Improving Spatiotemporal Inpainting with Layer Appearance Models

Thommen Korah and Christopher Rasmussen

Dept. Computer and Information Sciences
University of Delaware
Newark, DE 19716
{korah, cer}@cis.udel.edu

Abstract. The problem of removing blemishes in mosaics of building facades caused by foreground objects such as trees may be framed in terms of inpainting. Affected regions are first automatically segmented and then inpainted away using a combination of cues from unoccluded, temporally adjacent views of the same building patch, as well as surrounding unoccluded patches in the same frame. Discriminating the building layer from those containing foreground features is most directly accomplished through parallax due to camera motion over the sequence. However, the intricacy of tree silhouettes often complicates accurate motion-based segmentation, especially along their narrower branches. In this work we describe methods for automatically training appearance-based classifiers from a coarse motion-based segmentation to recognize foreground patches in static imagery and thereby improve the quality of the final mosaic. A local technique for photometric adjustment of inpainted patches which compensates for exposure variations between frames is also discussed.

1 Introduction

For a task such as vision-based architectural modeling, one subgoal is to obtain "clean" texture maps of the planar building faces with foreground objects removed. Image/video inpainting or completion [1, 2, 3, 4], a method for image restoration or object removal, suggests a way to remove larger foreground elements by interpolating building features spatially and/or temporally. Typically, the region to be filled is user-specified, but in previous work [5] we showed how problem areas could be automatically identified using motion cues. In that work, we observed that pure spatial inpainting is strictly necessary only where the background is never seen for the entire sequence, and that median filtering suffices when it is present in a majority of views. Our major contributions were (1) how to find the *holes* to be filled and (2) how to use building regions visible in a non-zero *minority* of views to constrain what should be painted there. By combining spatial information from pixels in a partially-completed mosaic with the temporal cues provided by images in the *timeline*, or sequence of images captured, sequences that present significant difficulties for existing background-subtraction techniques could be well-handled.

G. Bebis et al. (Eds.): ISCV 2006, LNCS 4292, pp. 1634–1646, 2006.

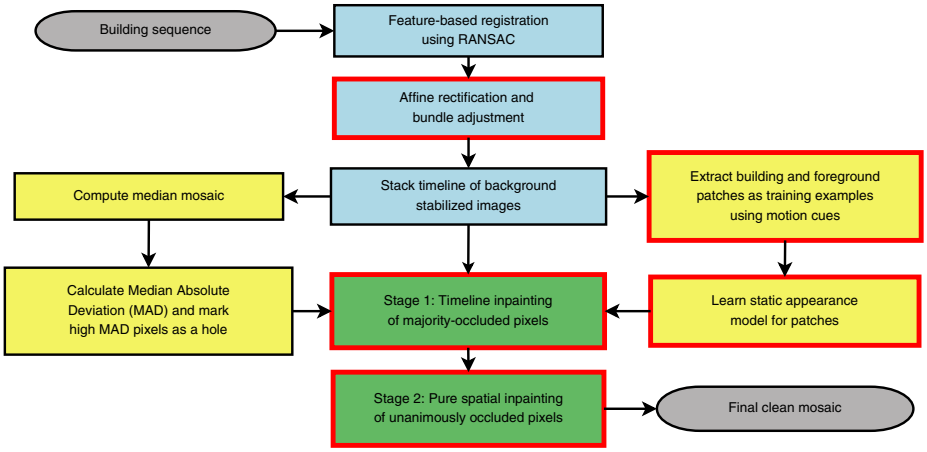


Fig. 1. System diagram for recovering clean texture maps. The thick red boxes indicate where the novel techniques discussed in this work fit into the overall framework.

This paper extends our previous work in several significant ways to achieve considerably improved results on difficult sets of images. First, we have found that motion cues and inpainting alone have shortcomings in eliminating certain foreground elements such as leaves and thin branches. An inability to recognize patches in the timeline containing such features tended to introduce subtle artifacts into the final mosaic, primarily because our method for measuring motion energy was not fine-grained enough to capture high-frequency occluders.

