Diminished Reality using Appearance and 3D Geometry of Internet Photo Collections

Zhuwen Li[1] Yuxi Wang[2] Jiaming Guo[1] Loong-Fah Cheong[1] Steven ZhiYing Zhou[1,2*]

[1]Dept. of Electrical Computer Engineering, National University of Singapore
[2]National University of Singapore (Suzhou) Research Institute

ABSTRACT

This paper presents a new system level framework for Diminished Reality, leveraging for the first time both the appearance and 3D information provided by large photo collections on the Internet. Recent computer vision techniques have made it possible to automatically reconstruct 3-D structure-from-motion points from large and unordered photo collections. Using these point clouds and a prior provided by GPS, reasonably accurate 6 degree of freedom camera poses can be obtained, thus allowing localization. Once the camera (and hence the user) is correctly localized, photos depicting scenes visible from the user's viewpoint can be used to remove unwanted objects indicated by the user in the video sequences. Existing methods based on texture synthesis bring undesirable artifacts and video inconsistency when the background is heterogeneous; the task is rendered even harder for these methods when the background contains complex structures. On the other hand, methods based on plane warping fail when the background has arbitrary shape. Unlike these methods, our algorithm copes with these problems by making use of internet photos, registering them in 3D space and obtaining the 3D scene structure in an offline process. We carefully design the various components during the online phase so as to meet both speed and quality requirements of the task. Experiments on real data collected demonstrate the superiority of our system.

Index Terms: H.5.1 [Information Systems]: Multimedia Information Systems—Augmented Reality; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Sensor Fusion, Tracking; I.4.9 [Computing Methodologies]: Image Processing and Computer Vision—Application;

1 INTRODUCTION

While augmented reality (AR) enhances users' perception of reality with context-specific information, such as text, sound, video or graphics, diminished reality (DR) aims at removing parts of the real world around us, including visual and vocal information. In particular, visual DR is so important that DR is usually taken to mean removing objects and replacing it with an appropriate background in a scene. The success of DR helps to make special effects possible for broadcast industry, plan and illustrate revamp or modification in structure design, and filter unimportant information in the scene so that the user can focus on crucial things.

The specific term DR is not frequently mentioned until recently, but there has been significant related works solving the problems of image completion and object removal in the literature. Previous approaches can be roughly separated into two categories:

Patch based methods [5, 6, 11, 12, 26, 33] generally sample patches from the user's images or video frames to fill in removed regions. However, these methods cannot be directly adopted for online DR operations, because they usually manipulate all video frames together in a batch manner. Recently, Herling and Broll [15, 16] have adopted similar techniques to provide object cancellation for DR purpose. This kind of methods does not rely on any pre-processing and is computationally efficient. Though they usually produce visually plausible results, the filled-in portions may not be authentic. In other words, the recovered background may not be the same as the real scene behind the occluding objects. Thus, when the background is complex and has rich semantics, these methods are likely to result in conspicuous artifacts and frame-to-frame inconsistency. This limits the applicability of these methods to outdoor scenes which often contain these rich structures.

Multi-view based methods [8, 18, 22, 38] use extra cameras to capture the occluded background, which is then used to restore the removed regions. This kind of methods usually recovers the real scene behind the occluding objects, hence giving more convincing results than that of patch based methods. However, these methods usually require the user to carefully set up multiple cameras or capture multiple images, which makes it user-unfriendly in practical use. Moreover, setting up the cameras in proper configuration and computing structure of the scene are very time consuming, which makes it impossible to run all these tasks in real time.

Recently, we have witnessed a phenomenal upsurge in the number of photographs uploaded on the internet, many of which are pictures of well-known touristic landmarks. These latter photographs are often captured from almost every conceivable viewing position and angle, different times of the day and night, spanning changes in season, weather, and decades. Furthermore, big cities are now being captured at street level (*e.g.* Google Streetview and Windows Virtual Earth) and related photos are often uploaded by local residents or tourists with the GPS information. The availability of such rich imagery of highly travelled places, and increasingly even those off the beaten path, presents opportunities in Mediated Reality. Most importantly, over the past few years, advances in computer vision [23, 32, 36] have made it possible to automatically obtain 3-D structure-from-motion (SfM) points from these large and unordered photo collections.

In this paper, we propose a new system level framework suitable for outdoor DR, leveraging for the first time both the appearance and 3D information provided by large photo collections on the Internet. We first obtain a rough estimate of the user's position using GPS and gyroscopes. A reasonable amount of the 3D scene structure data can then be pre-fetched through wireless wide-area networks for accurate user localization. For our purpose, we assume SfM point clouds of the environment are available, which is a reasonable assumption because these point clouds can be readily prepared by an automatic offline 3D reconstruction system using large photo collections on the Internet. Therefore, any 3D map service is able to integrate this feature into its system. We also assume photos used for 3D reconstruction are well registered in 3D space, so that we can use the information in them to restore removed regions. After accurate user localization, we search if there are sufficiently close enough images, and we denote them as reference photos. These photos often exist, because people tend to photograph from

*e-mail: elezzy@nus.edu.sg

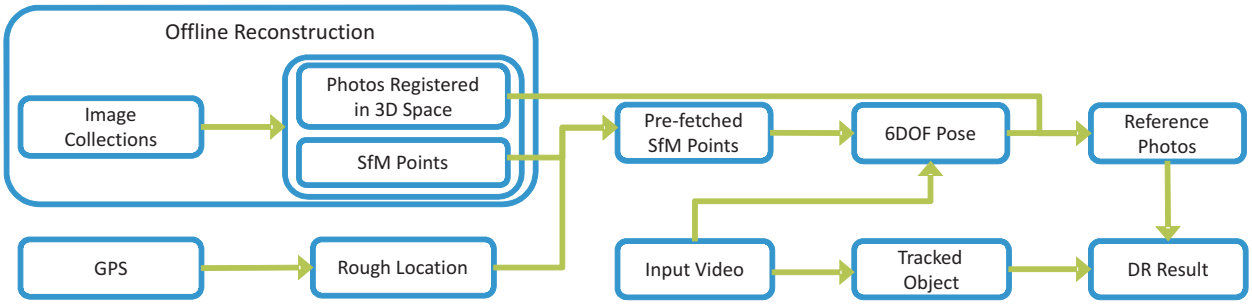


Figure 1: System workflow overview. See more details in the main text.

some popular vantage points and some parts of a scene are photographed much more often than others [31]. If sufficient reference images exist, several versions of the current view can be directly synthesized from these images by homographic transformations, since these reference photos differ from the current view and from one another by a rotation approximately.

