# Computational Rephotography



SOONMIN BAE
MIT Computer Science and Artificial Intelligence Laboratory
ASEEM AGARWALA
Abobe Systems, Inc.
and
FRÉDO DURAND
MIT Computer Science and Artificial Intelligence Laboratory

Rephotographers aim to recapture an existing photograph from the same viewpoint. A historical photograph paired with a well-aligned modern rephotograph can serve as a remarkable visualization of the passage of time. However, the task of rephotography is tedious and often imprecise, because reproducing the viewpoint of the original photograph is challenging. The rephotographer must disambiguate between the six degrees of freedom of 3D translation and rotation, and the confounding similarity between the effects of camera zoom and dolly.

We present a real-time estimation and visualization technique for rephotography that helps users reach a desired viewpoint during capture. The input to our technique is a reference image taken from the desired viewpoint. The user moves through the scene with a camera and follows our visualization to reach the desired viewpoint. We employ computer vision techniques to compute the relative viewpoint difference. We guide 3D movement using two 2D arrows. We demonstrate the success of our technique by rephotographing historical images and conducting user studies.

Categories and Subject Descriptors: H.5.2 [**Information interfaces and Presentation (e.g., HCI)**]: *User Interfaces*; I.4.9 [**Image Processing and Computer Vision**]: *Applications*

General Terms: Algorithms, Design, Human Factors

Additional Key Words and Phrases: Computational photography, pose estimation, rephotography

**ACM Reference Format:**

Bae, S., Agarwala, A., and Durand, F. 2010. Computational rephotography. ACM Trans. Graph. 29, 3, Article 24 (June 2010), 15 pages.
DOI = 10.1145/1805964.1805968 http://doi.acm.org/10.1145/1805964.1805968

## 1. INTRODUCTION

Rephotography is the act of repeat photography; capturing a photograph of the same scene from the same viewpoint of an existing photograph that is typically much older. An image and its rephotograph can provide a compelling "then and now" visualization of the progress of time (Figure 1). Rephotography is a powerful tool for the study of history. Well-known examples include *Second View* [Klett et al. 1990] (a rephotographic survey of landscape images of the American West), *New York Changing* [Levere et al. 2004], and a series of forty *Then and Now* books that each rephotograph a major world city (e.g., McNulty [2002]). Beyond history, rephotography is also used to document glacier melting as evidence of global warming [Gore 2006], and to monitor geological erosion and change [Hall 2002].

When a photograph and its rephotograph match well, a digital cross-fade between the two is a remarkable artifact; decades go by in the blink of an eye, and it becomes evident which scene elements are preserved and which have changed across time. To create a faithful rephotograph, the viewpoint of the original photograph must be carefully reproduced at the moment of capture. However, as we have confirmed through a user study, exactly matching a photograph's viewpoint "by eye" is remarkably challenging; the rephotographer must disambiguate between the six degrees of freedom of 3D translation and rotation, and, in particular, the confounding similarity between the effects of camera zoom and dolly. There are digital techniques for shifting viewpoint after capture [Kang and Shum 2002], but they are still brittle and heavyweight, and depend on the solution of several classic and challenging computer vision problems.

This work was supported by grants from Quanta, Shell, and Adobe.
Authors' addresses: S. Bae, MIT Computer Science and Artificial Intelligence Laboratory, MIT, 77 Massachusetts Avenue, Cambridge, MA 02139-4307; email: soonmin@csail.mit.edu; A. Agarwala, Abobe Systems, Inc.; F. Durand, MIT Computer Science and Artificial Intelligence Laboratory, MIT, 77 Massachusetts Avenue, Cambridge, MA 02139-4307.

Fig. 1. Rephotography gives two views of the same place around a century apart. Pictures are from *New York Changing* [Levere et al. 2004] and *Boston Then and Now* [McNulty 2002].

In this article, we present an interactive, computational technique for rephotography that focuses on the task of matching the viewpoint of a reference photograph at capture time. We envision our tool running directly on the digital camera; but since these platforms are currently closed and do not yet have enough processing power, our prototype consists of a digital camera connected to a laptop. The user simply points the camera towards the scene depicted in the reference image, and our technique estimates and visualizes the camera motion required to reach the desired viewpoint in real time. Algorithmically, we build on existing computer vision algorithms to compute the relative pose between two photographs [Stewénius et al. 2007; Hartley 1992] after detecting and matching features [Lowe 2004] common to both images.

The main contribution of our work is the development of the first interactive computer vision tool for rephotography. This tool includes a number of novel techniques, such as a method to calibrate a historical camera without physical access to it by photographing the same scene with a modern, calibrated camera. We also present a stabilization technique that substantially reduces the degrees of freedom that the user needs to explore while following the motions suggested by our tool. We demonstrate the success of our technique by rephotographing historical images and conducting user studies.

### 1.1 Previous Work

To the best of our knowledge, we are the first to build an interactive tool that directs a person to the viewpoint of a reference photograph. However, estimating camera positions and scene structures from multiple images has long been a core problem in the computer vision community [Faugeras 1993; Heyden and Sparr 1999; Hartley 1992; Hartley and Zisserman 2000].

We direct the user to the correct viewpoint at capture time. One alternative to our approach would be to capture a nearby viewpoint and warp it to the desired viewpoint after capture [Chen and Williams 1993; Werner et al. 1995; Sand and Teller 2004]. However, parallax and complex scene geometry can be challenging for these algorithms, and the possibility of inaccuracies means that the result might not be considered a faithful documentation of the viewpoint for scientific or historical purposes.

Our technique is related to *visual homing* research in robotics, where a robot is directed to a desired 3D location (e.g., a charging station) specified by a photograph captured from that location. The visual homing approach of Basri et al. [1999] also exploits feature matches to extract relative pose; the primary difference is that robots can respond to precise motion parameters, while humans respond better to visualizations in a trial-and-error process. More recent work exists on real-time algorithms that recover 3D motion and structure [Pollefeys et al. 2008; Davison et al. 2007], but they do not aim to guide humans. There exist augmented reality systems [Scheuering et al. 2002] that ease navigation. However, they assume that the 3D model is given, while the only input to our technique is an old photograph taken by an unknown camera.

We are not the first to exploit the power of historical photographs. The 4D Cities project (www.cc.gatech.edu/4d-cities) hopes to build a time-varying 3D model of cities, and Photo Tourism [Snavely et al. 2006] situated older photographs in the spatial context of newer ones; neither project, however, helped a user capture a new photograph from the viewpoint of a historical one.

Recent digital cameras and mobile phones employ a number of advanced computer vision techniques, such as face detection, the Viewfinder Alignment of Adams et al. [2008], feature matching and tracking on mobile phones [Wagner et al. 2008; Takacs et al. 2008], and the Panoramic Viewfinder of Baudisch et al. [2005]. The Panoramic Viewfinder is the most related to our technique, though its focus is the real-time preview of the coverage of a panorama with no parallax. The implementation of matching and tracking algorithms on mobile phones is complementary to our technique. We focus on the development of an interactive visualization method based on similar tools.

## 2. OVERVIEW

We designed the user interface and technical approach of our rephotography tool after performing two of initial experiments that helped us understand the challenges of rephotography. In our first pilot user study (Section 6.2.1), we addressed the obvious question: how hard is manual rephotography? We found that users untrained in rephotography were unable to reproduce the viewpoint of a reference image successfully even with the aid of simple visualizations, such as a side-by-side visualization of the current and reference views, or a linear blend of the two views.

In the next study (Section 6.2.2) we implemented a standard relative pose algorithm [Stewénius et al. 2007], and visualized, in 3D, the recovered camera frustums of the current and reference views to users whenever they captured a new photograph (Figure 11(a)). We again found that the users were unsuccessful, because they had difficulties interpreting the visualization into separate translation and rotation actions, and were challenged by the lack of real-time feedback.

These two pilot user studies, along with our own experiments using the tool to perform historical rephotography, helped us identify five main challenges in computational rephotography:

(1) It is challenging to communicate both a 3D translation and rotation to the user, since this motion has six degrees of freedom. Camera zoom adds a seventh degree.

(2) Even with calibrated cameras, 3D reconstruction from images alone suffers from a global scale ambiguity. This ambiguity makes it hard to communicate how close the user is to the desired viewpoint, or to keep the scale of the motion communicated to the user consistent over iterations.

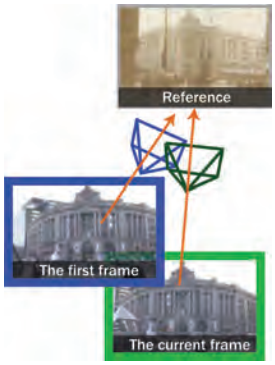


Fig. 2. The first photograph is captured from a location rotated about 20 degrees from the user's best approximation of the desired viewpoint. The second photograph is then captured from the user's best approximation.

