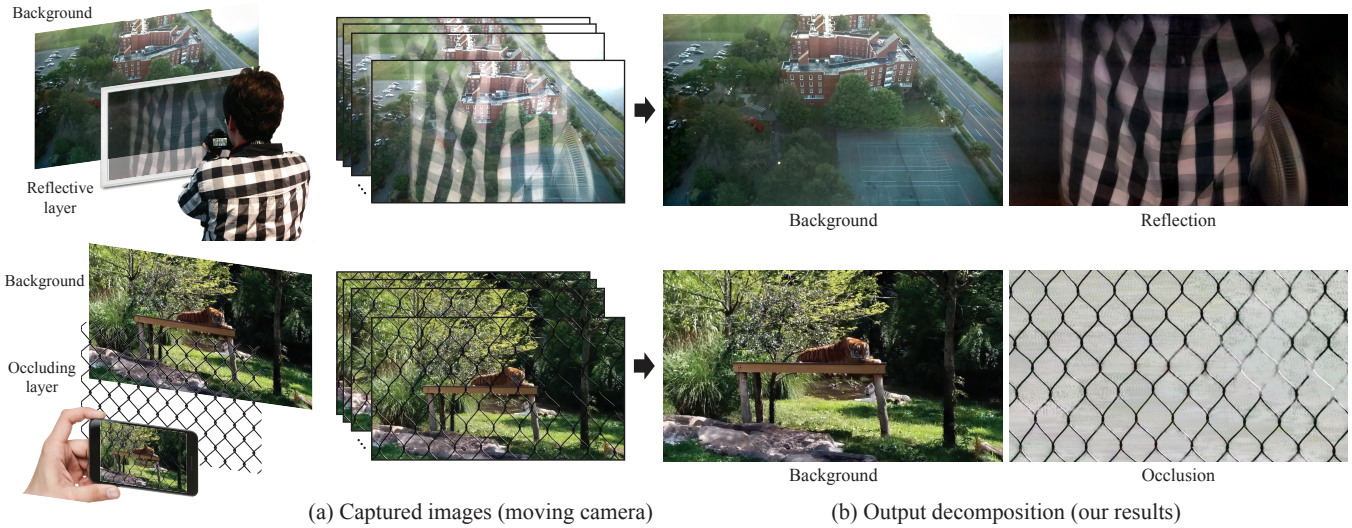
A Computational Approach for Obstruction-Free Photography

Tianfan Xue[1*] Michael Rubinstein[2†] Ce Liu[2†] William T. Freeman[1,2]

[1]MIT CSAIL [2]Google Research



(a) Captured images (moving camera) (b) Output decomposition (our results)

Figure 1: *In this paper we present an algorithm for taking pictures through reflective or occluding elements such as windows and fences. The input to our algorithm is a set of images taken by the user while slightly scanning the scene with a camera/phone (a), and the output is two images: a clean image of the (desired) background scene, and an image of the reflected or occluding content (b). Our algorithm is fully automatic, can run on mobile devices, and allows taking pictures through common visual obstacles, producing images as if they were not there. The full image sequences and a closer comparison between the input images and our results are available in the supplementary material.*

Abstract

We present a unified computational approach for taking photos through reflecting or occluding elements such as windows and fences. Rather than capturing a single image, we instruct the user to take a short image sequence while slightly moving the camera. Differences that often exist in the relative position of the background and the obstructing elements from the camera allow us to separate them based on their motions, and to recover the desired background scene as if the visual obstructions were not there. We show results on controlled experiments and many real and practical scenarios, including shooting through reflections, fences, and raindrop-covered windows.

CR Categories: I.3.7 [Image Processing and Computer Vision]: Digitization and Image Capture—Reflectance; I.4.3 [Image Processing and Computer Vision]: Enhancement

Keywords: reflection removal, occlusion removal, image and video decomposition

1 Introduction

Many imaging conditions are far from optimal, forcing us to take our photos through reflecting or occluding elements. For example, when taking pictures through glass windows, reflections from indoor objects can obstruct the outdoor scene we wish to capture (Figure 1, top row). Similarly, to take pictures of animals in the zoo, we may need to shoot through an enclosure or a fence (Figure 1, bottom row). Such visual obstructions are often impossible to avoid just by changing the camera position or the plane of focus, and state-of-the-art computational approaches are still not robust enough to remove such obstructions from images with ease. More professional solutions, such as polarized lenses (for reflection removal), which may alleviate some of those limitations, are not accessible to the everyday user.

In this paper, we present a robust algorithm that allows a user to take photos through obstructing layers such as windows and fences, producing images of the desired scene as if the obstructing elements were not there. Our algorithm only requires the users to generate some camera motion during the imaging process, while the rest of the processing is fully automatic.

We exploit the fact that reflecting or obstructing planes are usually situated in-between the camera and the main scene, and as a result, have different depth than the main scene we want to capture. Thus, instead of taking a single picture, we instruct the photographer to take a short image sequence while slightly moving the camera—an interaction similar to taking a panorama (with camera motion

∗ Part of this work was done while the author was an intern at Microsoft Research New England.

† Part of this work was done while the authors were at Microsoft Research.

perpendicular to the z-axis being more desired than rotation). Based on differences in the layers' motions due to visual parallax, our algorithm then integrates the space-time information and produces two images: an image of the background, and an image of the reflected or occluding content (Figure 1). Our setup imposes some constraints on the scene, such as being roughly static while capturing the images, but we find that many common shooting conditions fit it well.

The use of motion parallax for layer decomposition is not new. Rather, our paper's main contribution is in a more robust and reliable algorithm for motion estimation in the presence of obstructions. Its success comes from mainly: (i) a **pixel-wise flow field motion representation** for each layer, which, in contrast to many previous image decomposition algorithms that use parametric motion models (affine or homography), is able to handle depth variation as well as small motions within each layer; and (ii) an **"edge flow" method** that produces a robust initial estimation of the motion of each layer in the presence of visual obstructions, as edges are less affected by the blending of the two layers. Given an input image sequence, we first initialize our algorithm by estimating sparse motion fields on image edges. We then interpolate the sparse edge flows into dense motion fields, and iteratively refine and alternate between computing the motions and estimating the background and obstruction layers in a coarse-to-fine manner (Figure 3).

Importantly, we also show that the two types of obstructions—reflections and physical occlusions (such as fences)—can be handled by a single framework. Reflections and occlusions may appear different at a glance, and indeed, previous work have used different solutions to address each one. However, in this paper we present a unified approach to address the two problems from a single angle. With minimal tweaking, our system consists of largely shared modules for these two problems, while achieving results of higher quality than those produced by previous algorithms addressing either subproblem. In this paper we specify manually the type of obstruction present in the scene (reflective or occluding) to tweak the algorithm to each case.

We test our method in various natural and practical scenarios, such as shooting through fences, windows and other reflecting surfaces. For quantitative evaluation, instead of synthetically simulating obstructions by blending or composting images (as commonly done in previous work), we design controlled experiments in which we capture real scenes with ground truth decompositions. Our algorithm is fully automatic, can work with standard phone cameras, and only requires the user to move the camera in a freeform manner to scan the scene. In our experiments, we found that 5 images taken along a small, approximately horizontal baseline of a few centimeters are usually enough to remove the obstructing layer.

2 Background

The problems of removing reflections and removing occlusions from images have been explored in the past under different setups. Here we review related work in those two areas.

