Search-and-replace editing for personal photo collections

Citation: Hasinoff, S.W. et al. "Search-and-replace editing for personal photo collections." Computational Photography (ICCP), 2010 IEEE International Conference on. 2010. 1-8. Copyright © 2010, IEEE

As Published: http://dx.doi.org/10.1109/ICCPHOT.2010.5585099

Publisher: Institute of Electrical and Electronics Engineers

Persistent URL: http://hdl.handle.net/1721.1/63165

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Search-and-Replace Editing for Personal Photo Collections

Samuel W. Hasinoff Martyna Jóźwiak Frédo Durand William T. Freeman
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory

Abstract

*We propose a new system for editing personal photo collections, inspired by search-and-replace editing for text. In our system, local edits specified by the user in a single photo (*e.g.*, using the "clone brush" tool) can be propagated automatically to other photos in the same collection, by matching the edited region across photos. To achieve this, we build on tools from computer vision for image matching. Our experimental results on real photo collections demonstrate the feasibility and potential benefits of our approach.*

1. Introduction

Editing large photo collections can be a tedious task. With current tools, more sophisticated edits—such as retouching a facial blemish—must be applied in each photo individually. The cost of editing is felt even more sharply by professional photographers, who routinely capture thousands of photos when shooting portrait sessions, or when covering weddings.

Inspired by the *search and replace* feature in text editing, we propose a system that allows the user to make local edits in a reference photo and automatically propagate those edits to other photos in the same collection. Just like in a text editor, the user can accept or reject a given "replace".

While our approach can be used to enable more general edits, we focus on the most common edit used for image retouching—transferring image content between source and destination regions in the image. Our system implements both the "clone brush" tool, which copies RGB values directly, as well as its "healing brush" variant, which uses gradient-based blending to minimize seams [24, 14].

The key challenge addressed by our system is finding the detailed geometric correspondence matching the regions being edited between the reference photo and a new target image. Successfully transferring an edit entails matching both the source and destinations regions of the edit, even in the presence of varying subject pose and illumination. To meet this challenge, we build on techniques from computer vision for image matching.

After our system transfers the edit regions into the target image, the edit itself is applied wholly within the target. This has the useful feature of preserving the low-level properties of the target, including its defocus, noise grain, and color balance.

In contrast to less structured internet-based photo collections (*e.g.*, [18, 28]), we restrict our attention to personal photo collections (*i.e.*, photo "albums"), where all photos are captured by one photographer in a single session. Since all photos are taken at approximately the same time and location, our matching problem is potentially easier.

Our main contribution is a new system that provides search and replace functionality for photo collections. Absent such a tool, performing local edits across large photo collections would be prohibitively expensive. A side benefit of our system is that it helps maintain consistency over a photo collection being edited. Furthermore, as a consequence of the reduced effort needed for editing, our system could allow photographers to be less aggressive at rejecting photos from their collections.

1.1. Related work

For basic edits, like overall tonality adjustment, batch processing to apply the same edit over multiple photos can be useful [1]. Local edits, however, are more challenging to transfer between photos, since the subject being edited may move in the frame, change pose, or be differently illuminated. Despite recent advances in image matching [30, 27, 28], there has been limited work on propagating such local edits over photo collections.

Face-specific edit transfer. For the specific application of face retouching, search-and-replace editing along the lines we propose has recently been demonstrated: edits to a reference face can be transferred to new faces using landmarks given by a face detector [16], or specified manually [29]. In related work, a facial blemish detector and retouching edit was learned from labeled examples, and then the recovered operation was applied automatically to a new collection of face images [7]. Here the edit is "trained" once and for all, and does not follow a user's specification.

While our system incorporates face detection as well, we rely mainly on generic local image features, which enables