In order to remove unwanted objects from the real scene, these objects are selected by users in the first frame of a video sequence, and then tracked by an object tracking method in all consecutive image frames. Given the synthesized patches from the reference images corresponding to the "hole" of the current view, we compute the correlation among them to detect the real unobstructed background, based on the simple premise that the most highly correlated patches correspond to the true unobstructed background. Among these unobstructed patches, we select the one with the highest correlation, and blend it with the current view. We carefully design the various components during this online phase so as to meet both speed and quality requirements of the DR task.

The overall architecture of our system is presented in Figure 1, which outlines the above-mentioned workflow. Generally, our approach belongs to the category of multi-view methods. However, compared to previous multi-view methods, our approach has the following advantages. Firstly, we replace the user-unfriendly tasks of camera setup and photo registration with an automatic offline 3D reconstruction using large photo collections on the Internet. Secondly, based on the rich Internet photos, our approach is able to select reference images and detect unobstructed regions automatically; the patches synthesized by one transformation warping from the reference images are more realistic than those by pixel-wise reconstruction methods [8, 18, 38]. Last but not least, we perform user localization, object tracking and diminished reality simultaneously, providing a basis for next-generation Mediated Reality environments, capable of augmenting and diminishing the real world environment at once and with ease.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 introduces the offline data preparation and online user localization. Section 4 describes reference photo selection strategy. In order to apply the algorithm to videos, consistent video completion is introduced in Section 5. Then, our experimental results are illustrated in Section 6 and the limitations of this work are discussed in Section 7. Finally, we draw the conclusion in Section 8.

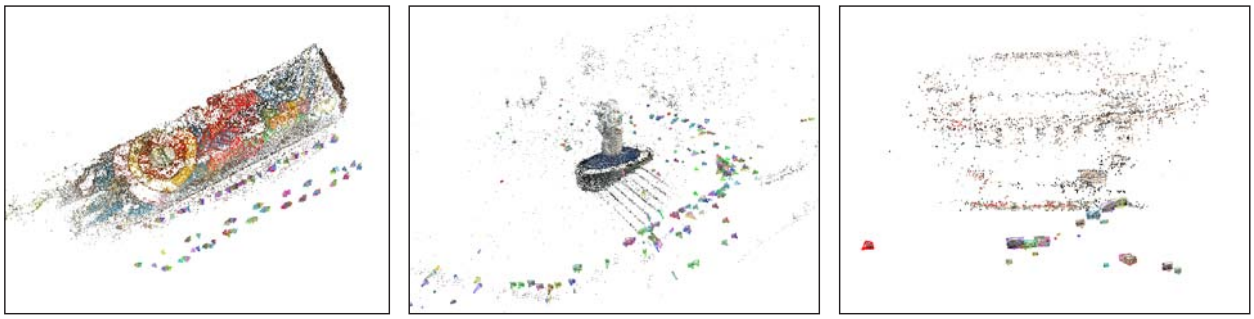
2 Related Work

There is a significant literature on object removal. Among patch based image completion methods, Wexler *et al.* [33] follow the spirit of the classic texture synthesis method [7] and extend it to 3D volume space to fill in missing portions of videos. Their well-defined objective function induces global coherence, thus leading to spatio-temporally consistent video sequences and images. Criminisi *et al.* [5] propose an exemplar-based method combining the advantages of texture synthesis and image inpainting. Drori *et al.* [6] propose an iterative method to approximate the unknown regions and then add details according to the most frequent and similar examples. Mansfield *et al.* [26] consider image patches remaining natural under a variety of transformations, thus having much more candidates for patch searching. Recently, Granados *et al.* have proposed two algorithms: one to deal with dynamic backgrounds [12] and the other to remove dynamic objects with a free moving cameras [11]. Though these methods perform well in offline video editing, they are not designed for online DR due to the batch processing of video frames. Among this line of works, Herling and Broll [15, 16] propose real-time object removal algorithms based on image completion and synthesis for DR purpose.

Among multi-view methods, Zokai *et al.* [38] use paraperspective projection model to generate the correct background from multiple calibrated camera views. Enomoto and Saito [8] use planar warping and marker based calibration to synthesize occluded scenes from multiple handheld cameras. Jarusirisawad *et al.* [18] present a plane-sweep algorithm for removing occluding objects of the scene from multiple weakly-calibrated cameras. In particular, our approach is most closely related to that of Lepetit *et al.* [22], which also uses full 3D reconstruction to restore the removed regions. Our work differs from theirs in several key ways. Firstly, while they need the user to select some key-frames over the whole video and outline the boundary of the unwanted object on these key-frames accurately, our approach can automatically track the unwanted object given the rough location of the object in the first frame. Secondly, they re-project the SfM points to the image plane and then perform 2D Delaunay triangulation with these re-projected points, following which homographic transformation is done in each triangle. Thus, their results will depend highly on how good and dense the reconstructed SfM points are in representing the shape of the occluded objects, while our approach only use SfM points to rapidly locate the pose of the current view, then realistic patches are synthesized from the rich Internet images. Thus, the SfM point clouds can be compressed for data reduction in our approach.