(3) Relative pose algorithms suffer from a degeneracy in the case of zero motion between the cameras [Torr et al. 1999], which is exactly our goal. This degeneracy means the estimation becomes unstable as the user reaches the reference view.
(4) Historical images can appear very different from new photographs because of architectural modifications, different film response, aging, weather, time-of-day, etc. These dramatic differences can make it challenging for even state-of-the art feature descriptors to find the correspondences needed to compute relative pose.
(5) Finally, historical images are captured with cameras of unknown calibration, for example, focal length and principal point. Furthermore, historical architectural photographs were often captured with noncentral principal points using view cameras, to make the vertical lines of building vertical in the image.

We address these challenges with a combination of user interaction and algorithms. Our approach has a number of key features which we describe in the rest of this article. The first is our approach to calibration of the unknown camera used to capture the historical image (Section 3), that is, challenge 5. We require the user to capture two images of the scene with a wide baseline (Figure 2). The user is instructed to capture a first frame and second frame with a roughly 20 degree angle about the main scene subject, with the second frame as the user's best eyeballed approximation of the desired viewpoint. We then reconstruct the scene in 3D and use this structure to calibrate the historical camera after asking the user to manually identify a few correspondences with the historical image (challenge 4). We also use this wide baseline to solve challenge 3 by performing pose estimation relative to the first frame rather than the reference view, which helps avoid degeneracy. The computed 3D structure also helps us to compute a consistent 3D scale across iterations (challenge 2). Finally, our calibration method also includes an optional interactive approach to calibrating a noncentral principal point (Section 3.2.1), which asks the user to identify sets of parallel lines in the scene.

Another key aspect of our approach is real-time visual guidance that directs the user towards the desired viewpoint (Section 4). This feedback includes a visualization of the needed 3D translation to the reference view computed by interleaving a slower, robust relative pose algorithm (Section 4.1) with faster, lightweight updates (Section 4.2). We also use the computed relative pose to perform rotation stabilization (Section 4.5); that is, we show the current view to the user after warping it with an infinite homography [Hartley and Zisserman 2000] fit to the reference view, which can account for both camera rotation and zoom. Because of this stabilization the user does not need to worry about precisely rotating the camera, and can focus on following the 3D translation directions from our tool (challenge 1).

Fig. 3. In our prototype implementation, a laptop is connected to a camera. The laptop computes the relative camera pose and visualizes how to translate the camera with two 2D arrows. Our alignment visualization, which consists of edges detected from the reference image composited onto the current view, helps users to evaluate whether they have reached the final viewpoint.

After giving an overview of the user experience of our rephotography tool, the rest of this article describes the technical details of our approach, evaluates it with several user studies, and presents results of using our tool to perform historical rephotography.

## 2.1 User Experience

While we ultimately wish to embed our tool entirely on a camera, our current implementation relies on a laptop connected to a camera as shown in Figure 3. The camera's viewfinder images are streamed to the laptop, which performs computation to visualize the necessary motions to the user.

The user begins by loading a reference image. If the user suspects that the image was shot with a noncentral principal point (e.g., vertical lines on tall buildings are vertical in the historical photograph), she can optionally calibrate the principal point by identifying three sets of parallel lines in the reference image. We allow the use of a screen magnifier to make the lines more visible. Identifying these lines is typically easy for photographs of buildings, which is the most common scenario in which a photographer chooses to manipulate the principal point using a view camera or a tilt-shift lens.

The user is next instructed to shoot an image that is rotated roughly 20 degrees away from the reference viewpoint. The actual amount of rotation is unimportant and is only specified to simplify instructions; it is only important that the baseline from the reference viewpoint be reasonably wide. We call this image the "first frame." Next, the user goes to her best estimate of the reference viewpoint, and shoots a "second frame." The system computes 3D structure from these two frames, and then asks the user to click six correspondences between the reference image and the 3D structure projected onto the second frame. After doing so, the real-time feedback begins and arrows direct the user towards the goal. Alongside the arrows we show the rotation-stabilized current viewpoint. The user can switch between several visualizations (Section 5), such as an overlay of edges detected from the reference image onto the current viewpoint, which can help the user in evaluating whether the current viewpoint is a successful rephotograph. Once the user is happy with the viewpoint, she can capture her final rephotograph.

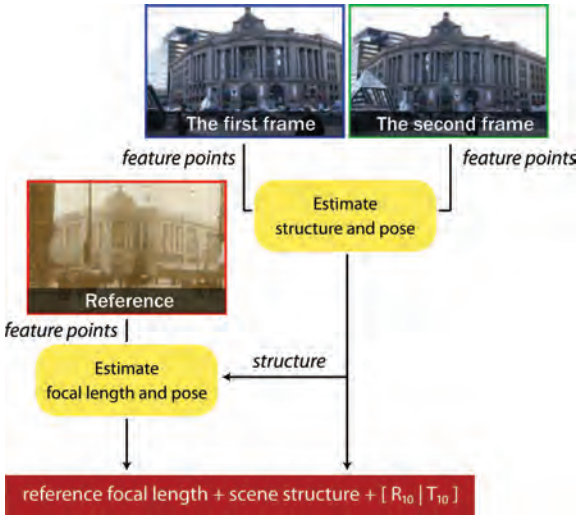


Fig. 4. Our procedure to calibrate and register the reference camera.

## 3. CALIBRATION

The first step of our computational rephotography tool is to calibrate both the intrinsic and extrinsic parameters of the unknown historical camera. We do so by performing a sparse 3D reconstruction of the same scene imaged by the historical camera using two user-captured images, and then optimizing the parameters of the unknown camera to minimize projection error of the features manually corresponded by the user. We also optionally allow the user to calibrate a noncentral principal point by specifying sets of parallel scene lines in the historical image. This process is shown in Figure 4.

### 3.1 Wide Baseline 3D Reconstruction

The user begins by capturing two images (the first and second frames) with a wide baseline (Figure 2); a wide baseline improves the accuracy and stability of 3D reconstruction. We assume the current camera is calibrated (we use Bouguet's calibration toolbox [2007]), and then perform structure-from-motion to register the two cameras and reconstruct sparse 3D structure. Specifically, we use the robust pose estimation algorithm described in Section 4.1. In brief, it uses the algorithm of Stewenius et al. [2007] to compute relative pose given SIFT [Lowe 2004] correspondences between the two views within a robust sampling loop similar to RANSAC [Fischler and Bolles 1981]. Then, given the projection matrices of the two cameras, we reconstruct the 3D coordinates of each correspondence using triangulation [Hartley and Zisserman 2000]. These 3D points are then projected into the second view, and displayed to the user alongside the reference photograph; the user is asked to click 6–8 correspondences. These correspondences are used to register the reference camera in the next step.

### 3.2 Reference Camera Registration

We next relate the reference image to the reconstructed scene from the first two photographs taken by the user, given matches between the reference and the second view. For this, we infer the intrinsic and extrinsic parameters of the reference camera using Levenberg-Marquardt optimization (specifically, Lourakis's LM package [2004]), minimizing the sum of squared projection errors of the matched points. We assume zero skew and optimize nine degrees of freedom: one for focal length, two for the principal point, three for rotation, and three for translation. We initialize the rotation matrix to the identity matrix, the translation matrix to zero, and the focal length to the focal length of the current camera. We initialize the principal point by analyzing the vanishing points as described in Section 3.2.1.

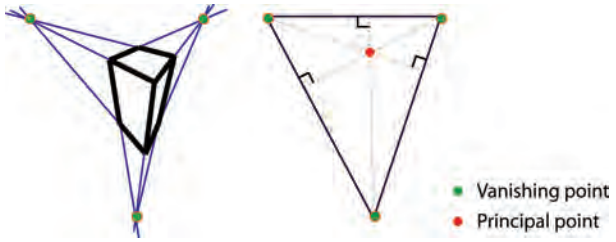


Fig. 5. Under perspective projection, parallel lines in space appear to meet at their vanishing point in the image plane. Given the vanishing points of three orthogonal directions, the principal point is located at the orthocenter of the triangle with vertices at the vanishing points.

Although this initialization is not close to the ground truth, we observe that the Levenberg-Marquardt algorithm converges to the correct answer since we allow only 9 degrees of freedom and the rotation matrix tends to be close to the identity matrix for rephotography.

