frame resolutions the inpainting process shows performance losses. This loss solely issues from the insufficient size of the CPU cache as the overall number of random seeking iterations remains constant, while the computational time for the SSD calculations increases. The test has been performed on an Intel i7 2.13GHz processor with 8MB L3 cache.

5.3 Visual Quality

In this subsection a comparison between well-known benchmark images of recent state-of-the-art approaches, our previous approach [12] and our current image inpainting algorithm is provided.

The figures 12, 13 and 15 depict that our proposed approach is able to recover simple image structures with visual results comparable to related approaches while processing several magnitudes faster.

Figures 13, 14, and 15 show that our previous approach provided undesired blurring effects for heterogeneous images. Although the patch similarity approach of Wexler et al. is known as tending to minor blurring artifacts, the blurring of our previous approach might be slightly larger due to the real-time performance

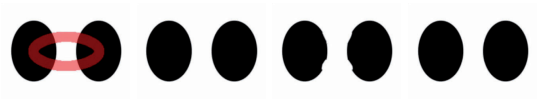


Figure 12: Result comparison for the *Blobs* image. The visual result by Kwok et al. [17] (kindly authorized by the author, ©2010 IEEE) is almost perfect (b), our previous result [12] (c) disturbs the geometric shape while our new approach is comparable to Kwok (d).

constraints. In contrast, the pixel mapping approach introduced in this paper cannot result in blurring artifacts as the final inpainting result is not determined by superimposing image patches.

A direct comparison between our previous and our current approach is provided in figure 16 showing that the new approach is able to preserve simple structures as well.

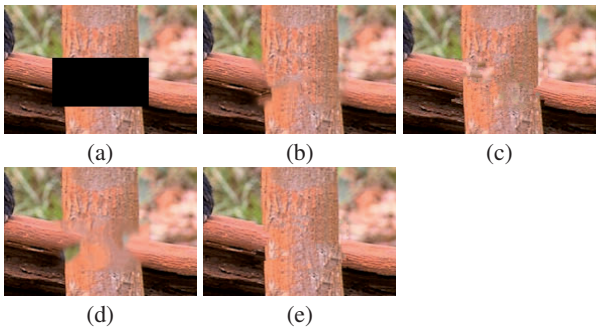


Figure 13: Result comparison for the *Wood* image. a) Original image with inpainting mask; b) result by Drori et al. [9], c) resulty by Shen et al. [25], d) our previous result [12], e) our result. The reference images are taken from [25], kindly authorized by Shen and Drori, ©2010 IEEE.

5.4 Performance Issues

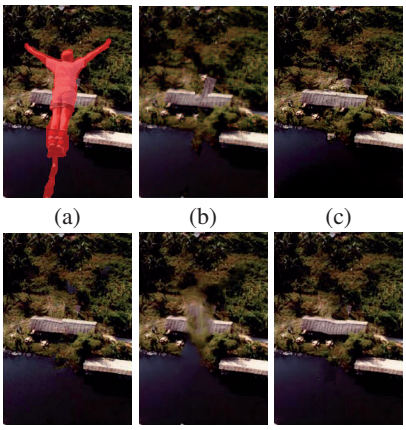
In this subsection we provide a detailed performance discussion of our approach. Unless mentioned otherwise, all performance tests are applied using a laptop Intel i7 Nehalem Core with 2.13GHz running Windows 7. The implementation is realized in C++.

The object selection and tracking performance have been measured for individual objects and backgrounds with a video resolution of 640×480. The values are measured during video sequences of 30 seconds and averaged afterwards. Table 1 shows the detailed results separated into the individual pipeline components. The table depicts that, once an object is reliably selected, the object tracking needs between 4.62ms and 6.31ms for each frame in average. Additionally, a frame filtering for the fingerprint function (with three color components) is necessary taking 2.92ms in average. Thus, even in the worst case the object tracking can be done in less than 10ms for each frame.