Two interesting works [14, 34] also use rich internet photo collections as sources for image inpainting, both based on an image retrieval step. While [14] aims to find semantically similar photos to complete the scene, [34] tries to retrieve photos of the same scene. However, it is not easy to extend these two methods to DR due to their computational cost. Furthermore, [14] may find different scene photos for different frames, thus resulting in video inconsistency. And since [34] does not carefully consider the camera positions of the inpainting image and the retrieved photos, the composite image may include new occlusions or mis-registered planes.

Since our method needs accurate camera pose estimation, it is also related to localization methods based on SfM points. Recent progresses include [1, 2, 17, 24]. Irschara *et al.* [17] generate a minimal set of synthetic images that covers the point cloud and use a vocabulary tree to retrieve similar images in this covering, while



(a) (b) (c)

Figure 2: 3D reconstructions and camera poses of the "NUSWall" scene with 175 photos, the "Merlion" scene (partial) with 647 photos and the "Fullerton Hotel" scene with 36 photos.

Li *et al.* [24] select a minimal set of 3D points that cover the images, and use these points themselves as the matching primitives. In their first work, Arth *et al.* [2] bring similar localization method to the AR community and use the idea of potentially visible sets to partition and organize the SfM points. Their subsequent work [1] is extended for outdoor AR applications using the GPS estimate as prior and an online panoramic view generation. Their implementation is eminently suitable for executing on current generation mobile devices.

3 3D RECONSTRUCTION AND USER LOCALIZATION

Reconstructing 3D geometry from large collections of Internet images of landmarks and cities is a large field of research. Powerful algorithms [23, 32, 36] have shown that it is possible to fulfil this task automatically. Using such reconstructed scene points, rather than a collection of raw images or a complete model of the scene, as a representation for user localization is promising.

3.1 3D Reconstruction

Our reconstruction system is based on Wu's *Visual SfM* [36]. For our particular purposes of user localization and diminished reality, we make some changes to the algorithm and store more information in each SfM point. First, noisy points are filtered for better matching as in [24]. We sort the SfM points in an order such that points visible from more image views are placed at the front of the list and we only keep a small set of points at the front. This is desirable because points visible from more images are more reliable and noisy points will be filtered out. Second, as stated in [24], the model is compressed for faster searching speed. We choose the smallest subset of SfM points, such that each image is covered by at least K points in the subset (K is 100 as in [24]). This step ensures that views depicting the same scene can at least find some correspondences between one another and builds a compact representation of all images, which in turn accelerates the searching speed in the later matching step. Third, other than the 3D positions of the SfM points, we also store the feature descriptor in each point for the ensuing 2D-3D correspondence matching. In this work, we use the 128-byte SIFT descriptor [25] for each point, because of its invariance to uniform scaling, orientation, and partial invariance to affine distortion and illumination changes, though the choice is not limited to this. For each SfM point, we record its image set from which the point is visible and the 2D position in each image in this set as well. Later we will show how these records can help to carry out immediate image warping. Figure 2 shows some reconstructed scenes used in our experiments.

3.2 User Localization

To localize the user's pose characterized by six degrees of freedom, we need to extract features from the current frame, match them to the SfM points and finally estimate camera pose using Perspective-n-Point (PnP) method given the 2D-3D correspondences.

As the rough location of the user is known from the GPS information, only SfM points in the vicinity are fetched from the map services. This strategy allows us to cope with the limited bandwidth of current generation wireless networks. It also reduces the search field for a correspondence to a much smaller set.

To meet the speed requirement, we opt for a GPU implementation of SIFT feature detection and extraction [35]. For fast querying, we select an approximate nearest neighbor search algorithm - Fast Library for Approximate Nearest Neighbors (FLANN) [28]. This algorithm uses hierarchical k-means trees or multiple randomized k-d trees to provide large speedup in querying with only minor loss in accuracy. Most importantly, this library can automatically choose the best algorithm and optimum parameters depending on the dataset.

To make sure that the correspondences obtained have high reliability, we use the ratio test to decide if a match is good or not. For a certain feature point p, we obtain the first and second nearest points in the search field and denote them as $p\prime_1$ and $p\prime_2$ respectively, and retain the correspondence only if the distance to the first nearest point $dist(p, p\prime_1)$ is much smaller than that to the second nearest point $dist(p, p\prime_2)$, *i.e.*

$$\frac{dist(p, p\prime_1)}{dist(p, p\prime_2)} < \lambda, \tag{1}$$

where λ is the threshold ratio (set to 0.5 in our implementation).

Given 2D-3D correspondences, if the intrinsic parameters of the camera are known, we can rely on the efficient PnP methods [10] with a fast RANSAC [30] to determine the absolute pose. Otherwise, a 4-point perspective pose approach [4] simultaneously estimating the pose and the focal length is applied. Due to the requirement for speed, we assume that the intrinsic parameters of the camera are well calibrated.

4 PATCHES SYNTHESIS AND SEAMLESS INSERTION

4.1 Patches synthesis

Once the user is accurately positioned, we can use nearby photos registered in 3D space to restore the information behind the occluding object. We define the distance d_{cv} from the camera to its visible point set as the median value of distances to all points in the visible set. Assuming we have two cameras, with distances to their visible sets as d^i_{cv} and d^j_{cv} respectively, if the baseline d^{ij}_b between these two

Figure 3: Correlation comparison used to detect unobstructed patches. The ROI is outlined in the green rectangle in the current view, the bottom row of figures are 7 patches warped from reference images, ranked according to average NCC scores.

camera is relatively much smaller than the distance to their visible sets, *i.e.*

$$\frac{d_b^{ij}}{max(d_{cv}^i, d_{cv}^j)} < \delta, \tag{2}$$

where δ is a threshold to set, we consider these two cameras to be close enough to each other. In other words, their poses differ by only a rotation (and possible changes in intrinsic parameters) so that pixels in one image can be directly transferred to the other by a homographic transformation. In our experiments, we are able to find these reference images very often, because people apt to photograph from some popular viewpoints. This fact allows "popular" views to be restored quickly, because under such favorable circumstances, we do not need to synthesize current views using multiple nearby images.