Reflection Removal. Separating transmission and reflection in images has been widely studied, both for the purpose of direct decomposition (e.g. [Levin et al. 2002; Szeliski et al. 2000]), as well as in the context of other graphics and vision applications such as image based rendering [Sinha et al. 2012; Kopf et al. 2013] and stereo [Tsin et al. 2006].

Previous work on reflection removal can be grouped into three main categories. In the first category are approaches that remove reflection from a single image. As this problem is highly ill-posed, researchers have proposed different priors to make the problem more constrained. For example, Levin et al. [2002] proposed to use image priors such as statistics of derivative filters and "corner detectors" in natural scenes to decompose the image. They later improved their algorithm using patch based priors learned from an external database [Levin et al. 2004; Levin and Weiss 2007]. However, their method requires a large amount of user input—relying on the user to mark points on the background and reflected content—and does not work well in textured regions. Recently, Li and Brown [2014] proposed to separate the reflection using a single image focused on the background, under the assumption that the reflection in that case will be blurrier. Even with these priors, single image reflection removal is extremely challenging and hard to make practical for real images.

The second line of work focuses on removing reflections from a set of images taken through polarizers. Using a polarized filter with different orientations, a sequence of images is captured, each of which is a linear combination of the background and reflection, $I^i = a^i I_B + b^i I_R$, where combination coefficients a^i and b^i depend on the direction of the polarized filters. This set of images is then used to decompose the background and reflection layers, again, using different priors on the two layers [Kong et al. 2014; Singh 2003; Sarel and Irani 2004; Farid and Adelson 1999]. These methods perform well, but the requirement of a polarized filter and two images from the same position limits their usefulness.

The third approach for removing reflections is to process an input image sequence where the background and reflection are moving differently. In this setup, both the intensity and the motion of each layer need to be recovered. To simplify the problem, most previous approaches in that category constrained the motion of each layer to follow some parametric model. Be et al. [2008] assumed translative motion, and proposed an algorithm to decompose the sequence using a parameterized joint diagonalization. Gai et al. [2009] assume that the motion of each layer follows an affine transformation, and found a new image prior based on joint patterns of both background and reflectance gradients. Guo et al. [2014] assume that the motion of each layers is a homography, and proposed a low-rank approximation formulation. For many practical scenarios, however, affine and perspective transformations cannot model well enough the motions of the background and reflectance layers. This is manifested as artifacts in the results.

Some authors used dense warp fields, as we do, to model the motions of the background and reflection. Szeliski et al. [2000] proposed a min/max alternation algorithm to recover the background and reflectance images, and used optical flow to recover a dense motion field for each layer. In addition to dense motion fields, our algorithm also incorporates image priors that were shown to be instrumental for removing reflections from image sequences [Gai et al. 2012], which are not used in [Szeliski et al. 2000]. Li and Brown [2013] extended that approach by replacing the optical flow with SIFT flow [Liu et al. 2008] to calculate the motion field. However, they model the reflection as an independent signal added to each frame, without utilizing the temporal consistency of the reflection layer, thus limiting the quality of their reconstructions.

Occlusion Removal. Occlusion removal is closely related to image and video inpainting [Criminisi et al. 2004; Bertalmio et al. 2000; Bertalmio et al. 2001; Newson et al. 2014]. To remove an object from an image or a video, the user first marks some regions to be removed, and then the inpainting algorithm removes those region and fills in the holes by propagating information from other parts of the image or from other frames in the sequence. Image and video inpainting is mainly focused on interactive object removal, while our focus is on automatic removal, as in most of the cases we address asking the user to mark all the obstructed pixels in the image would be too laborious and impractical.

Several other papers proposed to automatically remove visual obstructions of particular types from either an image or a video. Several algorithms were proposed to remove near-regular structures, like fences, from an image or a video [Hays et al. 2006; Park et al. 2008; Park et al. 2011; Yamashita et al. 2010]. Mu et al. [2012] proposed to detect fence patterns based on visual parallax, and to remove them by pulling the occluded content from other frames. [Barnum et al. 2010] proposed to detect snow and raindrops in frequency space. [Garg and Nayar 2004] remove rain drops based on their physical properties. All these work either focus on particular types of obstacles (e.g. raindrops), or rely on visual properties of the obstruction, such as being comprised of repeating patterns. Our goal is develop a general purpose algorithm that could handle common obstructions, without relying on their specific properties.

3 Problem Setup

Figure 2 shows our image formation model. A camera is imaging a scene through a visual obstruction, which can be either a reflective object, such as glass, or an opaque object, such as a fence. In order to remove the artifacts—either the obstruction itself, or the reflection introduced by the obstruction—the user captures a sequence of images while moving the camera. In this paper we assume that both the obstruction and background objects remain roughly static during the capture. If the obstruction is opaque (a fence, for example), we further require that each pixel in the background scene would be visible (not occluded by the obstruction) in at least one frame in the sequence, so that its information could be recovered. We also assume that the obstruction (or the reflected content) is not too close to the background, so that when moving the camera there would be sufficient difference between the motions of the two layers.

In this paper, we use a lower-case letter a to denote a scalar, a normal capital letter A to denote a vector, and a bold capital letter $\mathbf{A}$ to denote a matrix. We denote the matrix product as $\mathbf{A}B$, where $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^m$, and the element-wise product of two vectors as $A \circ B$, where $A, B \in \mathbb{R}^n$.

If there is no obstruction, we will get a clean image of the background, denoted as $I_B \in \mathbb{R}^n$ (n is the number of pixels). Now, due to the presence of obstructions, the captured image $I \in \mathbb{R}^n$ is a composition of the background image I_B and an additional, obstruction layer, I_O:

$$I = (\mathbf{1} - A) \circ I_O + A \circ I_B, \tag{1}$$

where $I_O \in \mathbb{R}^n$ is the obstruction layer we want to remove, $\mathbf{1} \in \mathbb{R}^n$ is a vector with all components equal 1, and $A \in \mathbb{R}^n$ is an alpha blending mask, which assigns a blending factor to each pixel. Notice that A multiplies (element-wise) the background image, not the foreground image (that is, $A = 1$ means a background pixel). This is a less conventional notation for alpha maps, but it will help simplify some of the math later on.

More specifically, if we are imaging through a reflective object, such as a clean window, the obstruction layer I_O is the image of objects on the same side of the camera, as shown in Figure 2(a). In this case we assume the alpha blending mask A is a constant, as reflective objects are usually homogeneous (i.e. glass as found in most windows typically reflect light similarly throughout it). This is a common assumption in the reflection separation literature [Szeliski et al. 2000; Li and Brown 2013; Guo et al. 2014]. If, on the other hand, we are imaging through a fence or other opaque objects, the obstruction layer I_O is the opaque object itself, as shown in Figure 2(b). In that case, the alpha map, A, equals 1 at the region where the background is not occluded, and is between 0 and 1 if the background is partially or fully occluded.

Decomposing I into the obstruction layer I_O and the background layer I_B is ill-posed from a single input image. We therefore ask the user to move the camera and take a sequence of images. Assuming the obstruction layer I_O is relatively closer (further away) to the camera than the background objects, its projected motion on the image plane will be larger (smaller) than the background objects due to the visual parallax. We utilize this difference in the motions to decompose the input image to the background and obstruction components.