Compared to static image inpainting, the manipulation of an entire video is more complex as the creation of a convincing coherent stream cannot be achieved by the sole application of inpainting approaches. As our original approach showed, solely applying inpainting allows for linear and smooth camera movements only and requires an almost homogenous background in order to produce a coherent video stream. Nevertheless, our original approach still showed noticeable artifacts and undesired flow effects whenever the camera moved around the objects to be removed. Therefore, our current approach applies the combination of image inpainting and homography detection to allow for an improved coherence. We

(a) (b) (c) (d) (e)

Figure 14: *Elephant* image from Drori et al. [9] compared with two related approaches: a) Original image with inpainting mask, b) result by Drori et al. [9], c) result by Xu et al. [36], d) our previous result [12], d) our new result. The reference images are been taken from [36], kindly authorized by Drori and Xu, ©2010 IEEE.



(a) (b) (c)

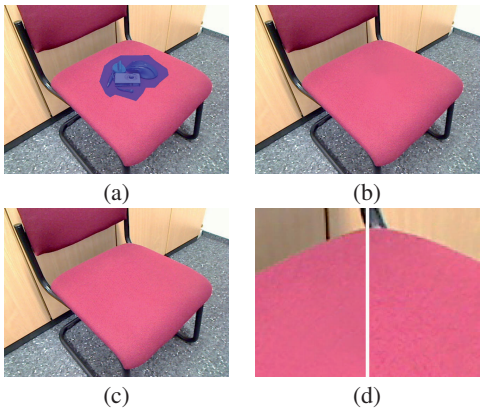
(d) (e) (f)

Figure 15: Result comparison for the *Bungee* image with four individual approaches. a) Original image with inpainting mask, b) result by Criminisi et al. [7], c) result by Shen et al. [25], d) result by Kwok et al. [17], e) our previous result [12], f) our new result. The reference images have been taken from [7], [25] and [17], kindly authorized by the authors, ©2010 IEEE.

measured the performance and the quality of the results of our approach for individual objects and different backgrounds. Further, we applied our approach on the data of our original approach. The comparison confirmed the significant coherence improvements of our current approach compared to our original results (please see the supplemental material for a detailed comparison). Our result for a non-homogenous background is depicted in figure 1. The object is removed reliably during the entire video sequence while almost no disturbing coherence errors are visible. We measured the performance for the video synthesis applying the same video sequences as already used for the tracking performance evaluation. Again the same video resolution of 640×480 pixels is applied. Table 2 provides the detailed performance values showing the dependency between performance and the amount of pixels to be removed in each frame.

Table 1: Performance overview of the fingerprint selection and tracking approach for four individual video sequences. From left to right: Used video sequence; averaged resulting mask size; time for the object selection in the first frame; for the successive frames: averaged processing time of contour point tracking; homography determination and contour adjustment.

Sequence	Mask	Init.	Track.	Homo.	Adjust.
Figure 6b)	12,758px	13.43ms	0.79ms	2.05ms	1.78ms
Figure 6e)	55,422px	22.77ms	0.82ms	1.96ms	3.53ms
Figure 6g)	60,503px	24.25ms	0.41ms	1.62ms	3.68ms
Figure 6h)	26,486px	19.78ms	0.85ms	2.02ms	2.47ms



(a) (b)

(c) (d)

Figure 16: a) Original image with inpainting mask, b) result by our previous approach [12], c) our result not blurring the structure of the chair, d) zoom-in of our previous result (left) and the new result (right).

Table 2: Performance overview of the video inpainting for four individual video sequences. From left to right: Used video sequence; averaged ratio between mask pixels and the entire frame pixels; averaged reference frame preparation time; averaged inpainting time; the resulting pixel fillrate corresponding to the ratio between mask pixels and inpainting time.