3.2.1 *Principal Point Estimation.* The principal point is the intersection of the optical axis with the image plane. If a shift movement is applied to the lens to make the verticals parallel or if the image is cropped, the principal point is not in the center of the image, and it must be computed. The analysis of vanishing points provides strong cues for inferring the location of the principal point. Under perspective projection, parallel lines in space appear to meet at a single point in the image plane; this point is the vanishing point of the lines. Given the vanishing points of three orthogonal directions, the principal point is located at the orthocenter of the triangle whose vertices are the vanishing points [Hartley and Zisserman 2000], as shown in Figure 5.

We ask the users to click on three parallel lines in the same direction; although two parallel lines are enough for computation, we ask for three to improve robustness. We compute the intersections of the parallel lines. We locate each vanishing point at the weighted average of three intersections. The weight is proportional to the angle between two lines [Caprile and Torre 1990], since the location of the vanishing point becomes less reliable at smaller angles. We discard the vanishing point when the sum of the three angles is less than 5 degrees.

During Levenberg-Marquardt nonlinear optimization, we initialize and constrain the principal point as the orthocenter, given three finite vanishing points. If we have one finite and two infinite vanishing points, we initialize and constrain the principal point as the finite vanishing point. With two finite vanishing points, we constrain the principal point to be on the vanishing line that connects the finite vanishing points.

In summary, the result of the preceding methods is a 3D reconstruction of the scene from the first and second frames, as well as a calibration of the reference view and its relative pose from the first view. This information is then used in the next stage, which guides the user to the viewpoint of the reference image.

## 4. REAL-TIME USER GUIDANCE

Our rephotography tool provides the user with real-time guidance towards the reference viewpoint. To do so, we compute relative

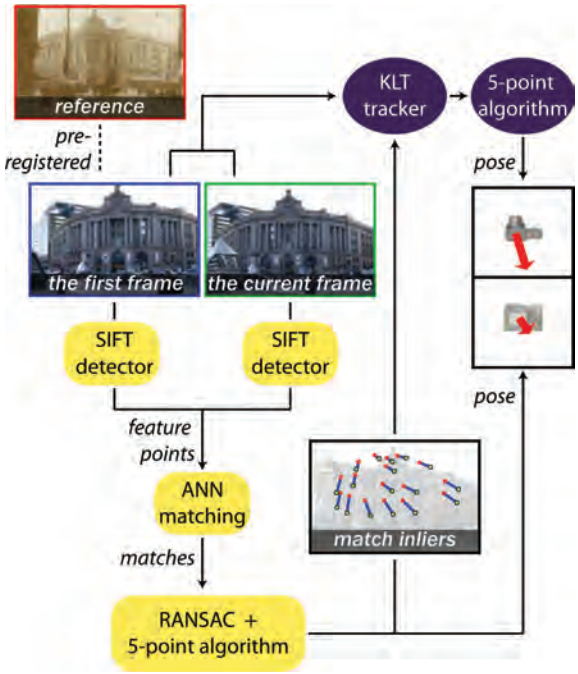


Fig. 6. Overview of our real-time guidance approach. In this stage, the reference camera is already registered to the first view. In our main process, we use an interleaved strategy where a lightweight estimation is refreshed periodically by a robust estimation to achieve both real-time performance and robustness. Yellow rounded rectangles represent robust estimation and purple ellipses are for lightweight estimation. The robust estimation passes match inliers to the lightweight estimation at the end.

pose between the current view and the reference view and visualize the needed 3D translation to the user in the form of two arrows, as shown in Figure 9. We also show the current view to the user after a best-fit rotation alignment between the current view and the reference view is applied. This rotational stabilization allows the user to focus on 3D translation and avoid worrying about precisely rotating the camera. To achieve real-time performance, we interleave a robust but slower relative-pose computation with a faster, lightweight updating scheme. A diagram of our real-time guidance approach is shown in Figure 6.

## 4.1 Robust Camera Pose Estimation

In our robust estimation process, we estimate the camera pose relative to the first frame instead of the reference in order to avoid degeneracy in the estimation when the user approaches the desired viewpoint. Since we know the reference camera location relative to the first frame $[R_{10}|T_{10}]$, we can derive the relative pose between the current and reference photographs. $[R_{ij}|T_{ij}]$ is the $j$th camera location relative to the $i$th camera location. $R$ is its rotational component, and $T$ is its translational component. For each frame $n$, we compute the current camera location relative to the first camera location $[R_{1n}|T_{1n}]$. The translational component $T_{0n}$ of the current camera location relative to the reference camera is

$$T_{0n} = T_{1n} - R_{1n} * R_{10}^{\top} * T_{10}. \quad (1)$$

In our full pipeline, we interleave this robust estimation with a lightweight estimation. We present details in Section 4.2.

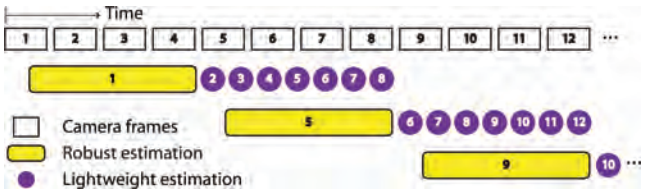


Fig. 7. Our interleaved pipeline. Robust estimation and lightweight estimation are interleaved using three threads: one receives the current frame from the camera, the other finds inliers, and another tracks the inliers. The robust estimation updates a set of inliers to be tracked. Numbers in this figure indicate the camera frame numbers. Note that for simplicity, this figure shows fewer frames processed per robust estimation.

4.1.1 *Correspondence Estimation.* To find matches between the first and current frames, we use SIFT [Lowe 2004] feature points. SIFT is designed to be invariant to scale changes and linear brightness changes. It is also partially invariant to viewpoint changes. For speed, we use a GPU implementation [Sinha et al. 2006]. Input images have around one megapixel and we downsample the images by two for speed. For the downsampled images, SIFT detects around one thousand feature points. We use an approximate searching method, ANN [Arya et al. 1998] to find correspondences. We set the threshold of the second ratio test [Lowe 2004] quite strictly at 0.6 to keep only trustworthy correspondences.

4.1.2 *Essential Matrix Estimation.* We compute relative camera pose between the first view and the current frame because the baseline is wide, which avoids a motion degeneracy when the user reaches the goal. Since the user's camera is already calibrated, we only need to estimate the essential matrix that relates the calibrated images. We use Stewénius's five-point algorithm [2007] which estimates the essential matrix between two calibrated cameras in real time. We run MSAC (m-estimator sample consensus) [Torr and Zisserman 2000] to find inliers and the best-fitting essential matrix. MSAC is similar to RANSAC, but it modifies the cost function so that outliers are given a fixed penalty while inliers are scored on how well they fit the data. The accuracy of MSAC is close to MLESAC (maximum likelihood consensus) without the loss of speed [Torr and Zisserman 2000]. We fix the number of iterations at 1000. We determine inliers and the best-fitting essential matrix using the symmetric epipolar distance [Hartley and Zisserman 2000]. Our threshold is 0.01 in normalized point coordinates.

## 4.2 Real-Time Camera Pose Estimation

We want to provide robust results but also interact with users in real time. Our robust estimation generates reliable results but its computation is expensive and takes seconds. To provide real-time feedback, we interleave our robust estimation with a lightweight estimation which is not as robust but inexpensive. In our lightweight estimation, we do not update the correspondences but track the most recent set of inliers using feature tracking and recompute the relative camera pose in one iteration.

We use Birchfield's KLT implementation [2007] to track feature points. It performs an affine consistency check [Shi and Tomasi 1994] and performs a multiscale tracking that refines the feature point locations from coarse to fine resolution.

Our robust estimation and lightweight estimation are interleaved as shown in Figure 7. Robust estimation detects feature points, finds matches, and estimates a new set of inliers and an epipolar geometry using robust statistics. This takes around two seconds while our lightweight estimation runs at more than 10 frames per

0. Register the reference camera
1. Robust estimation starts. Estimate correspondences.
2. Estimate camera pose.
3. Estimate the scale of the translation.
4. Check if the robust estimation result passes sanity testing. If yes, proceed to the next step. Otherwise repeat from Step 1.
5. Visualize the direction to move. The robust estimation ends.
6. Multi-threading starts. Thread A repeats robust estimation from Step 1, while Thread B performs a lightweight estimation.
7. Thread B tracks inliers found in Step 2 and estimates camera pose using only one iteration.
8. Estimate the scale of the translation.
9. Check if the lightweight estimation result passes sanity testing. If yes, proceed to the next step. Otherwise repeat from Step 7.
10. Visualize the direction to move.
11. Repeat from Step 7 until Thread A finishes Step 5 and updates the set of inliers.