In [6] we enhanced the algorithm with a PCA-based measure of *appearance* similarity for patch selection in the spatiotemporal inpainting algorithm. By learning a local model of the building appearance from a set of exemplar patches, inpainted regions were virtually indistinguishable from the previous SSD-based method and filled in more than an order-of-magnitude faster–a key improvement when dealing with long sequences and panoramas. While the focus of that work was efficiency, appearance cues (vs. solely motion-based ones) suggested additional avenues for increasing the quality of foreground-background discrimination that we explore further in this paper. In the next sections we will briefly review our basic algorithm from [5], which we call *timeline inpainting*, describe our procedures for patch classification and photometric alignment, and present results for several different building sequences.

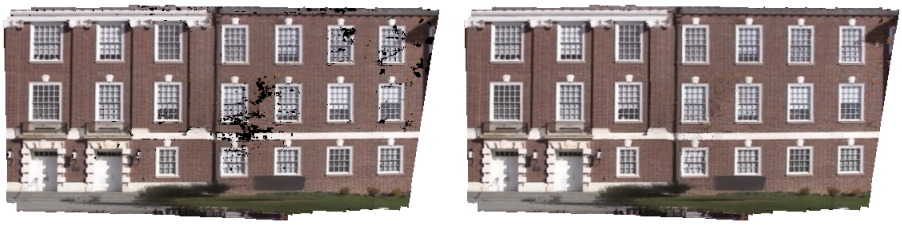
2 Timeline Inpainting

2.1 System Framework

Figure 1 shows a system diagram of the timeline inpainting framework introduced in previous work [5]. Given a building sequence, it outputs a texture map of the facade with all foreground elements removed - as illustrated by Fig. 2

Every alternate frame from 9 to 17 in the 18 frame sequence

(a) (b)



(c) (d)

Fig. 2. Result from various stages of our algorithm for the Building A sequence (top row). The central windows are occluded in almost all frames making the case for inpainting. (a) Median mosaic. Foreground pixels are intermingled with the background when they are in the majority of frames. (b) MAD outliers that will be fed to the inpainting algorithm for filling. (c) Result after timeline inpainting (Stage 1). (d) Result after spatial inpainting (Stage 2).

containing the result from various stages of this algorithm. The boxes in the system diagram are shaded according to its function, and the thick red boxes indicate where the novel techniques introduced in this work fit in. The pipeline begins by computing the dominant planar motion (assumed to belong to the building facade) between successive pairs of images $\mathbf{I}_t, \mathbf{I}_{t+1}$ in a sequence of N frames. This sequence is expected to be captured from a robot platform while moving around the building perimeter (1-D scanning motion). A line detector is applied on the warped images to compute the rectifying homography. Finally, a full bundle adjustment is carried out to compute a single frame-to-mosaic homography for each frame. The result is a stack of background stabilized images $\mathbf{W}_t$.

Each location $\mathbf{p} = (x, y)$ in the mosaic reference frame has a set of pixels from the background stabilized images $\{\mathbf{W}_t(\mathbf{p})\}$ associated with it which we call its *timeline* $\mathcal{T}(\mathbf{p})$. Intuitively, since all pixels on the building facade exhibit the dominant motion, they should appear stationary in the mosaic whereas foreground objects such as trees and signs move due to parallax. Given that each $\mathcal{T}(\mathbf{p})$ contains an unknown mixture of background and foreground object

pixels, our goal is to correctly pick or estimate each background pixel $\mathbf{M}(\mathbf{p})$ where $|\mathcal{T}(\mathbf{p})| > 0$, forming a timeline mosaic $\mathbf{M}_{time}$. It is also important to identify those pixels where the background was never imaged (unanimously occluded), in which case regular image completion needs to be done based on contextual information from the rest of the mosaic. The result of the second stage would be the required texture map of the building mosaic $\mathbf{M}$. These two stages are shown in the green boxes forming the central component of our algorithm.

The boxes to the left of the main pipeline indicate that the median mosaic $\mathbf{M}_{med}$ is used as an initial robust estimator. The temporal median belongs to the background only if it was imaged in a majority ($> 50\%$) of frames. We use the median absolute deviation (MAD) [7] metric to threshold out outlier pixels. These are marked as holes to be inpainted over.

