Separating reflections from a single image using local features

Anat Levin Assaf Zomet Yair Weiss
School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
{*alevin,zomet,yweiss*}@*cs.huji.ac.il*

Abstract

When we take a picture through a window the image we obtain is often a linear superposition of two images: the image of the scene beyond the window plus the image of the scene reflected by the window. Decomposing the single input image into two images is a massively ill-posed problem: in the absence of additional knowledge about the scene being viewed there is an infinite number of valid decompositions.

In this paper we describe an algorithm that uses an extremely simple form of prior knowledge to perform the decomposition. Given a single image as input, the algorithm searches for a decomposition into two images that minimize the total amount of edges and corners. The search is performed using belief propagation on a patch representation of the image. We show that this simple prior is surprisingly powerful: our algorithm obtains "correct" separations on challenging reflection scenes using only a single image.

1 Introduction

When we view a scene through transparent glass, the resulting image is often similar to the one shown in the top of figure 1. The image is a linear superposition of two images: the face and the reflection of a bookshelf. Perceptually, this input image is decomposed into two transparent layers. Can we get a computer vision algorithm to find this decomposition?

Mathematically, the problem can be posed as follows. We are given an image $I(x,y)$ and wish to find two layers I_1, I_2 such that:

$$I(x,y) = I_1(x,y) + I_2(x,y) \quad (1)$$

This problem is obviously ill-posed: there are twice as many variables as there are equations. In the absence of additional prior knowledge there are an infinite number of possible decompositions. Figure 1 show a number of possible decompositions. All of them satisfy equation 1. In order to choose the "correct" decomposition, we need additional assumptions.

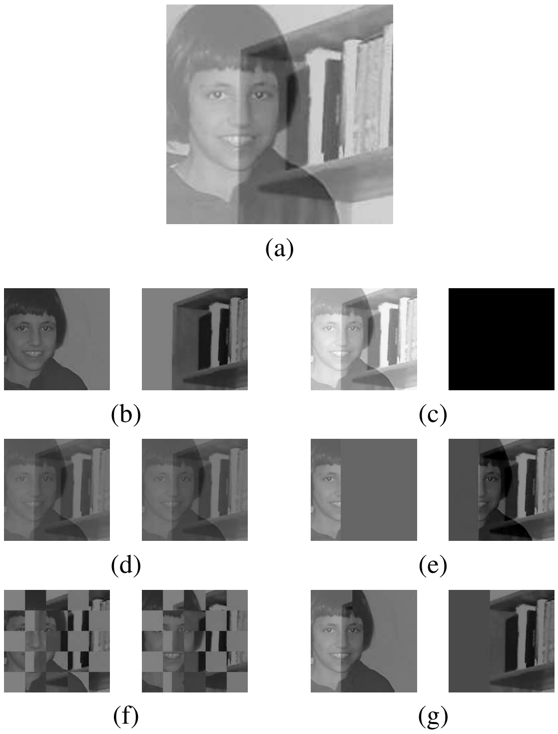


Figure 1: (a) Original input image (constructed by summing the two images in b). (b) the correct decomposition. (c)-(g) alternative possible decompositions. Why should the decomposition in (b) be favored?

One source of additional assumptions is the use of *multiple input images*. In [3, 10] two photographs of the same scene were taken with a different polarizing filter. The filter attenuates the reflection in different amounts and by using ICA on the two input images it possible to decompose the images. In [12, 6, 15] a movie sequence is analyzed in which the reflection and the non-reflected images have different motions. By analyzing the movie sequence, the two layers can be recovered.