Given a set of reference images, we want to warp them to the current view. A naïve but efficient way is to compute the homography using the rotation obtained via the estimated poses of the cameras and the calibrated intrinsic parameters. However, this method needs exact camera pose estimation, or else the result is often unstable. Traditional Direct Linear Transformation (DLT) method [13] gives better results but it needs 2D-2D correspondences, and feature extraction is very time-consuming, especially when the reference images often have high resolutions. Careful readers may remember that, for each SfM point, we have saved the image set from which it is visible, as well as its 2D positions in each of these images. Thus via the previous 2D-3D correspondence step, we already have at our disposal all 2D-2D correspondences. Finally, we use DLT method to compute a high quality homography.

Unfortunately, in some paths less well trodden, close images may not exist. In this case, we can synthesize the current view images from the next nearest photos. Given a camera pose and two well-calibrated views of a scene with matched points in these two views, we can determine the position of these points projected onto the image associated with the provided camera pose. This is the point transfer problem which is solvable in general given the above information. Rays back-projected from corresponding points in the first and second view intersect and thus determine the 3D point. The position of the corresponding point in the third view is computed by projecting this 3D point onto the image. [13] gives a trifocal tensor based point transfer algorithm which avoids the degeneracy of epipolar transfer. To synthesize the entire image of the current view, we need pixelwise correspondences, which lead to an optical flow estimation problem. However, the latter often gives rise to false correspondences, thus resulting in bad synthesis. Moreover, these procedures are very time-consuming which makes this solution impractical in real-time. Fortunately, in this age and time when pictures are uploaded onto the internet at a torrential rate, even if one ventures far away from the beaten track, there usually exist some relevant photos on the internet. For instance, the Fullerton Hotel in Singapore is not your conventional tourist spot, but there are still several photos on the internet yielding sufficient coverage for some viewpoints (See Figure 2). Yet another method is to mine (in an offline manner) pictures from the travel guides or articles, which will usually give us much more comprehensive coverage of places of interest around the world.

4.2 Background Detection

Within the warped reference images, the region of interest (ROI) patches corresponding to the ROI in the current view might also contain an obstructed scene and cannot be used as the background for other images. Usually, the obstructed view is due to different types of occluding objects. To filter out these images, we compute the correlation between each ROI patch pair. In particular, we use the NCC(Normalized Cross Correlation) measure. After running the comparisons for every pair of images, usually the unobstructed scenes would yield high correlation index, say, 0.8 and above. When two obstructed scenes are compared, they almost always give significantly lower correlation values since it is unlikely that the same object obstructs the same region over different times. Figure 3 shows the 7 different image patches warped to the current view, among which 3 patches are more or less unobstructed and have been detected using the aforementioned correlation comparison. Finally, we choose the image patch with the highest average NCC score as the candidate.

4.3 Seamless Blending

Since outdoor scenes typically exhibit changes in illumination and different camera sensors and settings may capture colors and brightness differently, we need to blend the images to compensate for these differences (and other misalignments). A popular approach to image blending is to perform the operation in the gradient domain [29]. It is typically carried out by solving a Poisson equation with Dirichlet boundary conditions. However, solving Poisson equation is often very time-consuming. Here, we use the Mean-Value Coordinate (MVC) [9] to perform seamless blending, as it is able to produce results close to those of Poisson blending but much faster and more stable in video image blending.

To better understand the method, we first introduce the MVC. Given a closed 2D polygonal boundary curve (in our case, it is the bounding box of the tracking result) $\partial P = (\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_m), \mathbf{p}_i \in \mathbb{R}^2$, the MVC of a point $\mathbf{x} \in \mathbb{R}^2$ *w.r.t* ∂P is

$$\lambda_i(\mathbf{x}) = \frac{w_i}{\sum_{j=0}^{m} w_j}, i = 0, \ldots, m, \tag{3}$$

where $w_i = \frac{\tan(\alpha_{i-1}/2)+\tan(\alpha_i/2)}{\|\mathbf{p}_i - \mathbf{x}\|}$, and α_i is the angle between $\overrightarrow{\mathbf{xp}_i}$ and $\overrightarrow{\mathbf{xp}_{i+1}}$ (Figure 4). These coordinates can be stored compactly in a matrix $\Lambda \in \mathbb{R}^{m \times n}$ with its element $\Lambda_{ij} = \lambda_i(\mathbf{x}_j)$, where n is the number of pixels inside the polygon. Note that for our specific configuration, the polygon is the rectangular bounding box and the shape of this bounding box is fixed once given in the first frame

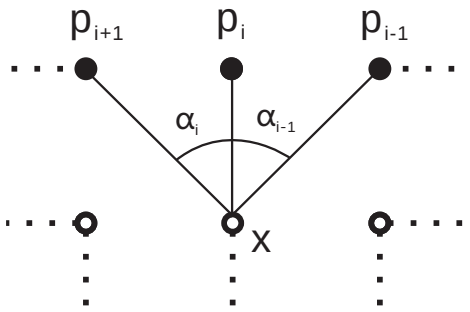


Figure 4: Angles of the MVC.

Figure 5: Left: Blending result with MVC. Right: Result with direct copy & paste.

by the user. Thus, we only need to compute the MVC matrix in the beginning. Moreover, since the MVC only depends on the relative positions of four points and these relative positions often stay unchanged, the computation time can be further saved.