The key contribution of this paper is shown in the yellow boxes to the right of the central pipeline. Most exemplar-based methods [8, 9] use the SSD as the distance function $d(\cdot, \cdot)$ between two image patches. For large search regions (as typically occurs with panoramas or videos), this could be very computationally expensive. We therefore try to "learn" a more compact representation of building and foreground patches to measure building likelihood. Since appearance models can vary with location or season, motion cues are exploited to bootstrap the learning process by generating positive and negative examples of building patches. A pixel-wise Gaussian color model is also learnt to complement the coarser patch-based models. The techniques will be elucidated in greater detail in the following sections.

We first discuss some other improvements we've made to the overall framework including patch selection by better fusion of motion and appearance likelihoods, radiometric alignment of patches from distant frames, and incorporating domain knowledge to assist Stage 2 of spatial inpainting. We will then describe the appearance models used.

2.2 Patch Selection

Consider a patch $\Psi_{\hat{\mathbf{p}}}$ on the fill-front boundary of the timeline mosaic $\mathbf{M}_{time}$ that is the next to be inpainted. Pixels in its unfilled part $\Psi_{\hat{\mathbf{p}}} \cap \Omega$ (where Ω is the hole) will come from the corresponding part of one timeline patch $\Psi^*_{\hat{\mathbf{p}}} \cap \Omega$. The building likelihood of each timeline patch in $\Psi^t_{\hat{\mathbf{p}}}$ is jointly measured by the appearance and motion energy as

$$B(\Psi^t_{\hat{\mathbf{p}}}) = p_{app}(\Psi^t_{\hat{\mathbf{p}}})\bar{p}_{motion}(\Psi^t_{\hat{\mathbf{p}}}). \tag{1}$$

Pixels are then copied from $\Psi^*_{\hat{\mathbf{p}}}$ determined by $* = \text{argmax}_t\, B(\Psi^t_{\hat{\mathbf{p}}})$. The exact definition of the terms on the right side will be deferred until the next section, but suffice it to say that they represent the likelihood of a patch belonging to either the building or the foreground based on appearance and motion cues respectively. To prevent copying patches from timelines where the background was never imaged, we set appearance and motion thresholds T_{app} and T_{motion}.

Without correction With correction

Fig. 3. Results of photometric correction during inpainting on Building C sequence

T_{app} varies with the appearance model used and is determined by cross-validation experiments on the training set. T_{motion} is set to 0.8 implying that patches with more than 80% of pixels above the motion threshold τ_{motion} will not to be included in the computation of (1). The learnt models will be shown to be discriminative with well separated decision boundaries, thus easing the task of setting these thresholds.

Since (1) operates on patches, it does not guarantee against blemishes in the mosaic that occur when tiny fragments of foreground pixels are copied over. Thus a per-pixel decision is also made before copying patches from the timeline - once again based on appearance and motion. The Gaussian color model is used to threshold out bad pixels based on the RGB values, while pixels with motion energy below τ_{motion} are also not copied to $\mathbf{M}_{time}$. This combined framework allows us to find the right balance between spatial and temporal cues at either fine-grained or coarse-grained resolutions.

2.3 Photometric Alignment

For a sequence with significant variations in intensity, either due to lighting changes in the scene or automatic camera controls, the seams between overlapping patches may be pronounced. Graph-cut and gradient-domain algorithms [10, 11] have been used to minimize these effects in mosaicing and texture synthesis. Since most of the photometric variations in our sequence arise due to varying camera parameters, we experimented with exactly recovering the camera response curves to compensate for these radiometric misalignments [12]. However this proved very sensitive to misalignments and foreground objects. Noting that an affine transformation across each RGB channel is able to fully account for contrast and brightness changes [13, 14], we simply use a multiplicative term λ_k that represents the contrast change across the patch. When pixels from the best chosen patch $\Psi^*_{\hat{\mathbf{p}}}$ are to be copied into the timeline mosaic $\mathbf{M}_{time}$, λ_k is estimated by least squares minimization over the overlapping pixels. This correction is applied before the missing pixels are filled in.