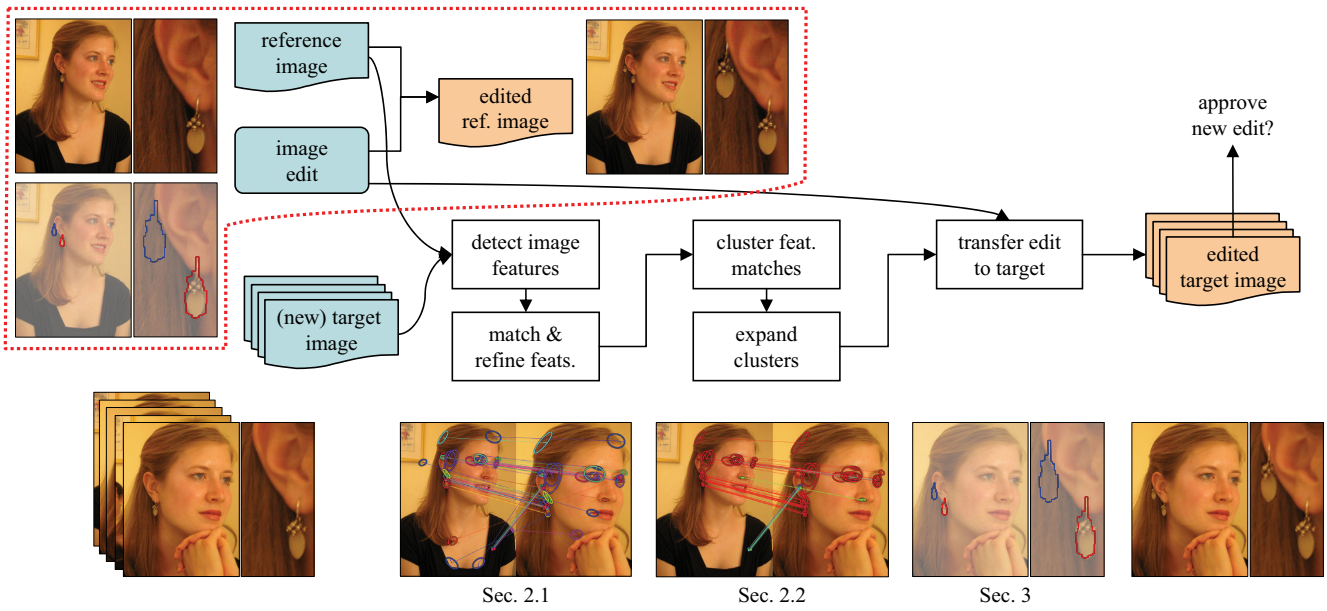


Figure 1. System overview. The user specifies edits in a reference image (in the example shown, replicating the subject's earring), and our system transfers those edits to the rest of the photo collection automatically. The outlined steps correspond to standard single-photo image editing. For each new target image, we match image features between the reference image and the target (Sec. 2). The detected correspondence lets us transfer the edit to the new image (Sec. 3), while preserving its intended meaning. Analogous to search and replace for text editing, we present the results of each edit to the user for approval.

us to match and edit arbitrary subjects. In our application, faces should be easier to relate, since a given edit is applied to the same person in each photo.

Within-photo search and replace. A variation on search-and-replace editing can apply *within* a single photo, where edits to one part of the image can be made to propagate across repeated structures. In early work, such a technique was described for vector illustrations [19], but the well-defined primitives in this context are much easier to match than real images.

Special approaches have also been proposed for editing "texture" images consisting of repeated elements. For more stochastic textures, patch-based matching has been used to define self-similarity within an image [8, 4]. For more regular textures, individual repeated elements can be detected explicitly [20].

Closer to our approach, Glassner proposed a method for replacing repeated objects in a single photo, using a new object from another source [15]. Unlike our automated system, his method requires a good initialization for each match to be provided manually.

Object removal. For the looser editing task of filling in a missing region in a single image, the only constraint is that the completed image look plausible. Recent methods have exploited large unstructured internet-based image collections to search for the most compatible image region to blend in [18, 6]. Like our system, these methods involve finding matches in image collections, but their notion of matching is weaker and extends across object class.

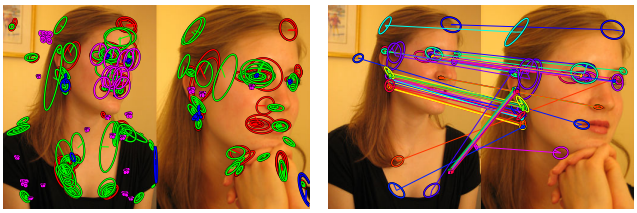
Editing video and multi-view stereo. Our system is also related to video editing methods where edits in one frame are propagated to other frames using automatic object tracking [3, 25]. In a sense, the photo collections we treat can be thought of as a very sparse, non-uniformly sampled video, where object motion is no longer continuous and image matching may be significantly more challenging.

Edit propagation has also been explored in the more constrained context of multi-view stereo [26], where all frames can be related according to a common rigid 3D geometry.

1.2. Overview

As described in Fig. 1, our system for editing photo collections is straightforward from the perspective of the user. After the user specifies edits in a single reference image, the edits are propagated automatically to the rest of the photo collection. Just like in search and replace editing for text, the only remaining interaction is to verify the correctness and desirability of each of the transferred edits.