Solving Poisson equation involves constructing a smooth membrane characterized by a harmonic function that interpolates the difference between the target and source patches (here the current view and the reference photos respectively) along the boundary of the entire region [9]. Analogously, a similar smooth interpolation membrane can be constructed using MVC. In particular, we assume a source image patch $\mathbf{I}_s$ with the RGB intensity of its boundary pixels stored in a matrix $\mathbf{Q}_s \in \mathbb{R}^{m\times 3}$ and inner pixels in $\mathbf{P}_s \in \mathbb{R}^{n\times 3}$. Similarly, we assume a target patch $\mathbf{I}_t$ with $\mathbf{Q}_t \in \mathbb{R}^{m\times 3}$ and $\mathbf{P}_t \in \mathbb{R}^{n\times 3}$. The interpolation is performed using the mean value:

$$\mathbf{P}_t^* = \Lambda^T(\mathbf{Q}_t - \mathbf{Q}_s) + \mathbf{P}_s, \tag{4}$$

where Λ is the previously computed MVC. Instead of solving a large linear equation, MVC blending approximates the membrane by two matrix addition and one multiplication operations, which are very fast and easy to parallelize. Figure 5 shows the composite results with and without MVC blending.

5 Video-consistent removal

5.1 Object selection and tracking

For object removal in a video sequence, the unwanted object needs to be selected by the user in the first frame. Since we assume a dynamic scene, where the hand-held camera or the objects in the scene can move freely, the selected object has to be tracked in subsequent video frames to ensure a continuous and coherent removal.

Object tracking has been well studied recently. Some works try to provide exact boundary of the object, such as those based on the active contour algorithm [21]. Other trackers use a bounding box as small as possible to contain the entire object being tracked. In our system, we apply the latter kind of methods for the following three reasons: 1) object selection by a bounding box is much easier for the user than by contour delineation; 2) unlike [15], we have enough registered images to fill in the unwanted region, and thus only need

Figure 6: Tracking results of the "NUS Wall" sequence. From left to right: results of frame 1, 34, 213 and 416.

Figure 7: Tracking results of the "Merlion" sequence. From left to right: results of frame 1, 218, 289 and 335.

to ensure that the selected region covers the entire object without having to worry about removing too large a region; 3) existing object trackers using a bounding box generally perform much more robustly than those based on the active contour algorithm when the object appearance (pose) is changing.

Specifically, the tracking algorithm we applied in our system is the real time compressive tracking algorithm [37], which in our experience has the best balance of performance and efficiency among the various competing algorithms that we have tried (such as Tracking-Learning-Detection [19] and ℓ_1 tracker [27] *etc.*). This approach trains a naïve Bayes classifier to classify unlabeled patches in an online learning manner; the learning process is guided by positive and negative image patches defined according to their proximity to the current tracking location. The kernel part of this algorithm is realized through its feature selection scheme, which projects the image appearance feature space to a low-dimensional compressed subspace, thereby facilitating efficient classification. In summary, the compressive tracker operates as follows:

1. Initial position of the target is given by the user, and the classifier is initialized accordingly in the first frame.
2. For each frame, a set of image patches near to the previous tracking location is sampled, and then the low-dimensional feature is extracted by projection.
3. Use the Bayes classifier to find the tracking location with the maximal classifier response.
4. Positive examples (patches very near to the current location) and negative examples (patches less near to the current location) are updated to train the classifier.

The advantages of the compressive tracker include the following: 1) the learning process can handle the changing of the object appearance, 2) it is robust even when the object is occluded, and 3) it is easy to implement and runs very fast. Interested user can refer to [37] for more details. Figure 6 and Figure 7 show some results of this algorithm. Notice that in Figure 6, the second and the third images have motion blurs, and in Figure 7 the boy turned his body around, and yet both the objects can be correctly tracked due to the online learning procedure.

5.2 Jitter removal

If the video frames are processed independently, jitter often occurs due to the subtle differences of the patches. To maintain temporal consistency, a popular way is to perform gradient domain fusion in the temporal dimension [11, 33]. However, its high computational

Table 1: Summary of processing time

Sequences	Forward	Sideway	Rotation	Merlion 1	Merlion 2	Merlion 3	Fullerton
Image size	536×360						
Patch size	45×45	40×40	40×40	45×86	60×125	32×60	62×72
Feature Extraction	46.0ms	46.3ms	52.4ms	52.4ms	46.1ms	40.7ms	51.5ms
Feature Matching	16.7ms	16.4ms	16.2ms	12.0ms	14.8ms	15.0 ms	15.0ms
RANSAC EPnP	53.3ms	53ms	53.2ms	36ms	46.5ms	29.9ms	29.5ms
Object Tracking	31.1ms	30.8ms	30.9ms	30.5ms	25.5ms	29.9ms	29.5ms
MVC Blending	3.4ms	3.0ms	3.0ms	6.4ms	14.8ms	3.4ms	7.5ms
Total time	150.5ms	149.5ms	155.7ms	137.3ms	147.7ms	118.9ms	132.9ms

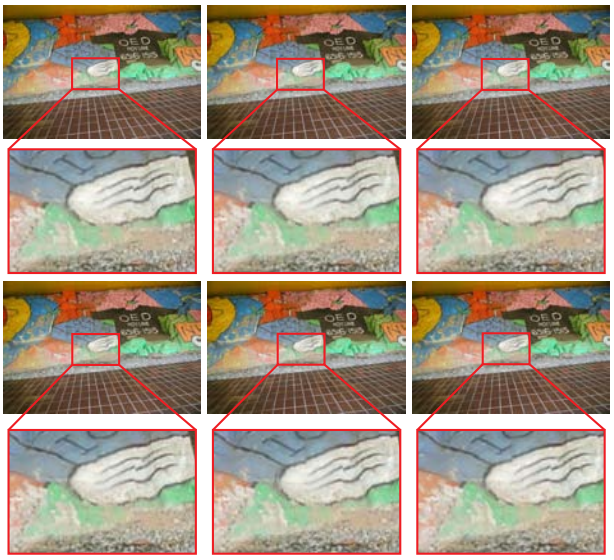


Figure 8: Composite results of three consecutive frames with (top) and without (bottom) the jitter removal filter. The middle frame of the bottom images has slightly fuzzier edge compared to the other frames.

cost hampers the application of this work to DR. Here, we propose a filter to reduce temporal inconsistency.