Figure 3 focuses on the result of inpainting with and without photometric alignment on a section of the Building C sequence. When compared to [5], the mosaics generated using this technique appear much more consistent and visually pleasing. Even though we only model contrast changes, it is sufficient when applied to a small patch and is able to propagate to a much larger scale.

Positive examples of building patches

Negative examples from RANSAC outliers

Fig. 4. Instances of 11×11 patches from the Building A sequence used for training the classifier

2.4 Heuristics for Spatial Inpainting in Stage 2

Mosaic pixels that were never imaged in the timeline are detected in Stage 1 and marked as a hole - to be completed in Stage 2 by a general spatial inpainting algorithm. Given that most of the background has been recovered in Stage 1, only a small fraction of pixels require conventional inpainting. We use the algorithm of [2] with a few heuristics derived from domain knowledge. Firstly, we search within the warped sequence $\mathbf{W}$ rather than the result of Stage 1 to improve the likelihood of finding a good match. Secondly, since building facades exhibit grid-like patterns, we limit the SSD search to lie within a horizontal and vertical grid around the target patch. This serves to speed up the search through the sequence and reduce the chances of picking a wrong patch to copy into the hole.

3 Measuring Building Likelihood

3.1 Appearance Modeling

The temporal information available from the timeline has already limited the candidate pixels that can be copied to a small set of aligned image patches in the stack $\mathbf{W}$. It was suggested in our previous work [6] that the appearance matching problem could be reformulated as classifying patches in a timeline as building or foreground. Compared to the exhaustive search over Φ using SSD, this could potentially be much more efficient. We now explain our design of a classifier that can disambiguate between building and foreground patches.

Training Set. Most classification and semantic recognition algorithms [15, 16, 17] rely on supervised training with a user manually marking out positive and negative examples. Learning from a universal set of examples does not seem appropriate to our task as the nature of building and non-building patches could vary with location or even seasons. We instead use motion cues to generate training examples specific to the particular sequence - thus bootstrapping the learning for static patch classification. Positive examples of building patches are selected from the MAD inliers Φ by extracting $n \times n$ patches spaced by m pixels on a regular grid (patches overlap when $m < n$). The patch size for learning

and inpainting n was typically 9 or 11, and the spacing varied from $m = 3$ to $m = n$. In addition, we detect Harris corners on the MAD inliers and add them to the positive examples, with the hope of capturing the most important building features also.

The negative examples belonging to trees, grass and so on are harder to extract. Rather than manually demarcate regions, we use the RANSAC outliers from the image registration and mosaicing stage as possible examples of non-building patches. This assumption is reasonable if the dominant motion from frame-to-frame is that of the building. Even if a small set of RANSAC outliers fall on the building - as it does in our case, the classification algorithm should be robust to these incorrect labellings. Figure 4 shows a few examples of 11×11 patches that were used for training. In a similar vein, if the MAD inliers alone are unable to capture the salient building features due to excessive occlusion, we add the RANSAC inliers to the training set as positive examples - albeit at the risk of introducing more errors into the training set.

Modeling Pixel Color. An obvious first step is to experiment with the discriminative ability of pixel color in separating the background from the foreground. If indeed a statistical model can be learnt from the RGB values, it would have the advantage of being spatially and temporally fine-grained. Tracking and segmentation algorithms [18] have used Gaussian Mixture Models to model color distributions. We take the set of RGB values from the training set patches to model the foreground (F) and background (B) as two Gaussian distributions described by means μ_f, μ_b and covariances Σ_f, Σ_b. The computation of the foreground likelihood for pixel y_i can now be framed as

$$p(y_i|F) = \frac{1}{Z_f} \exp\{-\frac{1}{2}(y_i - \mu_f)^T \Sigma_f^{-1}(y_i - \mu_f)\} + \epsilon \tag{2}$$

where $Z_f = (2 * \pi)^{3/2} \Sigma_f^{1/2}$. The background likelihood is also computed in a similar manner. Given the RGB values of a pixel, the probability of it belonging to the building background is computed as $P(y_i = B) = \frac{p(y_i|B)}{p(y_i|B)+p(y_i|F)}$. Despite the lack of context, it provides a strong cue which is first used to refine any obvious errors in the thresholded MAD inliers, especially in the homogeneous regions.