But humans can decompose this input from *a single*

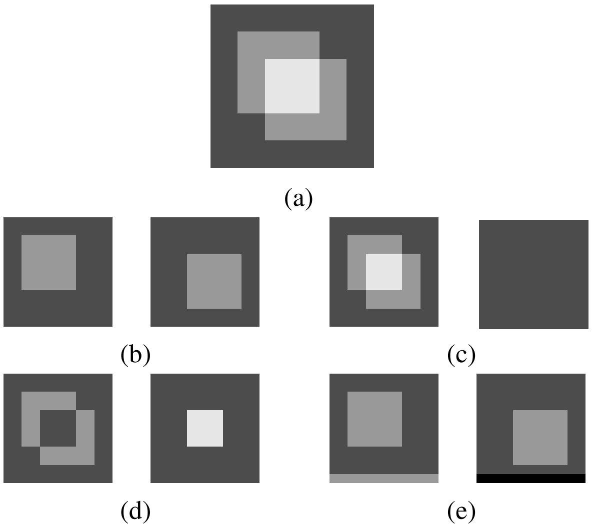


Figure 2: An input image and some decompositions

frame. On the related problem of separating shading and reflectance impressive results have been obtained using a single image [13, 4]. These approaches make use of the fact that edges due to shading and edges due to reflectance have different statistics (e.g. shading edges tend to be monochromatic). Unfortunately, in the case of reflections, the two layers have the same statistics, so the approaches used for shading and reflectance are not directly applicable.

One might think that in order to correctly decompose such images, an algorithm would need to know about faces and bookcases. The "bad" decompositions contain half a face and half a bookcase. An algorithm that possessed such high level knowledge would know to prefer the good decomposition. But can the "right" decomposition be favored without such high-level knowledge?

In this paper we present an algorithm that can decompose reflections images *using a single input image* and without any high level knowledge. The algorithm is based on a very simple cost function: it favors decompositions which have a small number of edges and corners. Surprisingly, this simple cost function gives the "right" decompositions for challenging real images.[1]

2 Transparency, edges and corners

To motivate the use of edges and corners, consider the simple image in figure 2(a). The input image can be decomposed into an infinite number of possible two layer decompositions. Figure 2(b-e) show some possible decompositions including the decomposition into two squares (the perceptually "correct" decomposition).

[1]An earlier version of this work appeared in [7]

Why should the "correct" decomposition be favored? One reason is that out of the decompositions shown in the figure, it minimizes *the total number of edges and corners*. The original image has 10 corners: 4 from each square and two "spurious" corners caused by the superposition of the two images. When we separate the image into two squares we get rid of the two spurious corners and are left only with eight corners. The decomposition shown in the third row increases the number of corners (it has 14 corners) while the bottom decomposition has 8 corners but increases the number of edges.

How can we translate the preference for a small number of edges and corners into cost function? We need to make two decisions (1) what operators to use for edge and corner detectors and (2) what mathematical form to give the cost. There is obviously a tremendous amount of literature in computer vision on how to find edges and corners and our goal in this work is not to devise sophisticated detectors. For synthetic images, we found that very simple operators work well: we used the gradient magnitude $|\nabla I(x,y)|$ as an edge operator and a Harris-like operator $c(x,y)$ as a corner operator:

$$c(x_0, y_0; I) = \det\left(\sum w(x,y) \begin{pmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{pmatrix}\right) \quad (2)$$

For the cost function, we simply use the negative log probability of these operators on natural images. Since the histogram of these operators can be fit with a generalized Gaussian distribution $\Pr(f) \propto e^{-|f|^\alpha/s}$ this leads to the following cost function for a single layer:

$$cost_1(I) = \sum_{x,y} |\nabla I(x,y)|^\alpha + \eta c(x,y;I)^\beta \quad (3)$$

with $\alpha = 0.7, \beta = 0.25, \eta = 15$. The values are obtained from the histograms of the operators in natural images and were shown to be critical for the successful decomposition [7].

The cost of a two layer decomposition is simply the sum of the costs for each layer separately:

$$cost_1(I_1, I_2) = cost_1(I_1) + cost_1(I_2)$$

When we evaluate this simple cost (equation 3) on the possible decompositions of the two squares (figure 2) we indeed find that it favors the "correct" decomposition out of the ones shown. These decompositions are, of course, just a handful out of an infinite number of possible decompositions. We can also consider a one dimensional family of solutions. We defined $s(x,y)$ the image of a single white square in the same location as the bottom right square in figure 2(a). We considered decompositions of the form $I_1 = \gamma s(x,y)$,$I_2 = I - I_1$ and evaluated the cost for different values of γ. Figure 3 shows the resulting one dimensional function. Indeed the minimum in this one dimensional subspace of solution is obtained at the "correct" solution.