Fig. 8. The flow chart of our interleaved scheme.

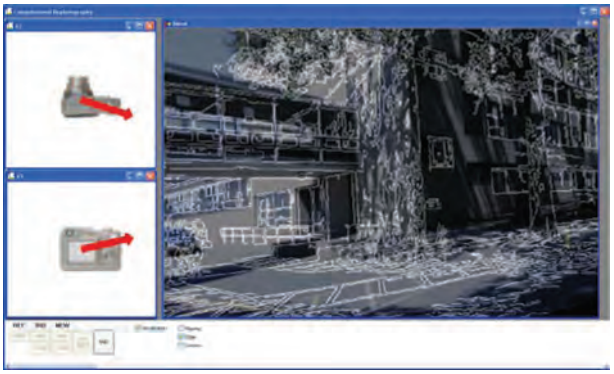

Fig. 9. A screen capture of our visualization, including our primary visualization of two 2D arrows, as well as an edge visualization. The upper left view shows the suggested motion direction from the top while the lower left view is perpendicular to the optical axis. The edge visualization shows a linear blend of the edges of the reference image and the current scene after rotation stabilization. The alignment of the edges can be used to evaluate whether the user has reached the desired viewpoint.

second. This interleaved process allows the accuracy of the inliers to be preserved and provides users with a real-time update.

## 4.3 Interleaved Scheme

Our interleaved pipeline is implemented as three threads: one communicates with the camera, the other conducts robust estimation, and another performs lightweight estimation. At the end of each robust estimation, a set of inliers is passed to the lightweight estimation thread. We store subsequent frames of the key frame on which the robust estimation computes inliers. When the light estimation is refreshed with an inlier set from the robust estimation, it starts tracking from the next frame of the key frame instead of the current camera frame. Since the lightweight estimation uses optical flow to track points, there should not be a large gap between the key frame where the inliers are computed and the first frame when tracking starts. When the inlier set is refreshed with a new robust estimation result users can observe a one-second delay. However,

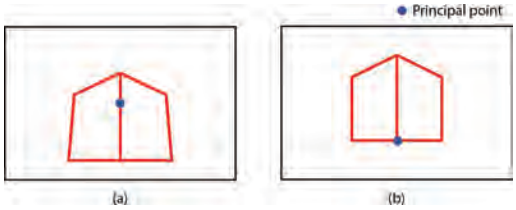


Fig. 10. The synthetic cube images we used to test the accuracy of our estimation of principal point and camera pose. The left cube image (a) has its principal point at the image center, while the right cube image (b) has had its principal point moved to the image bottom.

this is negligible compared to the whole rephotography process, and it does not affect the user performance or resulting rephotograph quality. Our interleaved version operates as in Figure 8.

4.3.1 *Sanity Testing.* For each resulting pose, we perform three sanity tests to make sure our visualization is reliable. We compare the 3D structure reconstructed from each frame with our initial 3D reconstruction from the first two images. We measure the 3D error of all points and ignore the pose estimation if the median of the 3D error is more than 10%. Typically, the median error is less than 5%.

In addition, we check if the current camera pose result is consistent with previous ones. We found that a simple filter works, although the Kalman filter [Kalman 1960] would likely generate a good result as well. We measure the mean and the standard deviation of the camera locations at the previous ten frames and confirm that the current estimated camera location is within 4 standard deviations from the mean. We assume the camera motion is smooth and the pose variation is small. The preceding two tests typically detect a wrong answer roughly once in 100 frames.

Finally, we test for a structure degeneracy caused when all the inliers come from one single plane in the scene. We find the best-fitting homography using RANSAC with 1.5 pixel average mapping errors within 500 iterations. If the number of homography inliers is more than 70% of the epipolar geometry inliers, we ignore the pose estimation result. Since we use a large-enough baseline, this error does not occur frequently.

When our estimation result fails to pass the previous tests, we simply do not update the visualization. Since wrong answers do not occur often, this does not affect the user experience significantly.

## 4.4 Scale Estimation

After relative pose is computed, a problem remains: the scale of the translation between the current frame and the first frame is ambiguous. We therefore scale it to maintain consistency between iterations. In the initial calibration step, we reconstructed a 3D structure between the first and second frames using triangulation. In a subsequent iteration $n$, we reconstruct 3D structure between the first and $n$th frames. The scale between these two reconstructions should be different by a constant factor. We can make the scales consistent by estimating the scale factor that causes the distance between the first camera and the 3D scene to be equivalent between the two reconstructions. To do so, we place the first camera at the origin for both reconstructions. We then compute the median ratio of distance to the origin for each 3D point in the first reconstruction and the $n$th reconstruction. Finally, we multiply the length of the translation vector by this ratio, which makes the length of our arrow visualization meaningful and consistent across frames.

Table I. Analysis of the Robustness of Our Estimation against Simulated User Input Error

| | Cube (a) | Cube (b) |
|---|---|---|
| Viewpoint error | 0.001 (0.02% of the camera distance) | 0.016 (0.25% of the camera distance) |
| Principal point error | 0.2 pixels (0.05% of the image size) | 1.8 pixels (0.4% of the image size) |

Small errors show that our principal point estimation is robust.

Table II. Our Principal Point Constraint with Vanishing Point Estimation Enables an Accurate Estimation of the Viewpoint in Comparison with Levenberg-Marquardt Nonlinear Optimization Alone

| | Principal Point Error | | Viewpoint Error | |
|---|---|---|---|---|
| | Cube (a) | Cube (b) | Cube (a) | Cube (b) |
| WITH our principal point constraint | 17 pixels | 13 pixels | 0.26 | 0.24 |
| | $<4\%$ of the image size | | 3% of the camera distance | |
| NO principal point constraint | 153 pixels | 126 pixels | 3.75 | 3.80 |
| | $>30\%$ of the image size | | $>50\%$ of the camera distance | |

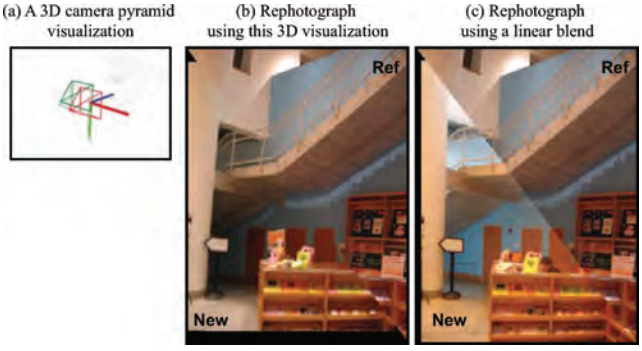


Fig. 11. User study results with our first visualization. We displayed the relative camera pose using 3D camera pyramids (a). The red pyramid showed the reference camera, and the green one was for the current camera. Although the rephotograph using this 3D visualization (b) was better than that using a linear blend (c), neither helped users to take accurate rephotographs.

## 4.5 Rotation Stabilization

We also use the result of relative pose estimation to rotationally stabilize the current frame before displaying it. Since users find it challenging to simultaneously follow instructions suggesting both translational and rotational motions, we instead only communicate translation to the user. We automatically compute the best camera rotation between the current and reference views, and apply this rotation as a warp before displaying the current frame. This rotation alignment allows the user to focus on translating the camera in the right direction without striving to hold the camera in the right orientation.

The effect of a 3D camera rotation and zoom can be described with an infinite homography [Hartley and Zisserman 2000]. The infinite homography is a subclass of the general homography, as it is restricted to rigid camera rotations and zooms. We use the algorithm of Brown et al. [2007] to compute the infinite homography that fits all the epipolar geometry inliers with the least squared error.

Table III. User Study Error

| | Our Method | Linear Blend | Remark |
|---|---|---|---|
| avg. (m) | (a) 0.47 | (b) 6.03 | (a)/(b) = 7.8% |
| std. (m) | 0.14 | 2.83 | |
| P-value | 0.02 | | |

The P-value is smaller than 0.05. This means that this result is statistically meaningful and significant. With our method, users made less than 8% of error than with a linear blend.

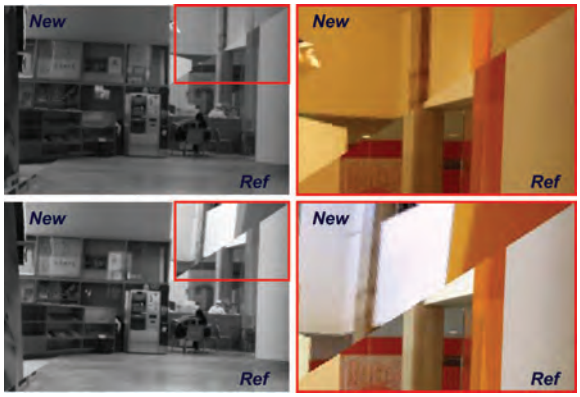


Fig. 12. User study results in the indoor scenes. Split comparison between the reference image and users' rephotographs after homography warping. The result at the top is from a user using our method, and the one at the bottom is from a user using a linear blend visualization. Though neither result is completely successful, notice that result at top is aligned much better.