Visual Features. Given several labeled image patches belonging to the building and foreground, we wish to build an empirical model of building and non-building features for the particular sequence. Our previous technique [6] used only color and intensity of a patch to measure its appearance similarity to other building patches. The $n \times n$ size patches are reduced to a low-dimensional space using Principal Component Analysis (PCA) which maximizes the scatter of all the training samples. However, PCA has various limitations (Gaussian distribution, orthogonal linear combinations etc.), which could make it less applicable despite its simplicity. We observe that building patches are likely to contain prominent linear structures with a majority of them oriented horizontally or vertically. In

addition, there could be texture variation in the brick or leaf that color alone might not capture - even at the patch level.

Filter banks have been widely used for texture recognition [19, 15] as well as object/scene categorization [16, 17]. We employ the common Base Filter Set (BFS) used by [15] with 34 filters (8 orientations at 2 scales for 2 oriented filters, plus 2 isotropic). Color patches are converted to grayscale before the convolution is performed. In addition, we append the mean R, G, and B color values across each channel to obtain a 37-dimensional feature vector. For better classification results, the input attributes were mapped to be within [-1,1] and the responses normalized to prevent some attributes from dominating over others.

Appearance-Based Building Likelihood. The classifier used should be robust to outliers in the training set and also generalize well over the training examples without being too sensitive to the number of patches used for training. We explore the two methods described below.

Nearest neighbor classifier: Given a test patch $\Psi_{\mathbf{y}}$, we can classify it as belonging to class $\hat{\nu}$ that has the maximum posterior probability. The patch $\Psi_{\mathbf{y}}$ is first projected into the k-dimensional feature space. Let $(< \mathbf{x}_1, V(\mathbf{x}_1) > \ldots < \mathbf{x}_N, V(\mathbf{x}_N) >)$ be the N nearest neighbors and their associated labels from the training examples. Then we return a distance weighted likelihood

$$p_{app}(\Psi_{\mathbf{y}}) = \frac{\sum_{i=1}^{N} w_i(\Psi_{\mathbf{y}}, \mathbf{x}_i)\delta(Building, V(\mathbf{x}_i))}{\sum_{i=1}^{N} w_i(\Psi_{\mathbf{y}}, \mathbf{x}_i)}$$

where $w(\cdot, \cdot)$ is the reciprocal of the Euclidean distance between the two patches and $\delta(a, b) = 1$ if $a = b$ and 0 otherwise.

Support Vector Machine: A Support Vector Machine (SVM) [20] classifier was trained on the set of patches to learn a model for building and foreground appearance. The 37-dimensional response of the filter bank was fed as the feature vector to SVM. The data was normalized for more accurate results. We used SVM with an RBF kernel to map the training vectors to a higher dimensional space - the best γ being chosen by cross-validation experiments. The freely available SVM-Light package was used for training and classification.

3.2 Motion Cues

The appearance likelihood method is coarse-grained since it operates on patches using the whole neighborhood of pixels for support. This can result in patches with a very small fraction of foreground pixels to be classified as building - a common problem around the edges of leaves or branches. The color models, in spite of working at pixel resolution, are not reliable enough as they completely disregard context. To complement this, we employ motion cues that are spatially fine-grained but temporally coarse.

The intersection of a pair of successive, thresholded difference images was suggested in [21] as a method for identifying foreground pixels. By converting

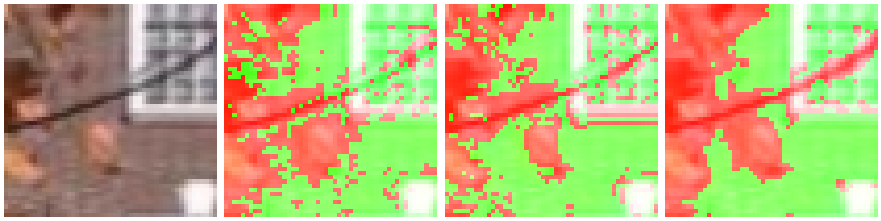


Fig. 5. Comparison of pixel classification on zoomed in section of a frame based on PCA and filter banks(FB) with k-nearest neighbor (KNN,$k = 10$) and SVM. Green shade indicates building while red regions indicate pixels classified as foreground.