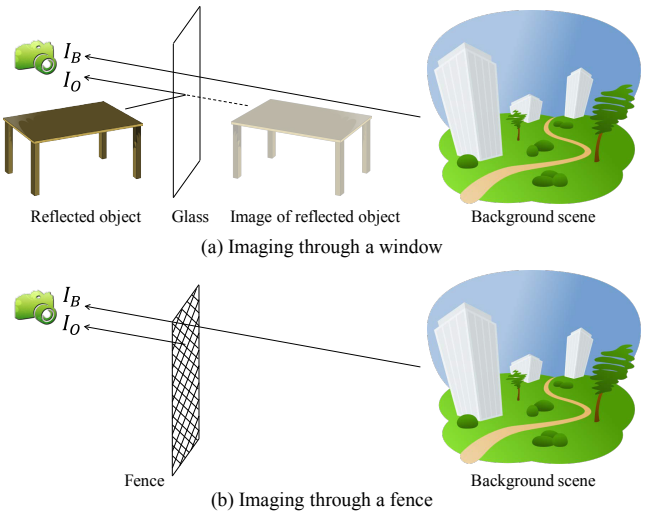


Figure 2: *The image formation model. A camera image of a desired scene though (a) a reflecting surface, and through (b) a partial obstruction.*

More formally, given an input sequence, we pick one frame t_0 from the sequence as the reference frame, and estimate the background component I_B and the obstruction component I_O of that frame, using the information from other frames. Assuming both the background objects and the obstruction layer are static, we can express the background and obstruction components of other frames $t \neq t_0$ as a warped version of the respective components of the reference frame t_0. Specifically, let V_O^t and V_B^t denote the motion fields for the obstruction and background layers from the reference frame t_0 to the frame t, respectively. The observed image at time t is then

$$I^t = (\mathbf{1}-\mathbf{W}(V_O^t)A) \circ \mathbf{W}(V_O^t)I_O + \mathbf{W}(V_O^t)A \circ \mathbf{W}(V_B^t)I_B, \tag{2}$$

where $\mathbf{W}(V_B^t) \in R^{n\times n}$ is a warping matrix such that $\mathbf{W}(V_B^t)I_B$ is the warped background component I_B according to the motion field V_B^t. Since the obstruction is between the camera and background objects, the alpha map shares the same motion of the obstruction component I_O, not the background component I_B. We can therefore simplify the formulation by defining $I_O = (1 - A) \circ I_O$ (with the abuse of the notation I_O), to get the following, simplified equation:

$$I^t = \mathbf{W}(V_O^t)I_O + \mathbf{W}(V_O^t)A \circ \mathbf{W}(V_B^t)I_B. \tag{3}$$

Note that in the case of reflection, since the alpha map is constant, $A = \alpha$, the formulation can be further simplified as $I^t = \mathbf{W}(V_O^t)I_O + \mathbf{W}(V_B^t)I_B$, where $I_O = (1 - \alpha)I_O$ and $I_B = \alpha I_B$. That is, the alpha map is essentially absorbed into the reflection and background components. Except for a small modification in the optimization for that case, which we will describe later on, both types of obstructions (opaque objects and reflective panes) are handled similarly by the algorithm.

Our goal is then to recover the background component I_B and the obstruction component I_O for the reference frame I^{t_0}, from an input image sequence $\{I^t\}$, without knowing the motion fields V_B^t and V_O^t, or the alpha map A (in the case of opaque occlusion).

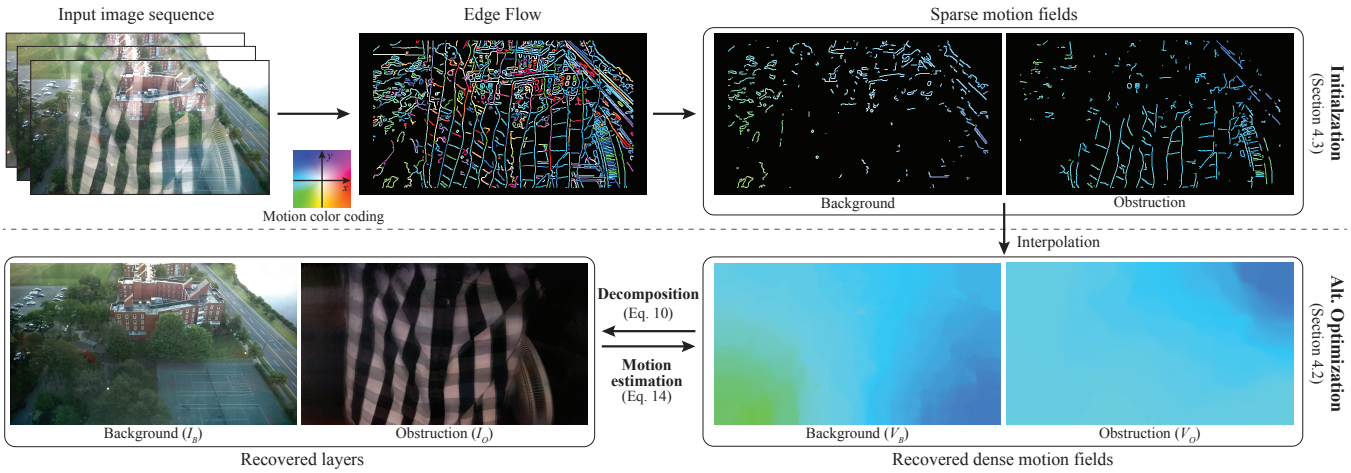


Figure 3: *Algorithm pipeline. Our algorithm consists of two steps: initialization and iterative optimization.* ***Initialization****: we first calculate the motion vectors on extracted edge pixels from the input images (we thicken the edge mask for a better visualization). Then we fit two perspective transforms (one for each layer) to the edge motion and assign each edge pixel to either the background layer or the obstruction layer. This results in two sets of sparse flow fields for the two layers (top right), which we then interpolate to produce an initial estimation of the dense motion fields for each layer (bottom right).* ***Optimization:*** *In this stage, we alternate between updating the motion fields, and updating the background and obstruction components, until convergence.*

4 Motion-based Decomposition

4.1 Formulation

Let us now discuss the optimization problem for recovering the background and obstruction components, I_B and I_O, from an input image sequence, $\{I^t\}$. We will first derive an algorithm for the more general case of an unknown, spatially varying alpha map, A, and then show a small simplification that can be used for reflection removal where we assume the alpha map is constant.

According to the image formation model (Eq. 3), we set our data term to be:

$$\sum_t \|I^t - \mathbf{W}(V_O^t)I_O - \mathbf{W}(V_O^t)A \circ \mathbf{W}(V_B^t)I_B\|_1, \qquad (4)$$

where $\{V_O^t\}$ and $\{V_B^t\}$ are the sets of motion vectors for the obstruction and background components, respectively.

To reduce the ambiguity of the problem, we include additional constrains based on priors on both the decomposed images and their respective motion fields. First, because the obstruction and background components are natural images, we enforce a heavy tailed distribution on their gradients [Levin and Weiss 2007], as

$$\|\nabla I_O\|_1 + \|\nabla I_B\|_1, \qquad (5)$$

where ∇I_B are the gradients of the background component I_B.

We assume that the alpha map is generally smoother than a natural image (smooth transitions in the blending coefficients). We assume the gradients follow a Gaussian distribution and penalize its l_2-norm:

$$\|\nabla A\|^2. \qquad (6)$$

We also assume that the background component and the obstruction component are independent. That is, if we observe a strong gradient in the input image, it most likely belongs either to the background component or the obstruction component, but not to both. To enforce this gradient ownership prior, we penalize the product of the gradients of background and obstruction, as

$$L(I_O, I_B) = \sum_x \|\nabla I_O(x)\|^2 \|\nabla I_B(x)\|^2, \qquad (7)$$

where x is the spatial index and $\nabla I_B(x)$ is the gradient of image I_B at position x.