Fig. 13. In the final user study, the reference photographs had aged appearances.

## 5. VISUALIZATION

Comparing the reference and current image side by side does not provide precise information about viewpoint difference. In our pilot user study, we provided a linear blend of the reference and current image, and users could not estimate the desired viewpoint by examining the pixel difference. In a subsequent test, we showed the relative pose information in 3D (see Figure 11(a)). Still we found that it was hard for users to interpret 3D information. In our final visualization design, we visualize the relative camera pose in two 2D planes: one is the direction seen from the top view and the other is perpendicular to the optical axis, as shown in Figure 9. In our final user studies, users found our arrow visualization easy to learn and follow.

In addition, we visualize the alignment between the reference and current views to help users refine and confirm the final viewpoint. Because of the potentially large appearance differences between historical and current images, we found that a linear blend of the old reference photograph and the current scene to be confusing. We experimented with three alternate visualizations: an edge

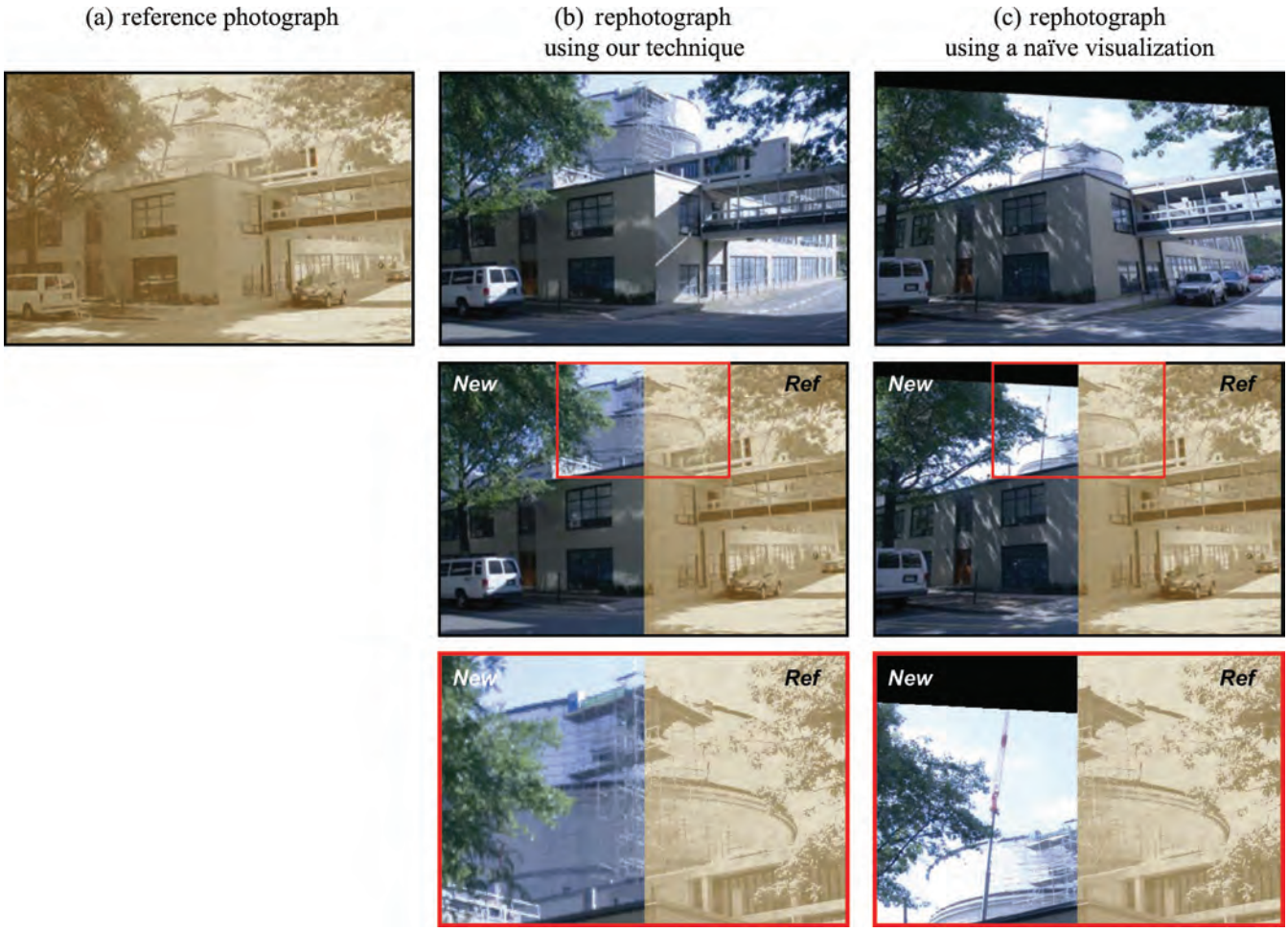


Fig. 14. User study results. Left to right: (a) the reference photograph and users' rephotographs using our technique (b) and a naïve visualization (c) after homography warping. The second row shows a split comparison between the reference image and users' rephotographs, and the third row shows a zoomed-in inset. With our method, users took more accurate rephotographs.

visualization, a flipping visualization, and a visualization with a reference camera projected onto the current frame.

In the edge visualization, we overlay the edges extracted from the reference image over the current frame. In the flipping visualization, users can quickly flip between the reference photograph and the current frame. In both the edge and flipping visualizations, we warp the current frame using a best-fitting infinite homography (Section 4.5), which is not particularly useful when the user is far from the desired viewpoint because the current and reference views cannot be related by the infinite homography, but very useful for small translations.

As the translational component becomes zero, the rotational component is well resolved by a homography. Finally, for the third visualization we simply project a 3D model of the reference camera onto the current frame, to provide a visual target for the user.

During user studies, we let users choose among three visualizations. All the users chose the edge visualization as their preferred visual feedback. Users used the flipping visualization only at the final viewpoint to confirm the viewpoint. Users did not find the projected reference camera very useful. Figure 9 shows our final visualization design.

## 6. RESULTS

In our prototype, we estimate relative pose using the output we get from the camera viewfinder. We use a Canon 1D Mark III live view, which outputs 5–10 frames per second. Each robust estimation takes about 2 seconds on a 2.4 GHz laptop with NVIDIA GeForce 8600M GT, while a lightweight estimation tracks inliers, estimates the relative pose, and visualizes the arrows at 10–20 frames per second. With multithreading, GPU-SIFT takes one second and the Approximate Nearest Neighbor (ANN) takes one second.

### 6.1 Preregistration Evaluation

We analyze the accuracy of our estimation of the principal point and camera pose using two synthetic images. We first evaluate the robustness of our estimation to user input error. Figure 10 shows our synthetic test cases: cube (a) has its principal point at the image center, and cube (b) has its principal point moved to the image bottom. The cube size is $3\times3\times3$, and the distance between the cube and the camera is around 6. The input image size is $512\times340$.