the warped images to grayscale and scaling their intensity values to $[0, 1]$ to get $\{\mathbf{W}'_t\}$, we can adapt this approach to define a motion energy or *foreground image* at time t. Also, $\bar{p}_{motion}(\Psi^t_{\tilde{\mathbf{p}}})$ for a patch is the fraction of pixels in $\Psi^t_{\tilde{\mathbf{p}}}$ with background likelihood above a minimum threshold τ_{motion}.

4 Results

4.1 Classification

The quality of the static classifier in distinguishing between building and non-building pixels based on the local statistics in a patch would have a direct impact on the spatiotemporal inpainting approach of [6], not only in preventing small foreground pixels from bleeding into the final mosaic, but also to pull in building patches that the motion cues would otherwise discard due to its temporal coarseness.

Figure 5 zooms in on a small section of a frame and the corresponding result of classification based on PCA, filter banks and SVM. The classification is done independently on patches centered around every pixel. The training for Building A was done with a total of 2908 patches, out of which 2169 were positive examples from MAD and RANSAC inliers. The decision to classify a pixel as foreground or background with k-nearest neighbor was made by setting a threshold of 0.8 on p_{app}. These numbers were arrived at after running the leave-one-out test with a series of parameters and the model with the least percentage of error was chosen.

A few key factors can be noted. All methods seem to detect most of the foreground or tree pixels. PCA essentially works on the RGB color values and might not be able to pick up some of the high frequency variations that the filter bank can. Texture-based classification is thus able to do marginally better on some of the tiny leaves or thin branches that occlude the building. SVM seems the cleanest among all three methods for segmentation. It can generalize well over the training set with several examples and the classification is done in fixed time. In contrast, nearest neighbor approaches become very inefficient as we add more patches.

Every third frame from 6 to 18 in the 22 frame sequence

(a) (b)

(c) (d)

Fig. 6. Building B sequence. (a) Median mosaic. (b) MAD outliers refined by color model that will be fed to the inpainting algorithm for filling. (c) Result after timeline inpainting (Stage 1). (d) Result after spatial inpainting (Stage 2).

For quantitative results, we ran a leave-one-out test on the 2908 patches for Building A. The best result using PCA was 15.1% error with 10-nearest neighbors and a distance threshold of 0.85. The lowest error using filter banks was 11.2% under the same settings. The best accuracy by far was obtained with SVM which misclassified only 3.6% of the training examples. On closer inspection, it was observed that most of these errors boiled down to incorrect labels in the training examples itself.

4.2 Spatiotemporal Inpainting

Each of the appearance models as well as photometric alignment was integrated into the inpainting framework of [6] for comparison. For lack of space, we only show results using SVM learnt on the filter responses. Similar to Fig. 2, figures 6 and 7 illustrate the output at various stages for two other building sequences.

Compared to the SSD metric used in [5], the three appearance models and classification results demonstrate the feasibility of using them to make a hard decision of background or foreground. Contrastingly, this is not possible using SSD and one can only rely on the motion/parallax information that requires a

Every alternate frame from 6 to 14 in an 18 frame sequence

(a) (b)

Fig. 7. Building C sequence. (a) Median mosaic. (b) Result after spatial inpainting (Stage 2).

background pixel to be imaged in three consecutive frames. The classification approaches are clearly much more efficient (SSD required 778 seconds) since the slowest of the 3 methods (FB+KNN) required only 115.3 seconds with definite improvements in the quality of the mosaic.

5 Conclusion

We have described a method of training appearance-based classifiers from a coarse RANSAC-based segmentation to recognize static imagery. The primary motivation behind this work was to identify high frequency features that motion-based approaches alone could not consistently capture. However, we use motion to bootstrap the learning and generalize over the training examples. We have applied the results of our models to both static scene segmentation as well as inpainting to recover texture maps of occluded building facades. Various types of visual features - both intensity-based and texture-based - were proposed in order to learn a low-dimensional representation of image patches to aid in accurate classification of image patches.