Internally, the key task of our system is to compute detailed geometric correspondence for scene structure shared between the reference image and each target image (Sec. 2).



all features feature matches

Figure 2. **Left:** Local feature detections in the reference and target image from Fig. 1, visualized by their associated affine frame. Features types are indicated by color: Harris-affine (red), Hessian-affine (green), MSER (blue), and face detection (magenta). **Right:** Candidate feature matches (colored randomly), after refinement. We match features whose descriptors are mutually closest, provided that the match meets our criterion for distinctiveness.

Our approach builds on feature-based methods from computer vision (Sec. 2.1). To find corresponding parts of the scene, we use the local geometry of individual features and cluster together features mutually explained by planar 3D structure (Sec. 2.2).

After computing detailed matching between in the reference and the target image, our system associates the edit regions with particular scene structures, and then uses their geometric correspondence to transfer the edit into the target image (Sec. 3). After applying the edit, we present the edited image to the user for approval.

2. Image Matching

The key component of our system is an automatic method to match the edited regions between images, and recover their geometric correspondence. To this end, we build on an approach widely used in computer vision—matching sparse sets of invariant local features (Sec. 2.1). This approach has shown to been successful for a variety of applications, including object detection [21, 27], geometric registration [9, 28], and generic object recognition [12].

The benefit of using local features is their ability to provide reliable matches, even in the presence of moderate changes in viewpoint, object pose, and lighting. In our application, we treat these features in groups (Sec. 2.2), where clusters of consistent features define local geometry that can be used to transfer the desired edit.

Since the image edits in our application are associated with particular people or objects, we require less flexible matching than for recognition across object class [12]. On the other hand, our system must handle greater subject deformation compared to static architectural scenes [28] or landscapes [9].

2.1. Feature detection and matching

Our system combines three different types of generic local image features, plus a special-purpose face detector (Fig. 2, left). As previous methods have shown [27], using multiple types of features provides complementary information about stable image structures.

Affine-invariant features. We use the popular implementation from Oxford [2] to detect three types of generic image features: (1) Harris-affine features for corners, [23], (2) Hessian-affine features for blob-like structures [23], and (3) maximally stable extremal regions (MSER) for more irregularly shaped regions [22]. All the feature detectors we use are affine-invariant, so the structures they detect should be recoverable up to affine deformation [30].

In addition to the position of each feature point $\mathbf{x}_i$, we recover the attached 2×2 affine frame $\mathbf{T}_i$ in which the feature was detected. This frame incorporates both an affine warp and a designated direction (illustrated in the figures by ellipses with designated radii), so that $\mathbf{T}_i(\mathbf{p}-\mathbf{x}_i)$ maps image point $\mathbf{p}$ to normalized coordinates in the feature's frame. In contrast to methods that use feature positions exclusively [27, 9, 28], we found it useful to analyze geometry using their affine frames as well (Sec. 2.2).

Note that the correspondence between a single affine frame over two images, $(\mathbf{x}_i^1, \mathbf{T}_i^1) \leftrightarrow (\mathbf{x}_i^2, \mathbf{T}_i^2)$, is sufficient to establish the full geometric relationship of all surrounding points [30]. As a result, it is possible to transfer an edit between images based on a single feature match, using

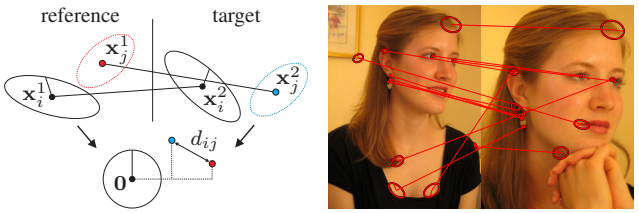
$$\mathbf{p}^2 = \mathbf{x}_i^2 + (\mathbf{T}_i^2)^{-1}\mathbf{T}_i^1(\mathbf{p}^1 - \mathbf{x}_i^1)\ . \qquad (1)$$

As discussed in Sec. 3, we found that the affine frames from feature detection are generally too imprecise, even after refinement, to be used directly for the purpose of editing.

Feature matching. To match feature detections between images, we represent the features using their 128-dimensional SIFT descriptors [21], based on histograms of differently-oriented gradients. Since these descriptors are computed in the affine frame of each feature, feature matching is invariant to local affine deformation. The SIFT descriptor also includes intensity normalization for invariance to illumination changes.