Sequence	Mask ratio	Ref.	Inpainting	Fillrate
Figure 6b)	4.2%	4.60ms	11.16ms	1,143.2 px/ms
Figure 6e)	18.0%	6.32ms	48.21ms	1,149.6 px/ms
Figure 6g)	19.7%	6.45ms	53.37ms	1,133.7 px/ms
Figure 6h)	8.6%	5.47ms	21.38ms	1,238.8 px/ms

Thus, the time for the entire Diminished Reality pipeline including tracking and video inpainting lies between 38.83ms for the example from figure 6b) and 59.08ms for the example from figure 6g) respectively. Therefore, our system reaches between 17 and 25 fps providing the visual quality result as provided in the supplemental material. However, if a reduced quality is acceptable, the inpainting performance can be improved by reducing the number of optimization iterations and mapping seeking repetitions as discussed in subsection 5.2.

6 Conclusion

In this paper we presented PixMix, our pixel based approach to image inpainting and real-time Diminished Reality. Further, a real-time capable object selection and tracking algorithm has been introduced. Our inpainting approach allows for easy weighting between the spatial and the appearance term of the cost function in order

to provide optimal inpainting results. The overall results showed fewer artifacts than other approaches, providing high-quality image inpainting in real-time. This allowed us to realize a high-quality Diminished Reality approach without any pre-processing or other knowledge about the environment. We achieved this by extending the overall cost function by a frame-to-frame coherence term and by applying a homography as a first guess for the mapping in the next frame providing a significantly better initialization. Our approach is based on the previous and the current frame only, allowing for a high-quality manipulation of live video streams. In our future work, we plan to extend the homography based approach to arbitrary 3D objects. We further intend to include automatic constraint generation in order to optimize the initialization. Finally, we will investigate smart temporal mapping approaches, providing more sophisticated results for moving objects, while still allowing for real-time video manipulations.

REFERENCES

[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898 –916, may 2011.

[2] M. Ashikhmin. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, I3D '01, pages 217–226, New York, NY, USA, 2001. ACM.

[3] X. Bai, J. Wang, D. Simons, and G. Sapiro. Video snapcut: robust video object cutout using localized classifiers. In *ACM SIGGRAPH*, pages 1–11, New York, 2009. ACM.

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. In *ACM SIGGRAPH*, pages 1–11, New York, 2009. ACM.

[5] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *Proceedings of the 11th ECCV*, pages 29–43. Springer-Verlag, 2010.

[6] A. Bugeau, M. Bertalmío, V. Caselles, and G. Sapiro. A comprehensive framework for image inpainting. *Trans. Img. Proc.*, 19(10):2634–2645, 2010.

[7] A. Criminisi, P. Perez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200 –1212, sept. 2004.

[8] L. Demanet, B. Song, and T. Chan. Image inpainting by correspondence maps: a deterministic approach. *Computer*, 1100(03-40):21750, 2003.

[9] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 303–312, New York, NY, USA, 2003. ACM.

[10] A. A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *The Proceedings of the Seventh IEEE ICCV*, volume 2, pages 1033 –1038, 1999.

[11] A. Enomoto and H. Saito. Diminished reality using multiple handheld cameras. In *ACCV'07 Workshop on Multidimensional and Multi-view Image Processing*, 2007.

[12] J. Herling and W. Broll. Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments. In *9th IEEE ISMAR*, pages 207 –212, oct. 2010.

[13] J. Jia, Y.-W. Tai, T.-P. Wu, and C.-K. Tang. Video repairing under variable illumination using cyclic motions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):832 –839, may 2006.

[14] J. Jia, T.-P. Wu, Y.-W. Tai, and C.-K. Tang. Video repairing: inference of foreground and background under severe occlusion. In *In Proceedings of CVPR' 2004*, volume 1, pages 364—371, June 2004.

[15] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *Int. Journal of Computer Vision*, 1(4):321–331, 1988.