Finally, as commonly done by optical flow algorithms [Black and Anandan 1996], we also enforce sparsity on the gradients of the motion fields, seeking to minimize

$$\sum_t \|\nabla V_O^t\|_1 + \|\nabla V_B^t\|_1. \qquad (8)$$

Combining all the terms above, our objective function is:

$$\min_{I_O, I_B, A, \{V_O^t\}, \{V_B^t\}} \sum_t \|I^t - \mathbf{W}(V_O^t)I_O - \mathbf{W}(V_O^t)A \circ \mathbf{W}(V_B^t)I_B\|_1$$
$$+\lambda_1\|\nabla A\|_2^2 + \lambda_2(\|\nabla I_O\|_1 + \|\nabla I_B\|_1) + \lambda_3 L(I_O, I_B) + \lambda_4 \sum_t \|\nabla V_O^t\|_1 + \|\nabla V_B^t\|_1$$

Subject to:

$$0 \leqslant I_O, I_B, A \leqslant 1, \qquad (9)$$

where $\lambda_1, \ldots, \lambda_4$ are weights for the different terms, which we tuned manually. For all the examples in the paper, we used $\lambda_1 = 1$, $\lambda_2 = 0.1$, $\lambda_3 = 3000$, and $\lambda_4 = 0.5$.[1] Similarly to [Szeliski et al. 2000], we also constrain the intensities of both layers to be in the range $[0, 1]$.

In Figure 4, we demonstrate the contribution of different components in our algorithm to the result, using a controlled sequence with ground truth decomposition. One of the main differences between our formulation and previous work in reflection removal is the use of dense motion fields instead of parametric motion. For comparison, we replaced the dense motion fields V_O^t and V_B^t in our formulation with a homography (the results using affine motion were similar). As can be seen in Figure 4(c,e), a dense motion representation greatly improves the quality and reduces artifacts. That is because the background/obstruction layer at different frames cannot be aligned well enough using parametric motion. Such misalignments show up as blur and artifacts in the decomposition. The decomposition quality also degrades slightly when not using the gradient sparsity prior, as shown in Figure 4(d).

[1] $L(I_B, I_O)$ is significantly smaller than other terms, and so we compensate for that by choosing a larger λ_3.

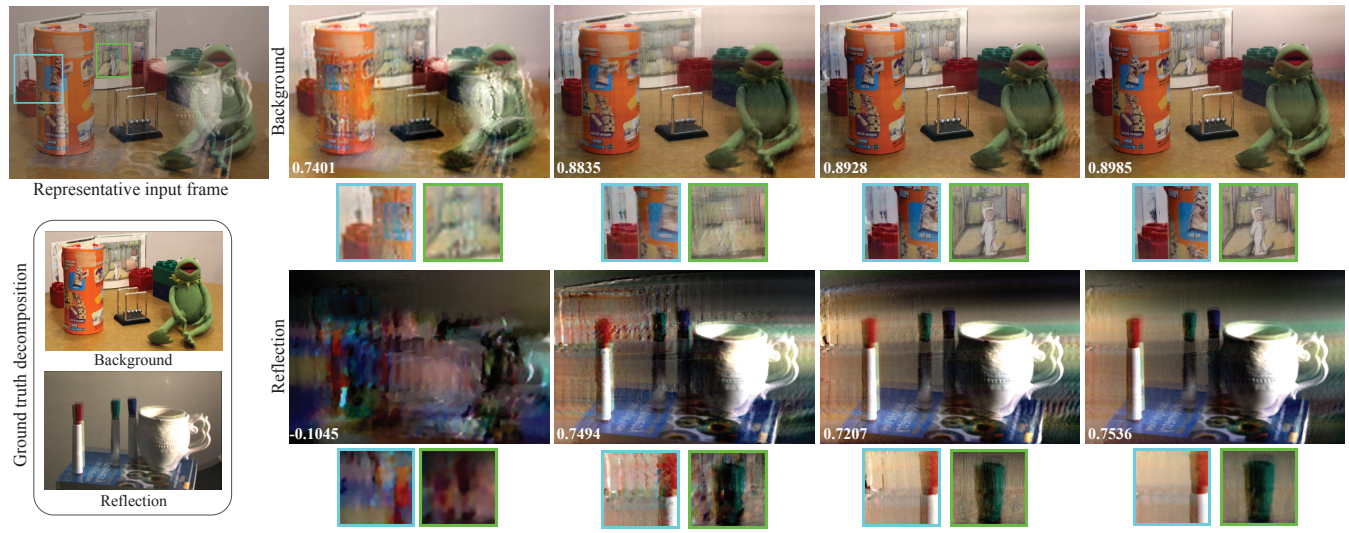


Figure 4: *The contribution of different components in the algorithm to the result. We use one of our controlled image sequences with ground truth decomposition (see Section 5), and compare our algorithm (e) with the following variants: (b) replacing the edge flow initialization with regular optical flow, (c) replacing the dense motion fields with parametric motion (we used projective transforms for both layers), and (d) when removing the sparsity prior on the image gradients (i.e. $\lambda_2 = 0$ in Eq. 9). One pair of ground truth background and reflection images for this 5-frame sequence are shown in (a) for reference. The normalized cross correlation (see Section 5 for details) between each recovered layer and the ground truth is shown at the bottom left of each image.*

4.2 Optimization

We use an alternating gradient descent method to solve Eq. 9. We first fix the motion fields $\{V_O^t\}$ and $\{V_B^t\}$ and solve for I_O, I_B and A, and then fix I_O, I_B and A, and solve for $\{V_O^t\}$ and $\{V_B^t\}$. Similar alternating gradient descent approach for joint estimation has been used in video super resolution [Liu and Sun 2014].

Decomposition step: fix motion fields $\{V_O^t\}$ and $\{V_B^t\}$, and solve for I_O, I_B, and A. In this step, we ignore all the terms in Eq. 9 that only consist of V_O^t and V_B^t:

$$\min_{\{I_O, I_B, A\}} \sum_t \|I^t - \mathbf{W}_O^t I_O - \mathbf{W}_O^t A \circ \mathbf{W}_B^t I_B\|_1 + \lambda_1 \|\nabla A\|^2 + \lambda_2 \left(\|\nabla I_O\|_1 + \|\nabla I_B\|_1\right) + \lambda_3 L(I_O, I_B), \quad (10)$$

Subject to

$$0 \leqslant I_O, I_B, A \leqslant 1, .$$

where we use $\mathbf{W}_O^t$ and $\mathbf{W}_B^t$ as short notes for $\mathbf{W}(V_O^t)$ and $\mathbf{W}(V_B^t)$. We solve this problem using a modified version of iterative reweighted least squares (IRLS). The original IRLS algorithm is designed for a non-constrained optimization with only l_1-and l_2-norms. To get this form, we linearize the higher-order terms in the objective function in Eq. 10. Let $\hat{I}_O$, $\hat{I}_B$ and $\hat{A}$ be the obstruction component, the background component, and the alpha map of the last iteration, respectively. Then the data term is linearized as[2]

$$\|I^t - \mathbf{W}_O^t I_O - \mathbf{W}_O^t A \circ \mathbf{W}_B^t \hat{I}_B - \mathbf{W}_O^t \hat{A} \circ \mathbf{W}_B^t I_B + \mathbf{W}_O^t \hat{A} \circ \mathbf{W}_B^t \hat{I}_B\|_1. \quad (11)$$

We also linearize the edge ownership term as:[3]

$$\lambda_3 (L(\hat{I}_O, I_B) + L(I_O, \hat{I}_B) - L(\hat{I}_O, \hat{I}_B)). \quad (12)$$

[2] we make an approximation commonly used in optimization: $xy \approx x\hat{y} + \hat{x}y - \hat{x}\hat{y}$, where $\hat{x}$ is very close to x and $\hat{y}$ is very close to y.