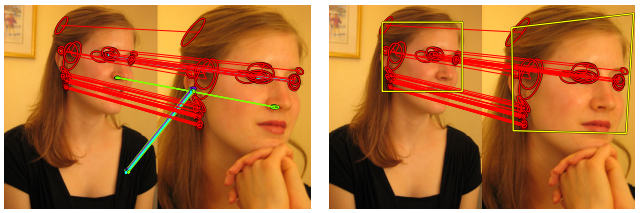
We search for candidate feature matches by evaluating L_2 distances between feature descriptors in the reference and target images, treating each of our three types of image features separately. To prune the set of candidates, we only retain matches that are mutually the best match from the point of view of both images. As is common, we impose a distinctiveness criterion as well [21], requiring that the descriptor distance for the best match be below some fraction τ_{ratio} of the second-best match (we use $\tau_{\text{ratio}} = 0.95$).

Feature refinement. Once an initial set of feature matches has been established, we refine the positions and affine frames of corresponding features, based on a direct



rejected matches

Figure 3. **Left:** Testing the consistency of a pair of feature matches (i, j), using the geometry of their affine frames. For each image we transform feature point j into the affine frame of feature i, then measure the distance between the transformed points. **Right:** Feature matches from Fig. 2 rejected by our pairwise consistency test.



feature clusters cluster for edit transfer

Figure 4. **Left:** Expanded feature clusters, labeled by color, as recovered from Fig. 2. We cluster feature matches whose centers approximately define a homography, and whose affine frames are consistent as well. **Right:** Cluster used to transfer the edit in Fig. 1, with the associated homography shown by the overlaid quadrilaterals.

similarity measure between RGB pixels in each frame [13]. We found that this refinement can significantly improve the accuracy of the local geometry, compared to the affine frames returned by feature detection, which are computed in each image independently

Face detection and matching. While generic local image features are useful for matching arbitrary parts of the scene, we found that augmenting our system with face-specific features improved robustness for edits in the vicinity of faces. Our implementation uses the frontal face detector described in [11]. This method incorporates a flexible parts-based model to identify 9 points per face detection, corresponding to specific facial features, all in a common affine frame. For faces matched between the reference and target image, we add their corresponding feature points to the overall list of matches.

To match faces, we use the proposed pixel-based face descriptor [11]. Analogous to our matching criteria for generic image features, we search for face matches whose descriptor distance is lowest in both directions, and which must pass our distinctiveness test (again using $\tau_{\text{ratio}} = 0.95$) if more than one face is detected.

While face detection can lead to false positives (for example, the cyan cluster in Fig. 4, left), these incorrect matches can generally be filtered out later, unless there are few reliable matches for other feature types.

2.2. Feature clustering

While the local feature matches we recover are a good starting point, they only provide a sparse correspondence between the reference and target image, and outliers generally remain. To improve robustness, we recover higher-level scene structure by clustering these features into geometrically-consistent groups. We do this by filtering out features whose affine frames are mutually inconsistent (Fig. 3), and clustering feature matches whose image positions are well explained by a common underlying 3D planar geometry (Fig. 4).

Pairwise feature consistency. As a pre-filtering step, we eliminate the feature matches that are inconsistent with all others. This reduces the number of feature matches to search, which can later improve the quality of clustering.

To test the consistency between a given pair of feature matches, we measure the position of one feature in the affine frame of the other, and compare its coordinates between images (Fig. 3, left). More specifically, for feature pair (i, j), we define the asymmetric measure

$$d_{ij} = \|\mathbf{T}_i^1(\mathbf{x}_j^1 - \mathbf{x}_i^1) - \mathbf{T}_i^2(\mathbf{x}_j^2 - \mathbf{x}_i^2)\| . \qquad (2)$$

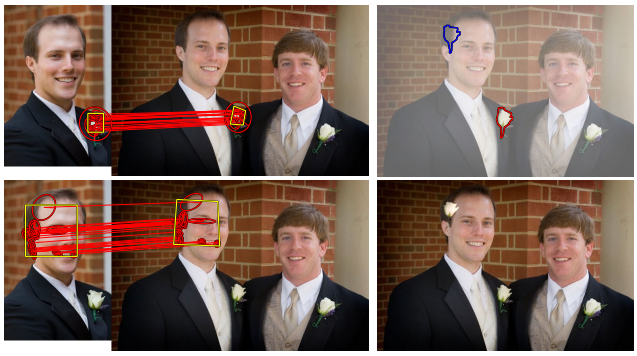
For a pair of matches to pass this test, both d_{ij} and d_{ji} must be below τ_{consis}, all of which are measured in normalized feature coordinates. We use a very loose threshold ($\tau_{\text{consis}} = 2$) so that only clear outliers are rejected, but many matches can often still be eliminated (Fig. 3, right).

Homography-based clustering. To cluster the feature matches, we search for sets of features whose correspondence between the reference and target images can jointly be described using a single homography, or plane-to-plane projective mapping [17].