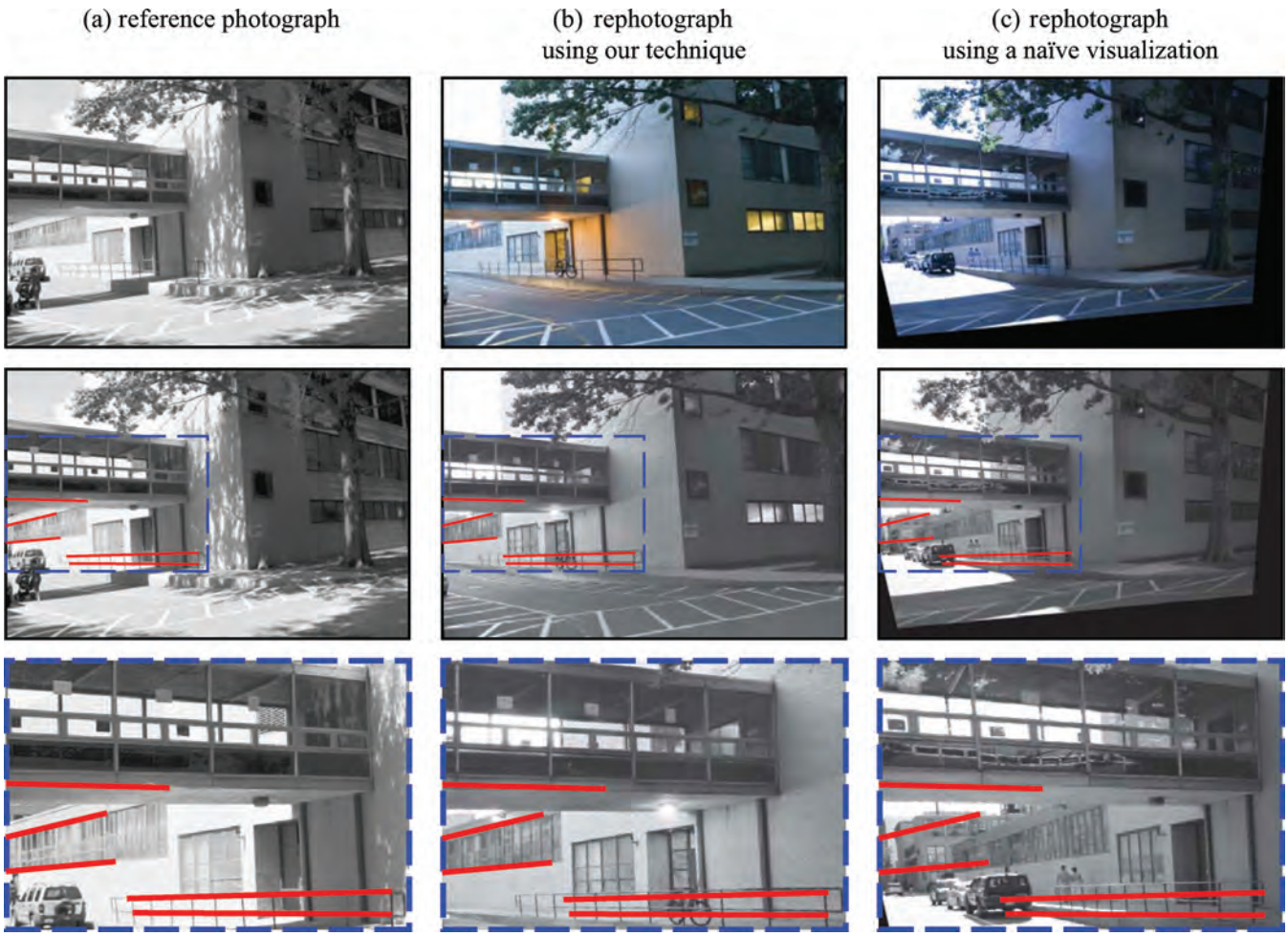


Fig. 15. User study results. Left to right: (a) the reference photograph and users' rephotographs using our technique (b) and a naïve visualization (c) after homography warping. The last row shows a zoomed-in inset blend of rephotographs and several features from (a) highlighted in red; these highlights match image features in (b) better than in (c). This shows that users take more accurate rephotographs using our technique than using a naïve visualization.

Table IV. User Study Error

| | Our Method | Linear Blend | Remark |
|---|---|---|---|
| avg. (m) | (a) 1.8 | (b) 4.4 | (b)/(a) = 2.5 |
| std. (m) | 0.6 | 2.9 | |

The error with a linear blend is 2.5 times larger than the error with our method.

For the first test, we simulate user errors by randomly adding or subtracting up to 2 pixels to the ground-truth input positions for principal point estimation and pose estimation. The inputs to our principal point estimation are 18 points for three parallel lines in the three orthogonal directions. The inputs to pose estimation are 6 points. We estimate the principal point and pose 100 times and record the average error. Table I shows the result. The average viewpoint errors are 0.001 for cube (a) and 0.016 for cube (b), which are 0.02% and 0.25% of the camera distance. The average principal point errors are 0.2 pixels and 1.8 pixels, respectively, which are 0.05% and 0.4% of the image size. This shows that our principal point estimation is robust against user input error.

For the second test, we add errors to the 3D coordinates used for the nonlinear optimization in addition to the user input error. We compare two cases: (1) the principal point is constrained by our estimation method using vanishing points, and (2) the principal point is estimated relying solely on Levenberg-Marquardt nonlinear optimization. Table II shows the results. With our vanishing point estimation, the average errors of the estimated principal points are 17 pixels for cube (a) and 13 pixels for cube (b), and the average viewpoint errors are 0.26 and 0.24. These numbers are 4% and 3% of the image size, and 3% of the camera distance. In contrast, if we only rely on Levenberg-Marquardt nonlinear optimization to estimate the principal point and the viewpoint, the principal point errors are 153 pixels and 126 pixels on average, respectively. These errors are 36% and 30% of the image size, more than 9 times larger than the errors using vanishing points. The average viewpoint errors are 3.75 and 3.8 respectively, which are almost 50% of the camera distance. Levenberg-Marquardt nonlinear optimization is a local descent approach and relies on good initialization. When significant measurement noise is present in the initialization, it might converge to a wrong local minimum. In addition, the projection errors are not


Fig. 16. Results. Left to right: the reference images, our rephotograph results, and professional manual rephotographs without our method.

discriminative enough to determine the viewpoint and the principal point at the same time. There exist ambiguities between changing the principal point and moving the camera. This is reduced by the vanishing point method.

Finally we analyze the effect of varying the focal length while changing the camera distance. As a result, the size of the projected cube stays the same, but camera rotation and principal point modification become harder to disambiguate. The focal lengths used are 400, 600, 800, and 1000. 400 is equivalent to 20mm, and 1000 is equivalent to 50mm for a 35mm film. The errors increase as the focal length and the camera distance increase. The principal point errors are 13, 27, 45, and 66 pixels respectively, which are 3%, 6%, 11%, and 15% of the image size. The viewpoint errors are 0.4, 0.6, 1.15, and 1.86, which are 5%, 5%, 7%, and 9% of the camera distance. The more we zoom, the more ambiguous the estimation becomes. This is related to the fact that the projection error is less discriminative for a photograph taken by a telephoto lens, because the effect of a 3D rotation and that of a 2D translation become similar.

## 6.2 User Interface Evaluation

We performed multiple pilot user studies before finalizing the design of our user interface. The studies included eight females and eight males with ages ranging from 22–35. Only one of them participated in multiple studies. We recruited the participants via personal contacts; eight of them had computer science backgrounds, while the other eight did not.

6.2.1 *First Pilot User Study.* In our first pilot user study, we wanted to test whether humans would be able to estimate the viewpoint differences by simply comparing two photographs.

*Procedure.* We asked users to estimate the viewpoint of a reference photograph by comparing it with the output of the camera viewfinder, while they moved the camera. We provided two users with three different visualization techniques: the reference and current image side by side, a linear blend of the reference and current image, and a color-coded linear blend of the reference in red and current image in blue. We asked the users questions upon completion of the task.

*Results and conclusions.* Comparing the reference and current image side by side did not seem to provide information about viewpoint differences; both the user's final rephotographs were poor. Although users preferred the linear blend among three visualization, the users could still not estimate the desired viewpoint by examining parallax. This leads to our first visualization design.