[3] $L(I_O, I_B) = \sum_x \|\nabla I_O\|^2 \|\nabla I_B\|^2 \approx \sum_x \|\nabla \hat{I}_O\|^2 \|\nabla I_B\|^2 + \|\nabla I_O\|^2 \|\nabla \hat{I}_B\|^2 - \|\nabla \hat{I}_O\|^2 \|\nabla \hat{I}_B\|^2 = L(\hat{I}_O, I_B) + L(I_O, \hat{I}_B) - L(\hat{I}_O, \hat{I}_B)$.

Second, we incorporate the two inequality constraints into our objective function using the penalty method [Luenberger 1973]. For example, for the non-negativity constraint $I_B \geqslant 0$, we include the following penalty function into the objective function:

$$\lambda_p \min(0, I_B^2), \quad (13)$$

where I_B^2 denotes element-wise square and λ_p is the weight for the penalty (we fix $\lambda_p = 10^5$). This function will apply a penalty proportional to the negativity of I_B (and will be zero if I_B is nonnegative).

Motion estimation step: fix I_O, I_B, A, and solve for the motion fields V_O^t and V_B^t. In this step, we ignore the terms dependent only on I_O, I_B, and A in Eq. 9:

$$\min_{V_O^t, V_B^t} \|I^t - \mathbf{W}(V_O^t) I_O - \mathbf{W}(V_O^t) A \circ \mathbf{W}(V_B^t) I_B\|_1 + \lambda_4 \left(\|\nabla V_O^t\|_1 + \|\nabla V_B^t\|_1\right). \quad (14)$$

This equation can again be solved using IRLS, similarly to the decomposition step.

Multi-scale Processing. To accelerate the algorithm, we optimize across multiple scales. We build a Gaussian pyramid for the input image sequence, and first solve all the unknowns—the motions, background and obstruction components, and the alpha blending mask—for the coarsest level. We then propagate the coarse solution to the next level using standard bicubic interpolation, and use it as initialization to solve for all the unknowns at the finer level. We get the final solution by solving the problem at the original resolution. At each level, we make a few iterations between solving for the motion and solving for the decomposition. The final algorithm is summarized in Algorithm 1. The functions $Decompose$ and $EstimateMotion$ are the two steps described above. Scale 1 is the coarsest scale and n_s is the finest scale (we use $3-4$ scales), and n_i is the number of iterations, which varies

Data: $\{I^t\}_t$, initial guess of I_O, I_B, A, $\{V_O^t\}$, and $\{V_B^t\}$
Result: I_O, I_B, A, $\{V_O^t\}$ and $\{V_B^t\}$.
for *Scale* $s = 1$ **to** n_s **do**
  $\{\hat{I}^t\} \leftarrow$ downsample input image sequence $\{I^t\}$ to scale s ;
  $I_O, I_B, A, \{V_O^t\}, \{V_B^t\} \leftarrow$ downsample/upsample I_O, I_B, A, $\{V_O^t\}$, and $\{V_B^t\}$ to scale s;
  for $i = 1$ *to* n_i **do**
    $I_O, I_B, A \leftarrow Decompose(\{\hat{I}^t\}, \{V_O^t\}, \{V_B^t\})$;
    $\{V_O^t\}, \{V_B^t\} \leftarrow EstimateMotion(\{\hat{I}^t\}, I_O, I_B, A)$;
  end
end

Algorithm 1: The motion-based decomposition algorithm.

across scales. We typically use 4 iterations for the coarsest scale and 1 iteration for each of the other scales.

Reflection Removal. As discussed earlier, in the case of a reflective pane, the alpha map A is essentially absorbed into the background and reflection images and there is no need to solve for it separately. We thus remove the prior term $\|\nabla A\|^2$ (Eq. 6) from the objective function, and only solve for I_O, I_B, $\{V_O^t\}$, and $\{V_B^t\}$. The data term in Eq. 9 becomes:

$$\|I^t - \mathbf{W}(V_O^t)I_O - \mathbf{W}(V_B^t)I_B\|_1, \tag{15}$$

and we use the same alternating gradient descent method described above to solve the decomposition. Currently we distinguish between the two sources of obstructions (opaque and reflective obstruction) manually.

4.3 Initialization

A key stage in our algorithm is the initialization of the motion fields and decomposition for the optimization. That part is vital for getting a clean separation as the objective function in Eq. 9 is nonlinear, and the IRLS algorithms (Section 4.2) may get stuck at a local minimum. Our initialization works by first estimating an initial motion field for each layer, then calculating the initial decomposition (background component I_B, obstruction component I_O, and alpha map A) from the initial motion fields. We will now describe these two steps in detail.

Initial motion estimation. Motion estimation in videos with reflection/occlusion is challenging. For videos with reflection, for example, each pixel has two motion vectors—one for the background and one for the reflectance. Therefore, we cannot directly use optical flow to estimate the motion fields. Notice that previous work in multi-layer optical flow [Jepson and Black 1993; Jojic and Frey 2001; Weiss and Adelson 1996; Liu et al. 2014] usually assume that the layer in the front occludes the layer on the back, while we assume the captured image is an additive superposition of two layers, so that both layers may be visible at the same location in the image.

Therefore, we propose to get an initial estimate of the motion fields using an *"edge flow"* algorithm. That is, we estimate a sparse motion field at each edge pixel identified in the image. As discussed before (Eq. 7), an observed image gradient will often belong to only one of the layers–either the background or the occlusion–but not to both. Indeed, we find that motion vectors estimated from pixels with large image gradients are generally more robust. A similar idea was used by [Kopf et al. 2013] for multi-view stereo.

More specifically, for a given input sequence, we first extract the edge map for each frame using the Canny edge detector [Canny 1986]. Then we calculate the motion of detected edge pixels by solving a discrete Markov random field (MRF):

$$\min_V \sum_{x \in Edge(I^1)} NCC(I^1(x), I^2(x + V(x))) + \sum_{x,x' \in Edge(I^1) \text{ and } (x,x') \in \mathcal{N}} S(V(x), V(x')), \tag{16}$$

where I^1 and I^2 are two neighboring input images, V is the motion field from image I^1 to I^2 that we want to estimate, $Edge(I^1)$ is the set of edge pixels in image I^1, and $\mathcal{N}$ is the 4-connected pixel neighborhood. Notice that different from the motion fields described in previous sections, this motion field is only defined on image edges. We also assume that V takes only integer values, so that $x + V(x)$ is also on the grid in the image I^2.

The first term in Eq. 16 is the data term that describes how well the patch located at position x in image I^1 matches the patch located at $x + V(x)$ in image I^2. Here, $NCC(I^1(x), I^2(x + V(x)))$ is the normalized cross correlation (NCC) between these two patches. The second term $S(V(x), V(x'))$ in Eq. 16 is the smoothness term that enforces neighboring edge pixels to have similar motion. We use the same penalty function described in Eq. 1 in [Kopf et al. 2013] for the smoothness term. We solve this Markov random field using belief propagation.