We extract clusters using a RANSAC approach, by randomly sampling quadruples of matching feature positions [17]. For each quadruple, we recover the corresponding 3×3 homography $\mathbf{H}$, and test all other feature matches for consistency. After a set number of trials, we retain the homography with the most inliers, cluster its associated feature matches and remove them from further consideration. We repeat the process until we fail to find new clusters.

For better numerical conditioning, we normalize feature positions to have their centroid at the origin and mean distance to the origin of $\sqrt{2}$. To define an inlier, we test the homography in both directions using the relatively loose threshold of 0.05 in normalized coordinates.

After extracting the clusters, we reapply the above pairwise consistency test to remove features without at least one other consistent feature in the cluster.



clusters for src. & dst. regions of edit edits in the target image

Figure 5. Copying the man's flower to his ear, described in Fig. 7(f). **Left:** Since the source and destination regions of the edit move independently (the man turns his head), our system associates different feature clusters with each. **Right:** To transfer the edit to the target image, we apply the corresponding homographies for each region of the edit separately.

Cluster expansion. To improve the reliability of the clusters, we use the geometry encoded in their homographies to guide the search for additional geometrically-consistent feature matches, along the lines of [13]. In this step we reconsider features that were discarded earlier in the method, for example, because they were not the top SIFT descriptor match, or they failed the distinctiveness criterion.

We expand the clusters conservatively, using a reduced threshold for consistency with the homography (0.025). From the new geometrically-consistent features, we add a match only if it passes a stronger version of the pairwise consistency test—any new feature match must meet at least the median level of within-cluster pairwise-consistency.

3. Transferring Image Edits

Given the computed feature clusters, which describe matching geometrically-consistent parts of the scene, we are nearly ready to transfer the edit from the reference image to the target. The only remaining decision our system must make is what clusters, representing matching scene structure, to associate with the edit.

In many cases, the both the source and destination regions of the "clone brush" style edit lie on the same rigid scene structure, so associating the edit with a single cluster is sufficient (Fig. 4). More generally, these regions may lie on independently moving parts of the scene, so we allow them to be associated with different clusters (Fig. 5).

Cluster selection. We use a simple scoring method to associate each of the edit regions with a cluster. In particular, we score cluster $\mathcal{C}_k$ based on the distance of its features to the centroid of the edit, $\bar{\mathbf{p}}_{\text{edit}}$, according to the formula

$$\sum_{\mathbf{x}_i \in \mathcal{C}_k} \frac{1}{\|\mathbf{x}_i - \bar{\mathbf{p}}_{\text{edit}}\| + \varepsilon}\,, \tag{3}$$

where $\varepsilon = 10$. This metric favors clusters both with many features and close to the edit. While we experimented with more sophisticated metrics for cluster selection, none of them performed consistently better overall.

Homography-based transfer. To transfer an edit region to the target image, we use the homography $\mathbf{H}_k$ for the selected cluster to warp pixels from the reference image. In homogenous coordinates, this simply involves computing $\tilde{\mathbf{p}}^2 = \mathbf{H}_k \tilde{\mathbf{p}}^1$ [17].

Given the new locations of both edit regions, as warped into the target image, we implement the transfer from source to destination using a form of inverse warping. To achieve more seamless transfer we use Poisson image editing [24, 14], which consists of transferring gradients rather than pixel values, and then re-integrating the result while respecting the boundaries of the destination region.

Non-linear transfer. Our initial scheme for transferring edits was inspired by Beier-Neely warping [5], and it directly used the affine frames of feature matches in the selected cluster to carry out the warping. In theory, this would have allowed us to capture non-linear deformations not captured by a homography, but we found that the affine frames of individual feature matches were not accurate enough for image editing, even after refinement [13]. Particularly problematic were orientation estimates, whose inaccuracy is magnified away from the feature. By adding outlier rejection we were able to achieve passable results with this approach, but we found that the simpler homography-based transfer led to more robust performance.

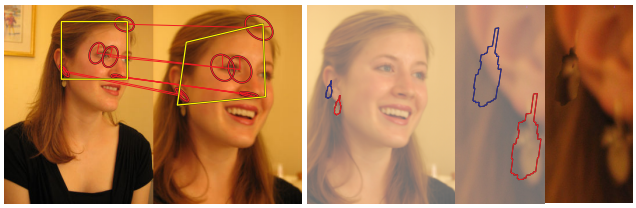
4. Experimental Results

We evaluated our system by editing a variety of real photo collections corresponding to single photography sessions, collected from Flickr, Picasaweb, and our personal archives. As shown in Fig. 7, we explored different types of edits, including face retouching and cloning arbitrary objects. For additional results and full-resolution images, please see the supplementary materials.