Assuming that we already have the reference image $\mathbf{I}_r$, what we want to solve is the homography $\mathbf{H}_{r,c}$ that warps $\mathbf{I}_r$ to the current frame $\mathbf{I}_c$. Like the Kalman filter [20], we first perform a prediction step. To predict the homography $\mathbf{H}_{r,c}$, we first compute the homography $\mathbf{H}_{r,c-1}$ between $\mathbf{I}_r$. Note that consecutive frames often fetch the same reference images, thus $\mathbf{H}_{r,c-1}$ would already be available from the previous step most of the time. Even if the consecutive frames obtain different references images, feature matching can be performed rapidly as in Section 4.1. Therefore, the computational cost of this step is very low. We then compute the homography $\mathbf{H}_{c-1,c}$ between $\mathbf{I}_{c-1}$ and $\mathbf{I}_c$ (the goodness of this homography assumption depends on how close $\mathbf{I}_{c-1}$ and $\mathbf{I}_c$ are). Finally, the predicted homography can be written as

$$\mathbf{H}'_{r,c} = \mathbf{H}_{c-1,c}\mathbf{H}_{r,c-1}, \tag{5}$$

Similarly, we can predict another $\mathbf{H}''_{r,c}$ using the frame $\mathbf{I}_{c-2}$ and so forth. After that, we compute the current $\mathbf{H}_{r,c}$ ("measurement" in the terminology of Kalman filter) and finally fuse all the results by averaging them. In our experiments, we find that prediction from two past frames ($\mathbf{I}_{c-1}$ and $\mathbf{I}_{c-2}$) produces adequately consistent results. Figure 8 shows the importance of the jitter removal filter. The first two rows are the composite results obtained with the filter and the bottoms are the results without the filter. Notice that the middle frame of the bottom row has slightly fuzzier edge compared to the other frames. Though the difference is not obvious when viewed from a static image, it shows up conspicuously as shimmering artifacts when viewed in the video mode (See it in the supplementary video).

6 Results

In this section, we present some experimental results using real data. We test four scenes in our experiments. One is the "NUS Wall" scene, in which the photos taken densely cover the scene. The others are the "Merlion", the "Fullerton Hotel" and the "Mount Rushmore" scene, whose photos are downloaded from the Internet. The 3D reconstructions of the former three scenes are presented in Figure 2.

Our system is implemented with C++ and runs on a laptop with Intel(R) Core(TM) i7 CPU Q740, 1.73GHz and a NVIDIA GeForce GT 435M graphics card. We parallelised the feature matching task, because the querying for each feature point is independent. Table 1 tabulates the average processing time of one image frame involved in each step for individual test sequences. Note that our system can run at interactive rates, while Hays and Efros's [14], which is also based on large Internet photo collections, takes about 5 minutes to process a single image by a parallelizing implementation running on 15 CPUs.

Using the "NUS Wall" scene, we first evaluate how our method performs when the video undergoes different kinds of motions. We tested three camera motions: forward translation, sideway translation and pure rotation. Figure 9 shows the qualitative results under the respective camera motions. Since the scene is densely covered by still photos, our algorithm generates high quality results for all the sequences. Even with translational motions and the relatively near scene points, the homography assumption (between $\mathbf{I}_{c-1}$ and $\mathbf{I}_c$) works well. More results can be seen from the supplementary video.

We then apply our algorithm to the unconstrained "Merlion" scene. Figure 10 shows the results of 3 video sequences. The first two sequences mainly contain camera rotations, with the viewpoints well-covered by the online photos, so our method produces high quality diminished reality results. In contrast, the last video consists of translations traversing over larger area, with some views not well covered by the online photos; thus the composite result show artifacts when the camera passes by a position not well covered. See video results in the supplementary video.

To evaluate the performance of our system on a less well-covered scene, we apply our algorithm to the "Fullerton Hotel" scene (Figure 2(c)). This scene contains only 36 photos, with the camera positions mainly restricted to some popular vantage points. We show some qualitative result in Figure 11. While the result is generally good and consistent, it does suffer from slight mis-alignment in some frames due to the sparse and uneven coverage of the scene.