After obtaining the sparse motion field using edge flow, we separate it into two sparse motion fields, one for each layer. For this we first fit a perspective transformation to the sparse motion field using RANSAC, and assign all the edge pixels that best fit this transformation to the background layer, assuming the background pixels are more dominant. We then fit another perspective transformation to the rest of the edge pixels (again using RANSAC), and assign the pixels best fitting the second transformation to the reflection layer. Finally, we compute the dense motion fields for both layers using visual surface interpolation [Szeliski 1990]. An illustration of this process is shown in Figure 3.

In Figure 4 we compare our result with the proposed edge flow initialization (Figure 4(e)) with the one produced when initializing using standard optical flow, as done in [Szeliski et al. 2000] (Figure 4(b)).

Initial decomposition. To get an initial estimation of the decomposition, we first warp all the frames to the reference frame according to the background motion estimated in the previous step. In this warped input sequence, the background pattern should be roughly-aligned, while the obstruction component should be moving (as the motions of the two components are assumed to be different).

In the case of an opaque occlusion, we take the per-pixel mean across the warped input frames as an initial estimation of the background image. We also compute a binary alpha map by thresholding the per-pixel difference between the estimated background image and the input images. If the difference is larger than a threshold (we used 0.1), we set the alpha map to 0 at that location; otherwise we set the alpha map to 1. We then get the obstruction component by plugging the initial estimation of I_B and A to Eq. 3.

For a reflective pane, we take the initial estimation of the background image to be the minimum intensity across the warped frames[4].

[4] we compute the minimum intensity instead of the mean, since the minimum is an upper bound for the background's intensity. See [Szeliski et al. 2000] for details.

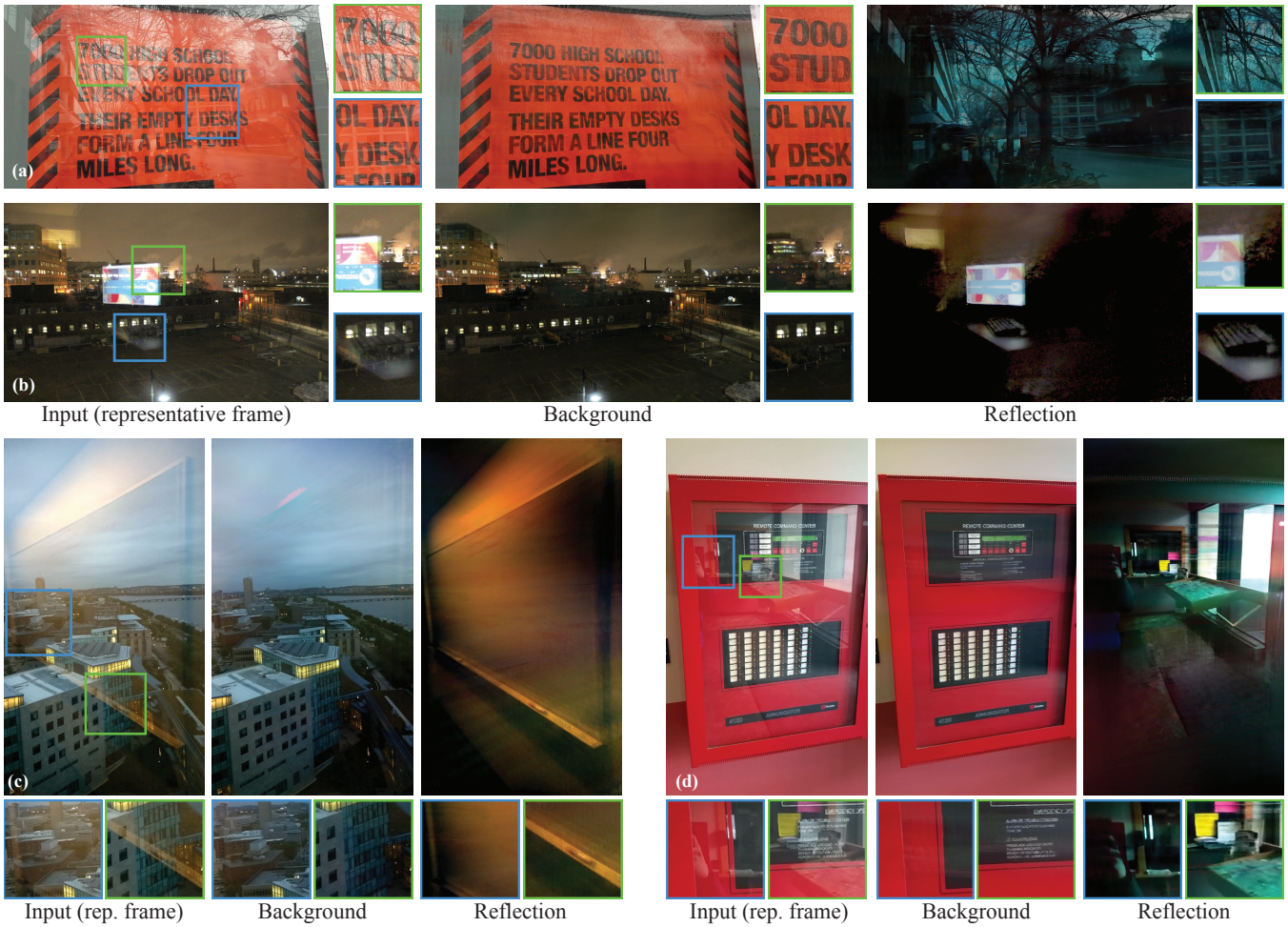


Figure 5: *Reflection removal results on four natural sequences. For each sequence we show a representative frame (each input sequence in this figure contain 5 frames), and the background and reflection images recovered automatically by the algorithm. Corresponding close-up views are shown next to the images (on the right of each image for the sequences in the top and middle rows, and below each image for the sequences in the bottom row). More results can be found in the supplementary material.*

5 Results

In this paper, we took most of the image sequences using the cell phone cameras of HTC One M8 and Samsung Galaxy 4, except for the sequence shown in Figure 5(c), which was taken using a Canon VIXIA HFG30. We processed the sequences on a desktop computer with Intel Xeon CPU (8 cores) and 64GB memory. With a non-optimized MATLAB implementation, processing a high-resolution image sequence (1152x648) took 20 minutes and required 3GB of RAM, and processing a low-resolution sequence (480x270) took 2 minutes and required 1GB memory. We also created a non-optimized Windows phone app prototype, implemented in C++, which produces equivalent results on the phone in less then 2 minutes for low-resolution images (480x270). Most of the sequences contain 5 frames sampled uniformly form the video, except for *gallery* (Figure 8, left) that contains 7 frames.

Removing Obstructions in Natural Sequences. We tested our algorithms under various scenarios, with different background objects, reflecting/occluding elements, and lighting conditions, and it worked consistently well in all these cases.

The top row in Figure 1 shows a common scenario when a photographer is taking a picture of an outside view through a window, while self reflection appears in the captured images. The strong reflection of the shirt covers most of image and obstructs a large part of the scene. Our algorithm generates a clean separation of the background and reflective components. Notice how the checkerboard pattern on the shirt is completely removed in the recovered background image, while most of the background textures, like the trees and the building, are well-preserved.

Figure 5 shows more scenarios where reflections frequently appear in photos. One common case is imaging reflective surfaces, such as a glass-covered billboard of a bus station (Figure 5(a)) and a glass-covered panel (Figure 5(d)). Notice that in Figure 5(a), due to the reflection, many letters on the billboard can be difficult to recognize. Even with such strong and textured reflection, our algorithm is able to produce a good separation, and words on the billboard become much clearer after the reflection is removed.