For 1 Mpixel images, our prototype takes approximately 20 seconds for each target image, to match against the reference, cluster features, and transfer the edit. We run feature detection as a pre-processing step, which requires about the same amount of time per image.

Based on our results, we can make four observations about our system. First, the system is generally successful at propagating image edits over photo collections. As

(a) bad image matching



(b) non-planar deformation

(c) occlusion in the target image

Figure 6. Representative failure cases for the photo collections shown in Fig. 7. (a) Feature clusters for two examples where image matching is unsuccessful. The limited number of good feature matches allow the outliers, mainly due to incorrect face detection, to dominate. (b) When subject deformation is modeled poorly by the best-fit homography, the transferred edit may be inaccurate. In this example, the change in facial expression and viewpoint causes the transferred edit to be aligned poorly in the target image. (c) Even when the geometry of the transferred edit is correct, our system fails to account for partial occlusion in the destination region of the edit.

expected, the best case for our method is editing a rigid, distinctively textured object (Fig. 7(b-c)). For more challenging cases (Fig. 7(a,d-f)), our system fails more often, typically due to poor image matching.

Second, the edited region of the image itself does not need to possess distinctive local image features. Rather, it is sufficient for the edit to be well-localized relative to other features on the same part of the scene. This is the case for Fig. 7(b), where the facial blemish being retouched is not distinctive enough to trigger local feature detection, but its location is well constrained by surrounding facial features.

Third, our homography-based transfer can yield geometrically consistent results in the presence of perspective changes (Fig. 7(b,c,e)), despite the fact that the source and destination regions of the edit are offset by a simple translation in the reference image. Note that when editing photos manually, achieving consistent perspective is difficult using a "clone brush" style tool alone.

Finally, since the edits themselves are carried out completely within each target image, our editing results preserve the low-level appearance of the target. This can be seen in the closeups of Fig. 1, where the edit preserves subtle details of the target image, including slight blur due to camera shake and JPEG compression artifacts.

5. Discussion

While our system can propagate edits over a broad range of photo collections, and handle various types of local edits, its performance is far from perfect. Particularly for less distinctive subjects, or for subjects exhibiting greater variation, the overall effectiveness of our system depends on the fact that the user remains in the loop, and must ultimately approve each of the edited images. To better understand the limitations of our system, as well as opportunities for improvement, we detail its three main failure modes, illustrated in Fig. 6.

Bad image matching. The most common failure mode for our system is due to bad image matching, shown in Fig. 6(a). This typically occurs when the set of candidate matches contains few inliers, in turn causing our RANSAC-based clustering method to be unstable. Even a single incorrect feature match in the selected cluster can corrupt the recovered homography.

Despite being an obvious parameter to adjust, we found that changing the sensitivity of feature detection had a limited ability to improve matching performance. For example, detecting a larger number of image features can actually be counterproductive, since correct feature matches may become even less distinctive.

The key issue is not just the lack of enough correct feature matches, but also the inability of our clustering method to handle very few inliers. While we already carry out a post-hoc search for new geometrically-consistent matches (Sec. 2.2), it would be better if clusters could be generated more robustly. One promising direction is to search for consistent geometry in more incremental way [13, 10], which may allow us to recover a denser and more flexible correspondence between images.

Non-planar deformation. A second common type of editing failure occurs when the image features we use for matching are not well approximated by a rigid plane, as in Fig. 6(b). Because the homographies we use for transferring the edit do not capture out-of-plane deformations, any such deformations can lead to inaccuracy.

To remedy this problem, we could potentially refine the transferred edit in an additional post-processing step, reminiscent of the method we use to refine affine feature matches