Inpainting a building facade is very hard due to the high amount of structure accentuating slight errors or misalignments. Most existing algorithms would not be able to recover the whole facade behind the tree - and we have shown promising results using appearance and motion cues. From the results at various stages of the process, it is obvious that Stage 2 spatial inpainting is the weakest. This is because most inpainting algorithms rely on greedy matching of patches. A single bad choice can propagate errors without any backtracking procedure. For example, the second window from the left in the upper story of Building B reveals an error in the filling in. Our higher level semantic knowledge suggests that a small white patch should be copied on the upper left edge of the window,

but this kind of reasoning is not built into the inpainting. Our future work involves incorporating such abilities into the hole-filling. At a lower level, we wish to experiment with gradient-domain methods [11] for blending patches and even building entities like whole windows or doors. If the higher-level module reasons from a single image that a window is completely occluded by a tree, we would like to first determine where the window ought to appear and seamlessly copy a whole window from a different part of the image.

References

1. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: SIGGRAPH. (2000) 417–424
2. Criminisi, A., Pérez, P., Toyama, K.: Region filling and object removal by exemplar-based image inpainting. IEEE Trans. Image Processing **13** (2004)
3. Jia, J., Wu, T., Tai, Y., Tang, C.: Video repairing: Inference of foreground and background under severe occlusion. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition. (2004)
4. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition. (2004)
5. Rasmussen, C., Korah, T.: Spatiotemporal inpainting for recovering texture maps of partially occluded building facades. In: IEEE Int. Conf. on Image Processing. (2005)
6. Korah, T., Rasmussen, C.: Pca-based recognition for efficient inpainting. In: Proc. Asian Conf. Computer Vision. (2006)
7. Tommasini, T., Fusiello, A., Trucco, E., Roberto, V.: Making good features to track better. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition. (1998) 178–183
8. Efros, A., Freeman, W.: Image quilting for texture synthesis and transfer. In: SIGGRAPH. (2001)
9. Bornard, R., Lecan, E., Laborelli, L., Chenot, J.H.: Missing data correction in still images and image sequences. In: ACM Multimedia. (2002)
10. Szeliski, R.: Video mosaics for virtual environments. IEEE Computer Graphics and Applications **16** (1996) 22–30
11. Prez, P., Gangnet, M., Blake, A.: Poisson image editing. In: ACM Transactions on Graphics (SIGGRAPH'03). (2003) 313–318
12. Kim, S.J., Pollefeys, M.: Radiometric alignment of image sequences. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition. (2004) 645–651
13. Capel, D., Zisserman, A.: Computer vision applied to super resolution. IEEE Signal Processing Magazine **20** (2003) 75–86
14. Jin, H., Favaro, P., Soatto, S.: Real-time feature tracking and outlier rejection with changes in illumination. In: Proc. Int. Conf. Computer Vision. (2001) 684–689
15. Varma, M., Zisserman, A.: A statistical approach to texture classification from single images. International Journal of Computer Vision: Special Issue on Texture Analysis and Synthesis **62** (2005) 61–81
16. Winn, J., Criminisi, A., Minka, T.: Object categorization by learned universal visual dictionary. In: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 2. (2005)
17. Lu, L., Toyama, K., Hager, G.D.: A two level approach for scene recognition. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition. (2005) 688–695

18. Stauffer, C., Grimson, W.E.L.: Learning patterns of activity using real-time tracking. IEEE Trans. Pattern Anal. Mach. Intell. **22** (2000) 747–757
19. Leung, T.K., Malik, J.: Recognizing surfaces using three-dimensional textons. In: ICCV. (1999) 1010–1017
20. Joachims, T.: Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods: Support Vector Learning. MIT Press (1999)
21. Toyama, K., Krumm, J., Brumitt, B., Meyers:, B.: Principles and practice of background maintenance. In: Proc. Int. Conf. Computer Vision. (1999)