In real images, however, the story is more complicated. Detecting edges by edge magnitude and corners with a simple Harris detector, gives responses in many seemingly flat regions of the image. As a result, when we apply the simple cost function to real images, it does not favor the "correct" decomposition (see figure 3). The minimum is obtained at $\gamma = 0$ indicating that a one layer solution is favored over the "correct" decomposition.

We have found, however, that a small modification can fix this problem. Instead of measuring gradients and Harris operators on the two layers, we first apply a nonlinear smoothing separately to each layer and then apply equation 3 to the *smoothed layers*. The intuition for this is that we want our edge and corner operators to return zero in regions that are nearly uniform. Since anisotropic diffusion transforms the nearly uniform regions into uniform ones, it greatly reduces the number of spurious "edges" and "corners" found by the gradient and Harris operators.

Thus our cost function for a single layer is now:

$$cost_2(I) = \sum_{x,y} |\nabla \hat{I}(x,y)|^{\alpha} + \eta c_2(x,y;\hat{I})^{\beta} \qquad (4)$$

where $\hat{I}$ is the layer after applying anisotropic diffusion [9]. The cornerness operator $c_2(x,y)$ is also slightly modified from equation 3 and is given in the appendix.

Figure 3 shows that once we use the modified cost function (equation 4) the "correct" decomposition is indeed favored in the one dimensional subspace. Now, the minimum is obtained with $\gamma = 1$.

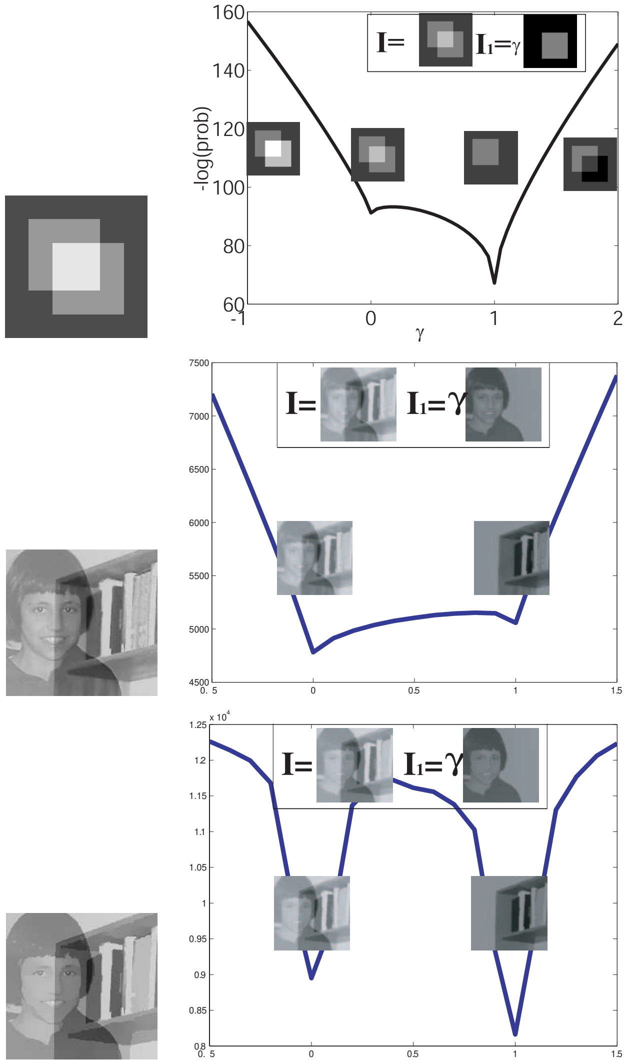


Figure 3: Testing a one dimensional subspace of solutions. **Top:** For the two squares stimulus the simple cost $cost_1$ gives a minimum at the preferred decomposition. **Middle:** For the real image, the simple cost $cost_1$ gives a minimum at a one layer solution. **Bottom:** For the real image, applying the cost to nonlinearly smoothed layers ($cost_2$) gives a minimum at the preferred decomposition.