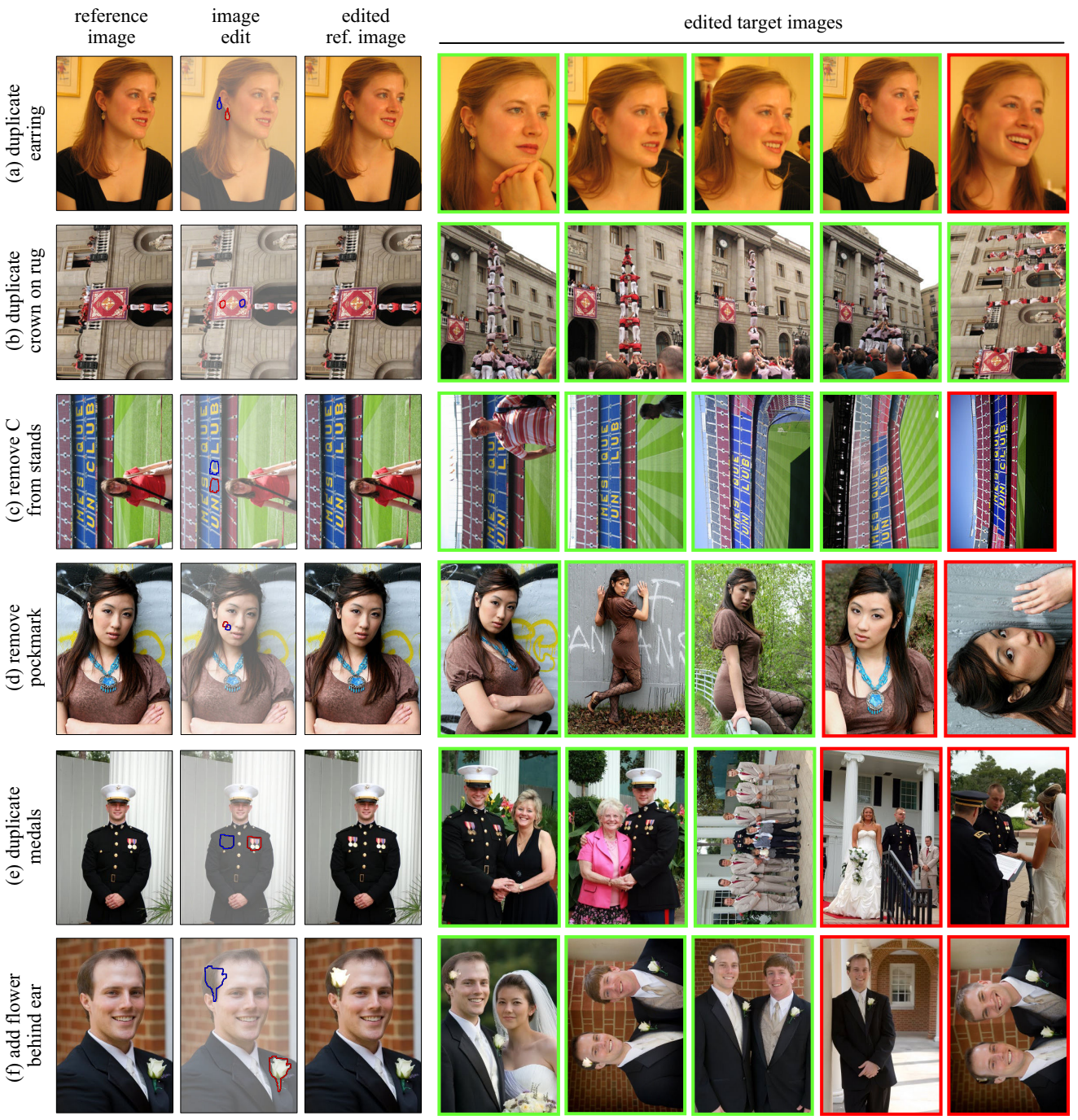


Figure 7. Gallery of results. Target images successfully edited using our system (judged subjectively) are outlined with green, and unsuccessful edits are outlined with red. Our system is capable of both face retouching (d) and editing arbitrary objects (a-c,e-f). The system also allows us to achieve perspective-consistent editing results, even in the presence of significant perspective changes (b,c,e). While the first four photo collections (a-d) can be handled reliably, the last two (e-f) push the limits of our system, with success rates of about 45% and 35% respectively. As discussed in Sec. 5, failures of our system are often caused by insufficient correct feature matches in the vicinity of the edit.

[13]. In practice, this could involve performing a local optimization, matching gradients in the edit regions between the reference and target images.

Missing or occluded regions. Another limitation of our system is that we currently do not handle cases where regions of the edit are outside the field of view or occluded in the target. This is illustrated in Fig. 6(c), where the destination region of the edit is partially occluded in the target.

In principle, we should be able to detect these situations, again by directly comparing the content of the edit regions between the reference and target images. For cases where part of the edit is missing, we could even fall back to copying information directly from the reference image.

6. Conclusions

In this paper we demonstrated a first step toward propagating generic local edits over personal photo collections. Even in its limited form, our system may still be a useful tool to help ease the burden of image editing. For future work, we are interested in extending the reach of our system by improving the robustness of image matching. We are also interested in testing our system on larger photo collections, and conducting user studies to evaluate its usefulness in practice. More generally, we believe that tools from image matching and object recognition hold great potential to improve the state-of-the-art in photo editing, particularly for editing tasks at the level of an entire photo collection.