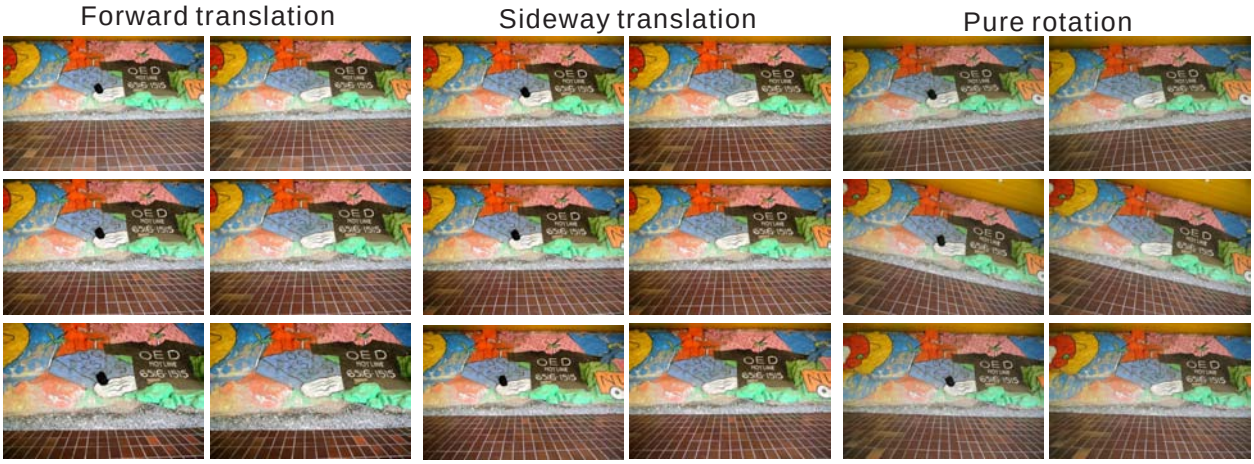


Figure 9: Diminished reality results under different kinds of motions. Under each motion, the left is the original frame, and the right is the corresponding diminished reality result. The first, middle and last frames of the videos are shown here.

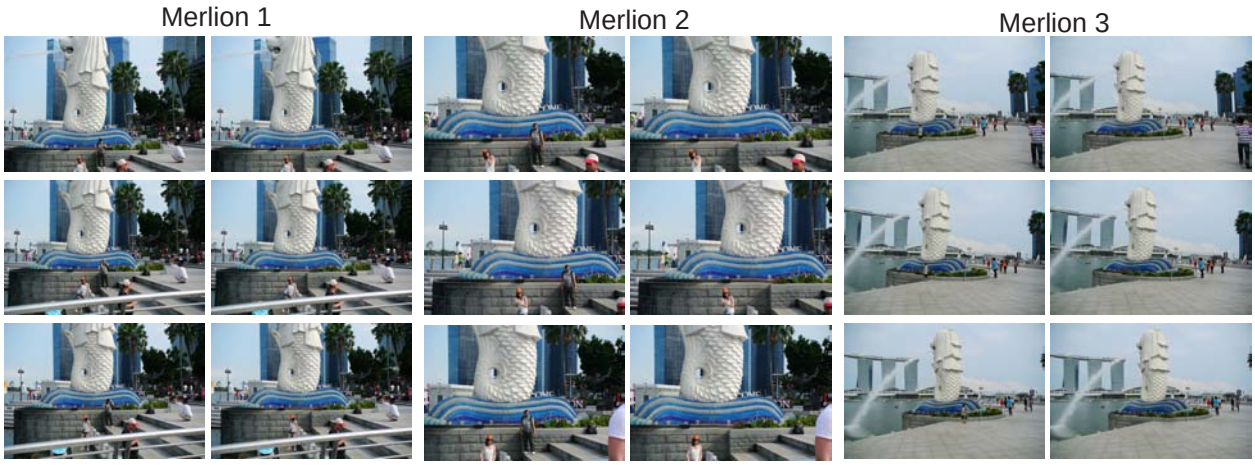


Figure 10: Diminished reality results of "Merlion" scene. Under each video, the left is the original frame, and the right is the corresponding diminished reality result. The first, middle and last frames of the videos are shown here.

This slight loss of quality notwithstanding, the results do demonstrate the potential of internet photos even for less well-known locations.

To further illustrate the superiority of the proposed diminished reality framework, we compare our results on the "Merlion" and "Mount Rushmore" scenes with those of Criminisi *et al.* [5], the commercial PatchMatch [3] in Adobe Photoshop CS5, and Hays and Efros [14]. The first two are patch based methods, whereas the last, like ours, is based on internet photos. In our method, the 3D structure of the "Mount Rushmore" scene is reconstructed using 318 photos. For fair comparison, the image set used for reconstruction in our method is the same as that used for querying in [14]. For the current view in which object removal is desired, we use one of the photos from the photo collections; this conveniently provides us with the camera intrinsic parameters. Note that this does not necessarily mean that the current view has near neighbors; indeed, in the "Mount Rushmore" scene, the nearest neighbor is several metres away. The unwanted objects in the current view are outlined in a bounding box for our and Hays' algorithm, while tighter boundaries are provided as an image mask for the other two algorithms. The image completion results are shown in Figure 12. Note that [14] produces 5 composite images; we only show the visually most correct result. The first row in Figure 12 shows an image taken from the "Merlion" scene and the man in red is what we want to remove in the image. This task is difficult, because this man is standing in front of the railing, which poses problem to most patched-based methods. Here, both [5] and PatchMatch are not structure-aware and thus fail to recover the correct patches. Moreover, the small white house in the distance is almost totally obliterated, while our method restores the exact background using information from other close photos. Though [14] is able to find a nearby camera image, the pasted patch is not well aligned. One little artifact of our method is the reflection of the man on the wet ground, but that is because this region is not outlined by the user. The second row highlights another strength of our algorithm. A large stone beam is selected and removed in the "Mount Rushmore" scene. It is very difficult for [5] and Patchmatch to find even visually plausible recoveries, not to mention authentic ones. [14] again produces a mis-aligned result, whereas our approach fills in correctly what is behind the beam. This experiment also shows that our algorithm tolerates displacements in the positions of the nearby cameras to some extent, provided that the visible point sets are relatively far away compared to the baseline of these cameras.

Figure 11: Diminished reality results of “Fullerton Hotel” scene. The first, middle and last frames of the videos before and after the couple removal are shown here.

Figure 12: Comparison of our image completion results with those of Criminisi, PatchMatch and Hays. From left to right: original images and tracked bounding box used in Hays and our approach, image masks used in the other two methods, results of PatchMatch, results of Criminisi, results of Hays, and our results. We recommend that these images be viewed on-screen and zoomed in.

7 Limitations

It should be pointed out that this system framework is mainly applicable to tourist snapshots of frequently visited landmarks, which have many pictures taken and available on the Internet. For some view points less well captured, the visual result may degrade. For non-public places, we are afraid it is out of the scope of this paper. Nevertheless, in this case, the user may capture some photos in advance and our method can still be applied.