[16] O. Korkalo, M. Aittala, and S. Siltanen. Light-weight marker hiding for augmented reality. In *9th IEEE International Symposium on Mixed and Augmented Reality, ISMAR 2010, Seoul, Korea, 13-16 October 2010*, pages 247–248. IEEE, 2010.

[17] T.-H. Kwok, H. Sheung, and C. C. L. Wang. Fast query for exemplar-based image completion. *Trans. Img. Proc.*, 19:3106–3115, December 2010.

[18] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):228 –242, feb. 2008.

[19] M. Liu, S. Chen, J. Liu, and X. Tang. Video completion via motion guided spatial-temporal global optimization. In *Proceedings of the 17th ACM international conference on Multimedia*, MM '09, pages 537–540, New York, NY, USA, 2009. ACM.

[20] J. Ning, L. Zhang, D. Zhang, and C. Wu. Interactive image segmentation by maximal similarity based region merging. *Pattern Recogn.*, 43(2):445–456, Feb. 2010.

[21] K. A. Patwardhan, G. Sapiro, and M. Bertalmio. Video inpainting under constrained camera motion. *IEEE Transactions On Image Processing*, 16(2):545–553, Feb 2007.

[22] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 313–318, New York, NY, USA, 2003. ACM.

[23] Y. Pritch, E. Kav-Venaki, and S. Peleg. Shift-map image editing. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 151 –158, 29 2009-oct. 2 2009.

[24] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, pages 309–314, New York, NY, USA, 2004. ACM.

[25] J. Shen, X. Jin, C. Zhou, and C. C. L. Wang. Technical section: Gradient based image completion by solving the poisson equation. *Comput. Graph.*, 31:119–126, January 2007.

[26] Y. Shen, F. Lu, X. Cao, and H. Foroosh. Video completion for perspective camera under constrained motion. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 63 –66, 0-0 2006.

[27] T. Shiratori, Y. Matsushita, X. Tang, and S. B. Kang. Video completion by motion field transfer. In *IEEE Conference on CVPR' 2006*, volume 1, pages 411 – 418, june 2006.

[28] S. Siltanen. Texture generation over the marker area. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '06, pages 253–254, Washington, DC, USA, 2006. IEEE Computer Society.

[29] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR*. IEEE Computer Society, 2008.

[30] J. Sun, L. Yuan, J. Jia, and H.-Y. Shum. Image completion with structure propagation. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 861–868, New York, NY, USA, 2005. ACM.

[31] R. Szeliski. *Computer Vision : Algorithms and Applications*. Springer, 2010.

[32] K. Takeda and R. Sakamoto. Diminished reality for landscape video sequences with homographies. In *Proceedings of the 14th international conference on Knowledge-based and intelligent information and engineering systems: Part III*, KES'10, pages 501–508, Berlin, Heidelberg, 2010. Springer-Verlag.

[33] R. Tong, Y. Zhang, and M. Ding. Video brush: A novel interface for efficient video cutout. *Comput. Graph. Forum*, 30(7):2049–2057, 2011.

[34] M. V. Venkatesh, S.-C. S. Cheung, and J. Zhao. Efficient object-based video inpainting. *Pattern Recogn. Lett.*, 30(2):168–179, Jan. 2009.

[35] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(3):463 –476, march 2007.

[36] Z. Xu and J. Sun. Image inpainting by patch propagation using patch sparsity. *Trans. Img. Proc.*, 19:1153–1165, May 2010.

[37] Y. Zhang, J. Xiao, and M. Shah. Motion layer based object removal in videos. In *Application of Computer Vision, 2005. WACV/MOTIONS '05 Volume 1. Seventh IEEE Workshops on*, volume 1, pages 516 –521, jan. 2005.

[38] S. Zokai, J. Esteve, Y. Genc, and N. Navab. Multiview paraperspective projection model for diminished reality. In *Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '03, pages 217–, Washington, DC, USA, 2003. IEEE Computer Society.