3 Optimization

In figure 3 we saw that the minimum of the cost function (equation 4) in a one dimensional subspace is obtained at the "correct" decomposition. What we really want, of course, is to find the minimum out of *all possible decompositions*. How can we search this huge space?

One option is to use a continuous optimization algorithm such as gradient descent on equation 4. Note however, that equation 4 is highly nonlinear: not only are the edge and corner operators nonlinear but they also operate on the nonlinearly smoothed image $\hat{I}$. We therefore use an alternative approach whereby the problem is first *discretized* using a database of natural image patches. We then use loopy belief propagation to optimize the cost function over the discrete possible values.

3.1 Discretization using a natural images database

Instead of optimizing over the infinite space of possible decompositions we discretized the problem by dividing the image into small 7×7 overlapping patches and restrict the search to 20 possible decompositions for each patch. The use of a patch representation was motivated by the success of this approach in a number of recent vision applications [5, 2]. These 20 decompositions are defined by search-

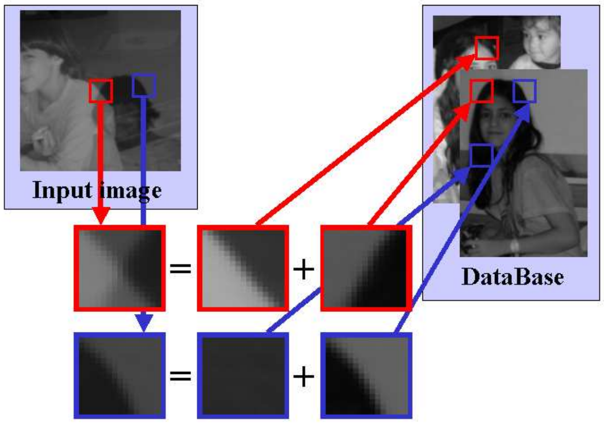


Figure 4: Optimization by discretization into patches. For each input patch we search a database of photographs for pairs of similar patches and use these to define a discrete optimization problem.

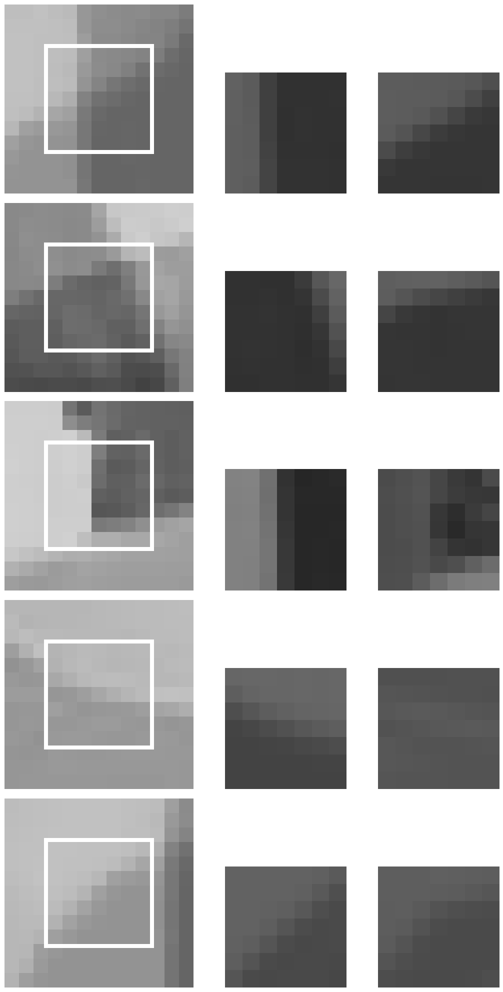
ing a database of natural images for pairs of patches that approximately sum to the input patch (see figure 4). The database of patches are simply all patches contained in two family photographs. This gives roughly 10^5 patches.