Reflections are also common when imaging outdoor scenes through windows during the night (Figure 5(b)) or at dusk (Figure 5(c)). Our algorithm is able to separate the background image from the reflection, producing a clean image of the outdoor scene, as well as revealing a lot of information about the indoor scene that is difficult to see in the original image. While the recovered background image is usually of most interest in this work, our high-quality reconstruction of the reflected scene may also be useful in some cases where more information needs to be extracted from an image sequence or a video.

Figure 6: *Occlusion removal results. For each sequence (row), we show a representative image from the input sequence (left column), the recovered background scene (second column) and the recovered occluding layer (third column). In the right column, we also show the alpha map, A, as inferred by the algorithm, with colors ranging from black (occlusion) to white (background). The alpha map, as expected, is tightly correlated with the occlusion image, but we show it here for completeness.*

Figure 1 (bottom row) and Figure 6 show some common scenarios when photographs are taken though more opaque, occluding elements, such as fences, dirty/textured windows ("simulated" by the SIGGRAPH logo), and surfaces covered by raindrops. In all cases, our algorithm is able to produce good reconstruction of the background scene with the occluding content removed.

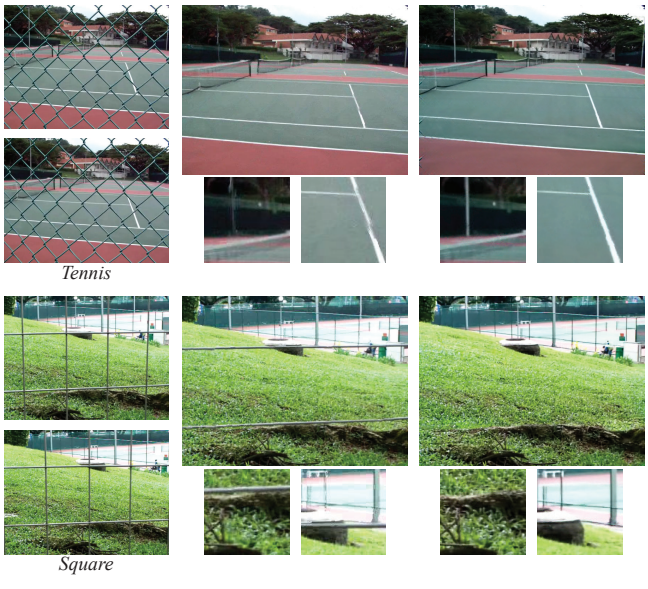


Figure 7: *Comparison with "Video De-Fencing" [Mu et al. 2012] on sequences from their paper. Left column: two representative frames from each input sequence. Middle column: backgrounds recovered by [Mu et al. 2012]. Right column: backgrounds recovered by our method.*

Comparison. We compared our algorithm with two state-of-the-art algorithms for removing reflections, [Guo et al. 2014] and [Li and Brown 2013], and with the recent work of [Mu et al. 2012] for fence removal from videos. In Figure 8 we show side-by-side comparisons of our results with the results by [Guo et al. 2014] and [Li and Brown 2013] on two sequences. On the "gallery" sequence[5], both [Guo et al. 2014] and our algorithm manage to produce a clean background, while there are noticeable artifacts near the boundary of the image in the results of [Li and Brown 2013]. Moreover, the reflection image produced by our algorithm is cleaner than the ones produce by the other methods. On "night", our algorithm generated a much cleaner separation than the other methods. Notice, for example, how the garbage bin is still visible in the results by the other methods, while our algorithm produces a clean separation in that region.

To compare with [Mu et al. 2012], we used the authors' own sequences "Tennis" and "Square", shown in Figure 7. On the "Tennis" sequence, the two methods produced comparable results, although our algorithm generated a slightly cleaner background image. On the "Square" sequence, since the direction of camera motion is mostly horizontal with a small vertical component (see input frames of "Square"), removing the horizontal part of the fence is challenging. Our algorithm can take advantage of this tiny vertical camera motion and remove both horizontal and vertical parts of the fence, while in the result by [Mu et al. 2012] only the vertical part of the fence is removed.

Quantitative Evaluation. To evaluate our results quantitatively, we took three sequences with ground truth background and obstructing layers, as shown in Figure 9. The first two sequences were taken through glass (reflective layer), while the third sequence was taken through an iron net (occluding layer). We captured a sequence of images over time while moving the camera, where at each time step we captured three images: a standard (composite)

[5]Original sequence by [Sinha et al. 2012]. We sampled 7 consecutive frames from their video.

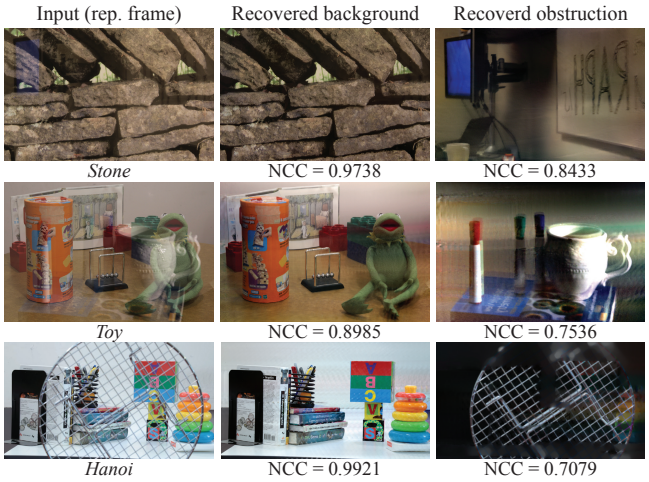
Figure 8: *Comparison with recent methods for reflection removal. More comparisons can be found in the supplementary material.*

image through the obstacle (Figure 9, left column), an image of just the background scene, captured by removing the obstacle, and an image of the reflective/occluding layer, which we captured by placing a black sheet of paper behind the obstacle, blocking the background component (and turning the glass into a mirror). All images were taken with a DSLR camera under fully manual control. The full sequences are available on the project web page.

We evaluate our algorithm by calculating the normalized cross correlation (NCC) of our recovered decomposition with the ground truth decomposition. The NCCs of our recovered background images with the ground truth backgrounds were 0.9738 (*Stone*), 0.8985 (*Toy*), and 0.9921 (*Hanoi*). On the project web page we give some additional comparisons with competing methods on those sequences. For example, on the *Stone* sequence, the NCCs of the recovered background images by [Guo et al. 2014] and [Li and Brown 2013] were 0.9682 and 0.9271, respectively, and the NCCs for their recovered reflections were 0.5662 and 0.2423, respectively.

Reflection-free Panoramas. Our algorithm can also be used for removing obstructions from panoramic images, for which the image capture already involves camera motion. In Figure 10, we show an example of automatically removing window reflections in a panorama of an outdoor scene taken from inside a building (Figure 10(a)). In this case we can use the camera motion that is already introduced by the user for creating the panorama, to also remove the reflections from it. We remove the reflection for each captured image using our algorithm as described above, and then stitch the results together to produce a reflection-free panorama image, as shown in Figure 10(b).