Another limitation is that our approach mainly focuses on removing objects whose occluding background is visible in at least one Internet picture. One example of conflict would be a statue which is very near to a wall. In this case, the wall cannot been ”seen” by any other views, so no satisfying patch can be synthesized for the wall. However, it does not mean that we cannot handle static occluding object, *e.g.*, if the background is far enough from the occluding object, it can still be captured from some other views and thus available for patch synthesis (See the result in Figure 12 below).

8 Conclusion

We have presented a system level framework for DR using prepared structure information and registered photos in the reconstructed 3D space. These data can be collected, captured and provided by 3D map services. The essence of this proposed work lies in how to organize and utilize the copious amount of photo collections on the Internet. Based on a GPS prior, we can localize the user quickly. From this location information and the pre-processed online photo collections, we then choose nearby images to restore the unseen background.

In this paper, we demonstrate the potential of large collections of data on the Internet. We envisage more applications in the future based on a similar framework, such as super-resolution, see-through effect, re-rendering assuming different lighting condition, weather, season, etc. Moreover, we need not restrict ourselves to just photo resources, but also take advantages of audio, texts and videos.

Acknowledgements

The authors wish to thank all the reviewers for their constructive comments to improve the manuscript. This work was partially supported by the PSF grant 1321202075 and the NUSRI grant of Mixed Reality and Interactive Multimedia Research Center R-2012-N-002.

References

[1] C. Arth, M. Klopschitz, G. Reitmayr, and D. Schmalstieg. Real-time self-localization from panoramic images on mobile devices. In *ISMAR*, 2011.

[2] C. Arth, D. Wagner, M. Klopschitz, A. Irschara, and D. Schmalstieg. Wide area localization on mobile phones. In *ISMAR*, 2009.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: a randomized correspondence algorithm for structural image editing. In *SIGGRAPH*, 2009.

[4] M. Bujnak, Z. Kukelova, and T. Pajdla. A general solution to the p4p problem for camera with unknown focal length. In *CVPR*, 2008.

[5] A. Criminisi, P. Perez, and K. Toyama. Object removal by exemplar-based inpainting. In *CVPR*, 2010.

[6] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. *ACM Trans. Graph.*, 22(3):303–312, 2003.

[7] A. Efros and T. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[8] A. Enomoto and H. Saito. Diminished reality using multiple handheld cameras. In *Proc. ACCV'07 Workshop on Multi-dimensional and Multi-view Image Processing*, 2003.

[9] Z. Farbman, G. Hoffer, Y. Lipman, D. Cohen-Or, and D. Lischinski. Coordinates for instant image cloning. In *SIGGRAPH*, 2009.

[10] F.Moreno-Noguer, V.Lepetit, and P.Fua. Accurate non-iterative o(n) solution to the pnp problem. In *ICCV*, 2007.

[11] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, and C. Theobalt. Background inpainting for videos with dynamic objects and a free-moving camera. In *ECCV*, 2012.

[12] M. Granados, J. Tompkin, K. I. Kim, O. Grau, J. Kautz, and C. Theobalt. How not to be seen - object removal from videos of crowded scenes. *Computer Graphics Forum*, 31(2):219–228, 2012.

[13] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[14] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), 2007.

[15] J. Herling and W. Broll. Advanced self-contained object removal for realizing real-time diminished reality in unconstrained environments. In *ISMAR*, 2010.

[16] J. Herling and W. Broll. Pixmix: A real-time approach to high-quality diminished reality. In *ISMAR*, 2012.

[17] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009.

[18] S. Jarusirisawad, T. Hosokawa, and H. Saito. Diminished reality using plane-sweep algorithm with weakly-calibrated cameras. *Progress in Informatics*, (7):11–20, 2010.

[19] Z. Kalal, J. Matas, and K. Mikolajczyk. P-n learning: Boostrapping binary classifiers by structural constraints. In *CVPR*, 2010.

[20] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[21] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. In *ICCV*, 1987.

[22] V. Lepetit and M. Berger. A semi-interactive and intuitive tool for outlining objects in video sequences with application to augmented and diminished reality. In *ISMAR*, 2001.

[23] X. Li, C. Wu, C. Zach, S. Lazebnik, and J. Frahm. Modeling and recognition of landmark image collections using iconic scene graphs. In *ECCV*, 2008.

[24] Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010.

[25] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. of Computer Vision*, 60(2):91–110, 2004.

[26] A. Mansfield, M. Prasad, C. Rother, T. Sharp, P. Kohli, and L. V. Gool. Transforming image completion. In *BMVC*, 2011.

[27] X. Mei and H. Ling. Robust visual tracking and vehicle classi cation via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(11):2259–2272, 2011.

[28] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP)*, pages 331–340, 2009.

[29] P. Pérez, G. M. Gangnet, and A. Blake. Poisson image editing. In *SIGGRAPH*, 2003.

[30] J.-M. F. R. Raguram and M. Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *ECCV*, 2008.

[31] I. Simon, N. Snavely, and S. Seitz. Scene summarization for online image collections. In *CVPR*, 2007.

[32] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph. (SIGGRAPH Proceedings)*, 25(3):835–846, 2006.

[33] Y. Wexler, E. Shechtman, , and M. Irani. Space-time completion of video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):463–476, 2007.

[34] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. In *BMVC*, 2009.

[35] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). http://cs.unc.edu/ ccwu/siftgpu, 2007.

[36] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *ICCV*, 2011.

[37] K. Zhang, L. Zhang, and M.-H. Yang. Real-time compressive tracking. In *ECCV*, 2012.

[38] S. Zokai, J. Esteve, Y. Genc, and N. Navab. Multiview paraperspective projection model for diminished reality. In *ISMAR*, 2003.