The use of such a large database of patches raises serious computational issues. For every patch in the image p we need to find a pair of patches (p_1, p_2) such that $p \approx p_1 + p_2$. Actually, to reduce the effects of patch contrast we search for a pair of patches $p \approx \alpha p_1 + \beta p_2 + \gamma$ (that is, we allow each patch to change its contrast and ignore the overall DC). A naive way of performing this search is to search all $10^5 \times 10^5$ possible pairs of patches but this will be incredibly slow.

To speed up the search we make use of the *sparsity* of derivative filters on natural images [11, 16]. We represent each patch in the database by the output of derivative filters on that patch. Since derivative filters tend to be sparse, we would expect high filter outputs in p_1 to still be present in $p = p_1 + p_2$ (because most likely, p_2 will not have high filter outputs in exactly the same location). Using this insight, we can find candidates p_1 by searching only 10^5 patches and then p_2 by performing a second search of 10^5 patches for $p - p_1$. Thus the search now requires $O(2 \times 10^5)$ operations rather than $O(10^5 \times 10^5)$.

We use this technique to find the 10 best candidates for p_1 and this yields a set of 10 possible decompositions $\{(p_1^i, p_2^i)\}$ for the input patch. $p \approx p_1^i + p_2^i$, $i = 1, ..., 10$. Where $p_j^i = a_i * q_j^i + b_i$ and q_j^i is a patch in the database. Figure 5 shows some typical decompositions.

The problem with the list of 10 decompositions obtained above is that it will very rarely include "one layer" decompositions: i.e. decompositions of the form $p = p + 0$. This is due to two reasons. First, a single patch similar to p may not be found in the database if p contains complex structure (e.g. the "T" junction in the middle of figure 5). Second, even when a similar patch is in the database, one can always obtain a better fit with two patches rather than one (e.g. the edge in the bottom of figure 5).



(a) (b) (c)

Figure 5: Local patches decomposition: (a) input image region. (b), (c) a possible two patches decomposition.

Therefore, for each of the 10 decompositions $p \approx p_1^i + p_2^i$ obtained from the database, we also consider the decomposition $p \approx \tilde{p}_1^i + \tilde{p}_2^i$ such that $\tilde{p}_1^i = p_1^i + p_2^i$ and $\tilde{p}_2^i \equiv 0$ is a zero patch. This yields 20 possible decompositions for each input patch. Finally for each decomposition (p_1, p_2) we add the symmetric decomposition (p_2, p_1) to obtain 40 possible decompositions for each input patch.

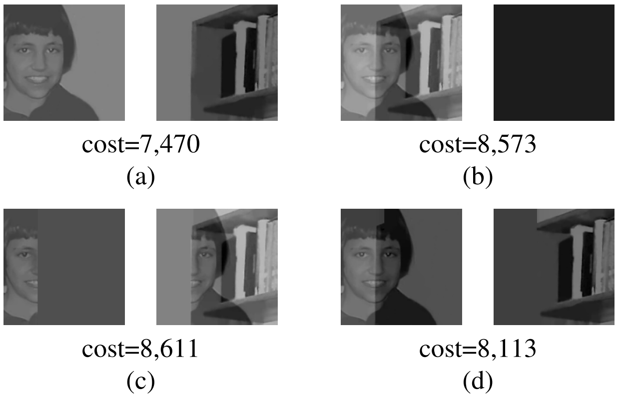


Figure 6: Testing different decompositions within the patches search space. (a) The patch from the database most similar to the original layer is chosen in each layer. (b) The patch from the data most similar to the input (sum) image is chosen in layer 1, and a zero patch in layer 2. (c) The most similar patch is chosen, but along the shelves edge, the layers where flipped. (d) The most similar patch is chosen, but along the vertical shelves edge, the layers where flipped.

3.2 Belief Propagation on Patches

By using the patch discretization we reduced the space of possible decompositions but we still have a huge number (40^N) of possible decompositions. Figure 6 shows a number of these decompositions along with their costs. While the "correct" decomposition indeed has minimal cost among the ones shown, we need an algorithm that can find it.