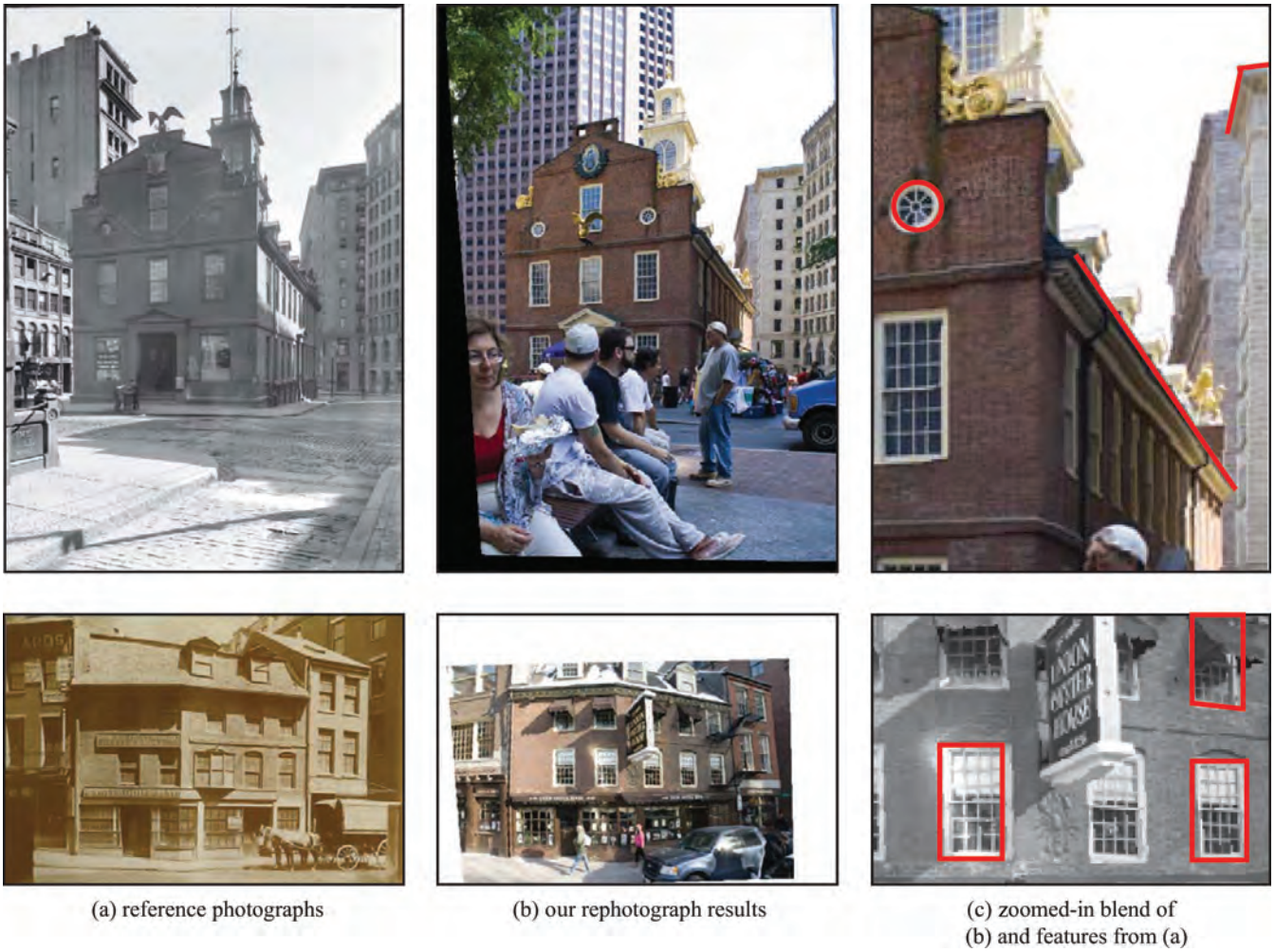


Fig. 17. Results. Left to right: the reference images, our rephotograph results, and a zoomed-in inset showing highlighted features from (a) superimposed on (b); their alignment is evidence of good rephotographs.

6.2.2 *Second Pilot User Study.* In our first visualization design, we showed the relative pose information in 3D and updated the camera pyramid every 3 seconds (see Figure 11(a)). In a pilot user study, we wanted to test whether users would be able to take more accurate rephotographs using our 3D pose visualization than using a linear blend visualization.

In addition to showing 3D camera pyramids, we provided the user with her distance to the desired viewpoint with respect to the scene distance. Among the reconstructed scene points, we picked one point roughly in the direction the user needed to move and told her how far she was from the desired viewpoint relative to the scene point. We asked users whether this additional information was useful upon completion of the task.

*Procedure.* Given the reference photograph taken by the same camera, we asked six users to reach the viewpoint where the reference photograph was taken. Note that this task was easier than typical rephotography: there was not a large appearance difference, and users did not need to estimate the focal length. We tested 4 indoor scenes. Each participant experienced all the scenes and both techniques, but each scene was paired with only a single rephotography technique for that participant. We measured the accuracy of rephotographs by comparing the pixel differences between the reference and resulting rephotographs.

*Results and conclusions.* We observed that neither the 3D pyramid visualization nor a linear blend visualization helped users to take accurate rephotographs. Figure 11 shows the resulting rephotographs.

In terms of the pixel differences, users made less error with this 3D visualization, specifically 70% of the error with a linear blend. However, we realized that comparing pixel differences was not a good metric. We decided to measure the distance from the users' final camera location to the ground truth in the next studies.

In the camera pyramid visualization, users found it hard to interpret 3D information; most had a hard time separating translation and rotation. Besides, users did not find the distance indicator useful. In general, users preferred a simple visualization; having one visualization window helped users focus on the task. Users


Fig. 18. Results. Left to right: the reference photographs, our rephotograph results, split comparisons between the references and our rephotographs.

became tired and lost when they had to jump between different visualization windows. In addition, users asked for real-time feedback.

6.2.3 *First Final User Study.* In our final estimation and visualization, we compute relative camera pose in real time and show the direction to move using two 2D arrows (See Figure 9). We conducted two additional user studies to validate our technique.

In the first final user study, we wanted to compare our arrow visualization technique with a linear blend visualization. In addition, we included the focal length estimation: users had to manually estimate the zoom setting of the lens using the linear blend estimation, while our visualization automatically resolved the focal length difference. We evaluated accuracy as measured by the distance between the reference and new cameras.

*Procedure.* Given the reference photograph taken by a different camera, we asked four users to reach the viewpoint where the reference photograph was taken, with a 3 minute time limit. We tested two indoor scenes. Each participant experienced both scenes and techniques, but each scene was paired with only a single

Fig. 19. Results with style transfer. Left to right: the reference photographs, our rephotograph results, and our rephotographs with styles transferred from the reference photographs.

rephotography technique for that participant. We marked the reference camera location on the map and measured the distance from the users' final camera location to the ground truth. We did not ask users to choose the first and second viewpoints; they were fixed among all the users.

*Results and conclusions.* Table III shows the average distance between the ground truth and the final locations where four users took the rephotographs for two test cases. The error with our method was less than 8% of the error with a linear blend. Users found that our 2D arrows were easy to learn and follow. Figure 12 compares two rephotograph results using both techniques. In every test case, users took more accurate rephotographs with our arrow visualization than with a linear blend visualization.

6.2.4 *Second Final User Study.* In our final user study, we wanted to test our user interaction schemes including providing a wide-baseline, clicking on matches, and comparing two photographs with large appearance differences. We compared the accuracies of the resulting rephotographs using our technique against those with a naïve visualization. In particular, we sought to compare the accuracy of the viewpoint localization.

*Procedure.* We compared our technique with a naïve visualization and used reference images for which the ground-truth location

is known. To make the scenario more realistic, we simulated an aged appearance on the reference photographs that we captured: we transferred the tonal aspects from old photograph samples to the reference photographs [Bae et al. 2006], as shown in Figure 13. As a result, the reference photographs had large appearance differences from current photographs. We asked six users to reach the viewpoint where the reference photograph was taken within 10 mins. We tested three outdoor scenes. Each participant experienced all the scenes and both techniques, but each scene paired with only a single technique for that participant.

For both methods, all users started from the same initial location. With our technique, we only fixed the first viewpoint (as the initial location), and asked users to choose the second viewpoint. In addition, users provided correspondences between the reference and the second frame by clicking six matches. In the naïve visualization method, we showed both linear blend and side-by-side visualizations of the reference and current frame, since a linear blend suffered from large appearance differences between the reference and current photographs. Again, users had to manually estimate the zoom setting of the lens using the naïve visualization, while our visualization automatically resolved the focal length difference. Before each user study, we provided users with a quick tutorial of both methods.

*Results and conclusions.* In every test case, users took more accurate rephotographs with our arrow visualization than with the naïve visualizations. Figures 14 and 15 compare the rephotographs taken with our technique and those using a naïve visualization. The remaining parallax in the rephotograph results using a naïve visualization is quite large, while our technique allowed users to minimize parallax.

Table IV shows the average distance between the ground truth and the final locations where six users took the rephotographs for three test cases. The average error with our method is 40% of the average error with a linear blend. The distance difference became smaller than the indoor cases. In the indoor scenes, the parallax was subtle, but in the outdoor scenes, users could notice some important cues such as whether buildings were occluded or not. Still many people could not figure out how to move the camera to resolve the parallax. With a naïve blend, users had to estimate the location and focal length of the reference camera by themselves. With our method, users needed only to follow our arrow visualization while our technique automatically estimated the location and focal length of the reference camera.

### 6.3 Results on Historical Photographs

Figures 16, 17, 18, and 19 show our rephotograph results of historical photographs taken by unknown cameras. It usually took 15–30 minutes to reach the desired viewpoint. This task required more extensive time because we often had to walk 50–100m with a laptop and tripod, and cross busy roads. In Figure 19, we apply style transfer from the reference to the rephotographs [Bae et al. 2006]. By matching the tonal aspects, it becomes even more evident which scene elements are preserved and which have changed across time. Faithful rephotographs reveal the changes of roofs, windows, and the overall neighborhood.