Our approach will not work if the camera motion is purely rotational. In that case, we can still produce reflection-free panoramas if



Method \ Sequence	*Stone*		*Toy*	
	I_B	I_O	I_B	I_O
[Li and Brown 2013]	0.9271	0.2423	0.7906	0.6084
[Guo et al. 2014]	0.9682	0.5662	0.7701	0.6860
Ours	**0.9738**	**0.8433**	**0.8985**	**0.7536**

Figure 9: *Quantitative evaluation on controlled sequences.* ***Top:*** *for each sequence (row), we show a representative frame from the controlled sequence (left column) and our decomposition result (middle and right columns). The normalized cross correlation (NCC) between each recovered layer and the ground truth (not shown, but available on the project web page) is written below the image.* ***Bottom:*** *numerical comparison with recent techniques for reflection removal (visual comparisons can be found on the web page).*

(a) Normal panorama processing

(b) Reflection-free panorama

Figure 10: *Reflection-free panoramas. Often when taking panoramas of outdoor scenes through windows, reflections of the indoor scene on the window cannot be avoided. Our algorithm can be used to produce reflection-free panoramas from the same camera motion used to capture the panoramas—i.e. without any additional work needed from the user. (a) The panorama produced with a mobile phone and a state-of-the-art stitching software, where indoor reflection are very apparent. (b) Our reflection-free panorama result. A panorama stitching of the estimated reflection is shown in the inset. On the right are close-up views of corresponding patches in the two panorama images.*

the user additionally translates the camera. However, we found that in many practical scenarios—a user taking a panorama with a hand-held camera or a phone—the camera motion will not be purely rotational, and will introduce sufficient parallax for the algorithm to work.

6 Conclusion

In this paper, we have demonstrated that high quality images can be captured automatically, with mobile cameras, through various types of common visual obstructions such as reflections, fences, stains, and rain-drop covered windows. Our algorithm can dramatically increase the quality of photos taken in such scenarios, and only requires that the user record a short image sequence while moving the camera—an interaction similar to taking a panorama. Our algorithm then combines the visual information across the image sequence to produce a clean image of the desired background scene with the visual obstruction removed. We have shown promising experimental results on various real and challenging examples, and demonstrated significant improvement in quality compared to prior work.

Acknowledgements

We thank Dr. Rick Szeliski for useful discussions and feedback, and the anonymous SIGGRAPH reviewers for their comments. Tianfan Xue is supported by Shell Research and ONR MURI 6923196.

References

BARNUM, P. C., NARASIMHAN, S., AND KANADE, T. 2010. Analysis of rain and snow in frequency space. *International Journal of Computer Vision (IJCV) 86*, 2-3, 256–274.

BE, E., YEREDOR, A., AND MEMBER, S. 2008. Blind Separation of Superimposed Shifted Images Using Parameterized Joint Diagonalization. *IEEE Transactions on Image Processing 17*, 3, 340–353.

BERTALMIO, M., SAPIRO, G., CASELLES, V., AND BALLESTER, C. 2000. Image inpainting. In *Computer Graphics and Interactive Techniques*.

BERTALMIO, M., BERTOZZI, A. L., AND SAPIRO, G. 2001. Navier-stokes, fluid dynamics, and image and video inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

BLACK, M. J., AND ANANDAN, P. 1996. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding 63*, 1, 75–104.

CANNY, J. 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 6, 679–698.

CRIMINISI, A., PÉREZ, P., AND TOYAMA, K. 2004. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing 13*, 9, 1200–1212.

FARID, H., AND ADELSON, E. H. 1999. Separating reflections and lighting using independent components analysis. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

GAI, K., SHI, Z., AND ZHANG, C. 2009. Blind separation of superimposed images with unknown motions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

GAI, K., SHI, Z., AND ZHANG, C. 2012. Blind separation of superimposed moving images using image statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 34*, 1, 19–32.

GARG, K., AND NAYAR, S. K. 2004. Detection and removal of rain from videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

GUO, X., CAO, X., AND MA, Y. 2014. Robust Separation of Reflection from Multiple Images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

HAYS, J., LEORDEANU, M., EFROS, A. A., AND LIU, Y. 2006. Discovering texture regularity as a higher-order correspondence problem. In *European Conference on Computer Vision (ECCV)*.

JEPSON, A., AND BLACK, M. J. 1993. Mixture models for optical flow computation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

JOJIC, N., AND FREY, B. J. 2001. Learning flexible sprites in video layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

KONG, N., TAI, Y.-W., AND SHIN, J. S. 2014. A physically-based approach to reflection separation: from physical modeling to constrained optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 36*, 2, 209–21.

KOPF, J., LANGGUTH, F., SCHARSTEIN, D., SZELISKI, R., GOESELE, M., AND DARMSTADT, T. U. 2013. Image-Based Rendering in the Gradient Domain. *ACM SIGGRAPH*.

LEVIN, A., AND WEISS, Y. 2007. User assisted separation of reflections from a single image using a sparsity prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 29*, 9, 1647–1654.

LEVIN, A., ZOMET, A., AND WEISS, Y. 2002. Learning to perceive transparency from the statistics of natural scenes. *Advances in Neural Information Processing Systems (NIPS)*.

LEVIN, A., ZOMET, A., AND WEISS, Y. 2004. Separating reflections from a single image using local features. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

LI, Y., AND BROWN, M. S. 2013. Exploiting Reflection Change for Automatic Reflection Removal. *IEEE International Conference on Computer Vision (ICCV)*.

LI, Y., AND BROWN, M. S. 2014. Single image layer separation using relative smoothness. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

LIU, C., AND SUN, D. 2014. On bayesian adaptive video super resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 36*, 2, 346–360.

LIU, C., YUEN, J., TORRALBA, A., SIVIC, J., AND FREEMAN, W. T. 2008. Sift flow: Dense correspondence across different scenes. In *European Conference on Computer Vision (ECCV)*.

LIU, S., YUAN, L., TAN, P., AND SUN, J. 2014. Steadyflow: Spatially smooth optical flow for video stabilization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

LUENBERGER, D. G. 1973. *Introduction to linear and nonlinear programming*, vol. 28. Addison-Wesley Reading, MA.

MU, Y., LIU, W., AND YAN, S. 2012. Video de-fencing. *IEEE Circuits and Systems Society*.

NEWSON, A., ALMANSA, A., FRADET, M., GOUSSEAU, Y., PÉREZ, P., ET AL. 2014. Video inpainting of complex scenes. *Journal on Imaging Sciences, Society for Industrial and Applied Mathematics*.

PARK, M., COLLINS, R. T., AND LIU, Y. 2008. Deformed lattice discovery via efficient mean-shift belief propagation. In *European Conference on Computer Vision (ECCV)*.

PARK, M., BROCKLEHURST, K., COLLINS, R. T., AND LIU, Y. 2011. Image de-fencing revisited. In *Asian Conference on Computer Vision (ACCV)*.

SAREL, B., AND IRANI, M. 2004. Separating transparent layers through layer information exchange. *European Conference on Computer Vision (ECCV)*.

SINGH, M. 2003. Computing Layered Surface Representations : An Algorithm for Detecting and Separating Transparent Overlays. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

SINHA, S., KOPF, J., GOESELE, M., SCHARSTEIN, D., AND SZELISKI, R. 2012. Image-based rendering for scenes with reflections. *ACM SIGGRAPH*.

SZELISKI, R., AVIDAN, S., AND ANANDAN, P. 2000. Layer extraction from multiple images containing reflections and transparency. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

SZELISKI, R. 1990. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 12*, 6, 513–528.

TSIN, Y., KANG, S. B., AND SZELISKI, R. 2006. Stereo matching with linear superposition of layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) 28*, 2, 290–301.

WEISS, Y., AND ADELSON, E. H. 1996. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE.

YAMASHITA, A., MATSUI, A., AND KANEKO, T. 2010. Fence removal from multi-focus images. In *International Conference on Pattern Recognition (ICPR)*.