Acknowledgments. This work was supported in part by an NSERC Postdoctoral Fellowship, the MIT UROP program, NSF CAREER award 0447561, the Quanta T-Party, NGA NEGI-1582-04-0004, MURI Grant N00014-06-1-0734, and by a gift from Microsoft Research. F. Durand acknowledges a Microsoft Research New Faculty Fellowship and a Sloan Fellowship. Special thanks to Josef Sivic for helpful discussions and sharing code. We also thank the Flickr user `Alaskan Dude` and the Picasaweb users `hadtoshare`, `riggs.brock`, and `ToddandApril08`, for releasing their photos under a Creative Commons license.

References

[1] Adobe Photoshop Lightroom, http://adobe.com/. 1

[2] K. Mikolajczyk, Affine covariant features, http://www.robots.ox.ac.uk/~vgg/research/affine/, Visual Geometry Group, University of Oxford, 2007. 3

[3] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. In *SIGGRAPH*, pp. 584–591, 2004. 2

[4] X. An and F. Pellacini. Appprop: All-pairs appearance-space edit propagation. In *SIGGRAPH*, pp. 1–9, 2008. 2

[5] T. Beier and S. Neely. Feature-based image metamorphosis. In *SIGGRAPH*, pp. 35–42, 1992. 5

[6] D. Bitouk, N. Kumar, S. Dhillon, P. N. Belhumeur, and S. K. Nayar. Face swapping: automatically replacing faces in photographs. In *SIGGRAPH*, 2008. 2

[7] M. Brand and P. Pletscher. A conditional random field for automatic photo editing. In *CVPR*, pp. 1–7, 2008. 1

[8] S. Brooks and N. Dodgson. Self-similarity based texture editing. In *SIGGRAPH*, pp. 653–656, 2002. 2

[9] M. Brown and D. Lowe. Automatic panoramic image stitching using invariant features. *IJCV*, 74(1):59–73, 2007. 3

[10] M. Cho, J. Lee, and K. M. Lee. Feature correspondence and deformable object matching via agglomerative correspondence clustering. In *ICCV*, 2009. 6

[11] M. Everingham, J. Sivic, and A. Zisserman. Taking the bite out of automatic naming of characters in TV video. *IVC*, 27(5):545–559, 2009. 4

[12] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, vol. 2, pp. 264–271, 2003. 3

[13] V. Ferrari, T. Tuytelaars, and L. V. Gool. Wide-baseline multiple-view correspondences. In *CVPR*, vol. 1, pp. 718–725, 2003. 4, 5, 6, 8

[14] T. Georgiev. Photoshop healing brush: a tool for seamless cloning. In *Workshop on Applications of Comp. Vision*, pp. 1–8, 2004. 1, 5

[15] A. Glassner. Image search and replace. *IEEE Computer Graphics and Applications*, 23(3):80–88, 2003. 2

[16] F. Grabler, M. Agrawala, W. Li, M. Dontcheva, and T. Igarashi. Generating photo manipulation tutorials by demonstration. In *SIGGRAPH*, pp. 1–9, 2009. 1

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004. 4, 5

[18] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *SIGGRAPH*, pp. 1–7, 2007. 1, 2

[19] D. Kurlander and E. Bier. Graphical search and replace. In *SIGGRAPH*, pp. 113–120, 1988. 2

[20] Y. Liu, W.-C. Lin, and J. Hays. Near regular texture analysis and manipulation. In *SIGGRAPH*, pp. 368—376, 2004. 2

[21] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 3

[22] J. Matas, Š. Obdržálek, and O. Chum. Local affine frames for wide-baseline stereo. In *ICPR*, pp. 363–366, 2002. 3

[23] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 60(1):63–86, 2004. 3

[24] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. In *SIGGRAPH*, pp. 313–318, 2003. 1, 5

[25] A. Rav-Acha, P. Kohli, C. Rother, and A. W. Fitzgibbon. Unwrap mosaics: a new representation for video editing. In *SIGGRAPH*, pp. 1–9, 2008. 2

[26] S. M. Seitz and K. N. Kutulakos. Plenoptic image editing. *IJCV*, 48(2):115–129, 2002. 2

[27] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, vol. 2, pp. 1470–1477, 2003. 1, 3

[28] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from Internet photo collections. *IJCV*, 80(2):189–210, 2008. 1, 3

[29] W.-S. Tong, C.-K. Tang, M. S. Brown, and Y.-Q. Xu. Example-based cosmetic transfer. In *Pacific Conference on Computer Graphics and Applications*, pp. 211–218, 2007. 1

[30] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2007. 1, 3