### 6.4 Discussion

The bulk of the laptop currently limits portability and we hope that open-platform digital cameras with additional processing power will enable rephotography directly from the camera.

Our relative pose estimation works best when there is sufficient parallax between the images. When nearing the viewpoint, the user typically relies more on the alignment blend, which can limit final precision. Our technique requires a reasonable number of feature points (around 20), and can suffer in scenes with little texture. The scene must present enough 3D structure to make viewpoint estimation well-posed. If the scene is mostly planar, a homography can match any pair of views and the viewpoint cannot be inferred.

In addition, the resulting rephotograph's precision depends on the user's tolerance for error. For example, users typically focus on landmarks in the center of the image, and may not notice that features towards the periphery are not well-aligned. If the user only checks alignment in the center, parallax towards the periphery may not be resolved, as in Figures 15 and 17.

We share a number of limitations with traditional rephotography: if the desired viewpoint is not available or the scene is occluded and cannot be seen at the desired viewpoint, rephotography is impossible. Nevertheless, our technique can still help users realize that the viewpoint is no longer available.

Audio feedback is a natural extension to our visualization that we hope to explore in the future.

## 7. CONCLUSIONS

In this article we described a real-time pose estimation and visualization technique for guiding a user while performing rephotography. Our method includes an approach to calibrating the historical camera by taking several photographs of the same scene, a wide-baseline reconstruction that avoids relative pose estimation degeneracy when the user reaches the desired target, and a visualization technique that guides the user with two 2D arrows and a rotationally-stabilized current view.

We believe that our work points towards an exciting longer-term direction: embedding more computation in cameras to support more complex interaction at capture time than is offered by current commodity hardware. While our prototype requires a laptop connected to a camera, we hope that more open camera platforms are developed in the future that allow more experimentation in designing novel user interfaces that can run on the camera.

### ACKNOWLEDGMENTS

We thank the MIT Computer Graphics Group, Adobe's Advanced Technology Labs, the anonymous reviewers and prereviewers, and Noah Snavely for their comments. This work was supported by grants from Quanta, Shell, and Adobe. Thanks to the publishers of the rephotography books for their images.

### REFERENCES

ADAMS, A., GELFAND, N., AND PULLI, K. 2008. Viewfinder alignment. *Comput. Graph. Forum 27*, 2, 597–606.

ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., AND WU, A. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM 45*, 6, 891–923.

BAE, S., PARIS, S., AND DURAND, F. 2006. Two-Scale tone management for photographic look. *ACM Trans. Graph. 25*, 3, 637–645.

BASRI, R., RIVLIN, E., AND SHIMSHONI, I. 1999. Visual homing: Surfing on the epipoles. *Int. J. Comput. Vis. 33*, 2, 117–137.

BAUDISCH, P., TAN, D., STEEDLY, D., RUDOLPH, E., UYTTENDAELE, M., PAL, C., AND SZELISKI, R. 2005. Panoramic viewfinder: Providing a real-time preview to help users avoid flaws in panoramic pictures. In *Proceedings of*

*the 17th Australia Conference on Computer-Human Interaction (OZCHI)*. 1–10.

BIRCHFIELD, S. 2007. KLT: An implementation of the kanade-lucas-tomasi feature tracker. http://www.ces.clemson.edu/stb/klt/.

BOUGUET, J.-Y. 2007. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/.

BROWN, M., HARTLEY, R., AND NISTER, D. 2007. Minimal solutions for panoramic stitching. *Comput. Vis. Pattern Recog.*, 1–8.

CAPRILE, B. AND TORRE, V. 1990. Using vanishing points for camera calibration. *Int. J. Comput. Vis. 4*, 2, 127–139.

CHEN, S. E. AND WILLIAMS, L. 1993. View interpolation for image synthesis. In *SIGGRAPH'93*. ACM, New York, 279–288.

DAVISON, A., REID, I., MOLTON, N., AND STASSE, O. 2007. Monoslam: Real-time single camera slam. *IEEE Trans. Pattern Anal. Mach. Intel. 29*, 6, 1052–1067.

FAUGERAS, O. 1993. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA.

FISCHLER, M. A. AND BOLLES, R. C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM 24*. 6, 381–395.

GORE, A. 2006. *An Inconvenient Truth*. Paramount Classics.

HALL, F. C. 2002. Photo point monitoring handbook. Tech. rep. PNW-GTR-526, USDA Forest Service.

HARTLEY, R. I. 1992. Estimation of relative camera positions for uncalibrated cameras. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 579–587.

HARTLEY, R. I. AND ZISSERMAN, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press.

HEYDEN, A. AND SPARR, G. 1999. Reconstruction from calibrated cameras—A new proof of the Kruppa-Demazure theorem. *J. Math. Imag. Vis. 10*, 2, 123–142.

KALMAN, R. E. 1960. A new approach to linear filtering and prediction problems. *Trans. ASME – J. Basic Engin. 82*, Series D, 35–45.

KANG, S. B. AND SHUM, H.-Y. 2002. A review of image-based rendering techniques. In *Proceedings of the IEEE/SPIE Visual Communications and Image Processing Conference*. 2–13.

KLETT, M., MANCHESTER, E., VERBURG, J., BUSHAW, G., AND DINGUS, R. 1990. *Second View: The Rephotographic Survey Project*. University of New Mexico Press.

LEVERE, D., YOCHELSON, B., AND GOLDBERGER, P. 2004. *New York Changing: Revisiting Berenice Abbott's New York*. Princeton Architectural Press.

LOURAKIS, M.. 2004. levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. http://www.ics.forth.gr/∼lourakis/levmar/.

LOWE, D. 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis. 60*, 2, 91–110.

MCNULTY, E. 2002. *Boston Then and Now*. Thunder Bay Press.

POLLEFEYS, M., NISTÉR, D., FRAHM, J. M., AKBARZADEH, A., MORDOHAI, P., CLIPP, B., ENGELS, C., GALLUP, D., KIM, S. J., MERRELL, P., SALMI, C., SINHA, S., TALTON, B., WANG, L., YANG, Q., STEWÉNIUS, H., YANG, R., WELCH, G., AND TOWLES, H. 2008. Detailed real-time urban 3D reconstruction from video. *Int. J. Comput. Vis. 78*, 2-3, 143–167.

SAND, P. AND TELLER, S. 2004. Video matching. *ACM Trans. Graph. 23*, 3, 592–599.

SCHEUERING, M., REZK-SALAMA, C., BARFUFL, H., SCHNEIDER, A., AND GREINER, G. 2002. Augmented reality based on fast deformable 2D-3D registration for image-guided surgery. In *Proceedings of the SPIE (Medical Imaging: Visualization, Image-Guided Procedures, and Display)*. Vol. 4681. 436–445.

SHI, J. AND TOMASI, C. 1994. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 593–600.

SINHA, S. N., FRAHM, J.-M., POLLEFEYS, M., AND GENC, Y. 2006. GPU-based video feature tracking and matching. In *Proceedings of the Workshop on Edge Computing Using New Commodity Architectures (EDGE)*.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: Exploring photo collections in 3D. *ACM Trans. Graph. 25*, 3, 835–846.

STEWÉNIUS, H., ENGELS, C., AND NISTÉR, D. 2007. An efficient minimal solution for infinitesimal camera motion. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–8.

TAKACS, G., CHANDRASEKHAR, V., GELFAND, N., XIONG, Y., CHEN, W.-C., BISMPIGIANNIS, T., GRZESZCZUK, R., PULLI, K., AND GIROD, B. 2008. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceedings of the ACM International Conference on Multimedia Information Retrieval*. 427–434.

TORR, P., FITZGIBBON, A., AND ZISSERMAN, A. 1999. The problem of degeneracy in structure and motion recovery from uncalibrated images. *Int. J. Computer Vis. 32*, 1, 27–44.

TORR, P. H. S. AND ZISSERMAN, A. 2000. MLESAC: A new robust estimator with application to estimating image geometry. *Comput. Vis. Image Understand. 78*, 1, 138–156.

WAGNER, D., REITMAYR, G., MULLONI, A., DRUMMOND, T., AND SCHMALSTIEG, D. 2008. Pose tracking from natural features on mobile phones. In *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*. 125–134.

WERNER, T., HERSCH, R. D., AND HLAVAC, V. 1995. Rendering real-world objects using view interpolation. In *Proceedings of the International Conference on Computer Vision (ICCV)*. 957–962.

Received October 2009; rivised February 2010; accepted April 2010