In order to find the most probable decomposition, we follow [5] and use max-product belief propagation on the patch representation. We define a two dimensional grid in which each node corresponds to a patch in the image and is connected to four other nodes corresponding to neighboring patches. To use BP we maximize a "probability" which is inversely related to the cost:

$$\begin{aligned} \Pr(I_1, I_2) &\propto e^{-\beta cost_2(I_1, I_2)} \\ &= e^{-\beta \sum_r cost_2(p_1(r), p_2(r))} \end{aligned}$$

Where $(p_1(r), p_2(r))$ denotes the two patches chosen at location r.

Since the cost can be simplified into a sum of local costs, the "probability" is a product of local "probabilities". We use this to rewrite the probability as:

$$\Pr(\{p_1(r), p_2(r)\}_{r=1}^{N}) = \tag{5}$$
$$\frac{1}{Z} \prod_r \Psi_r(p_1(r), p_2(r)) \prod_{<rs>} \Psi_{rs}(p_1(r), p_2(r), p_1(s), p_2(s))$$

The local potential $\Psi_r(p_1(r), p_2(r))$ is defined by:

$$\Psi_r(p_1(r), p_2(r)) = exp\left(-\beta_1 cost_2(p_1(r)) - \beta_1 cost_2(p_2(r))\right) \tag{6}$$

with $cost_2(p)$ given by equation 4.

Similar to previous uses of BP using a patch representation [5] we use pairwise potentials between patches that require neighboring patches to agree on the pixels in the overlap region.

Since equation 6 is completely symmetric in $\{p^1\}$ and $\{p^2\}$ we break the symmetry by choosing *a single location* and requiring that p_1 in that location be the zero patch. We choose this location automatically by finding a patch in the input image that is similar to a straight edge.

We found that max product BP when applied to the full image often failed to converge. In order to find an update order for which the algorithm converged, we randomly divided the input image into a list of ordered subimages and ran BP for a fixed number of iterations within the first subimage. We then fixed the messages within the first subimage and proceeded to run BP on the second subimage. We continues this process until BP had been run on all subimages. We repeated this process for 100 different divisions of the image into subimages and chose as the output of the algorithm, the result that had the lowest cost.

4 Results

The belief propagation algorithm outputs two patches $p_1(r), p_2(r)$ for every patch in the input image that is most "probable" given equation 6. In order to piece these patches together into two output images we need to take into account the fact that each patch is only defined up to an arbitrary DC level. To reconstruct layer 1 we take from each patch $p_1(r)$ its gradients and build a gradient field for layer 1. We then search for an image I_1 whose gradients minimize the L_1 norm from the gradient field. We repeat the process for layer 2. For a location r for which BP indicated that $p_1(r) = 0$ we set the gradients of layer 1 to be zero and the gradients of layer 2 to be the gradients of the input image. As we discuss below, by copying gradients of layer 2 from the original image, we are able to output textured layers even though the cost is calculated on anisotropically smoothed layers.

Figure 7 shows the output of our algorithm on a number of input images. For the purpose of testing our model, the top image was generated synthetically by summing two real images. The bottom three images are photographs of scenes with transparency. The same parameters were used for all the images. The algorithm received as input *a single image*. Even though the algorithm knows nothing about faces, bookcases, dolls or computers, it finds the "correct" decompositions by minimizing the number of edges and corners.

Figure 7: Results of our algorithm. The algorithm receives a single image as input and outputs two layers as output by minimizing the total number of edges and corners. Even though it has no high level knowledge and is searching an exponentially large number of possible decompositions, it finds a "perceptually correct" decomposition.

Figure 8: Images without transparency are not separated by the algorithm, even though all the parameters are the same.

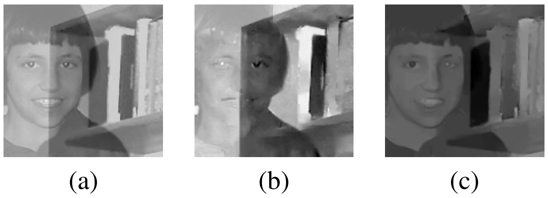


Figure 9: Importance of minimizing edges and corners to obtain the correct results. Decomposition results when local potentials are ignored.

Since the cost is calculated on anisotropically smoothed layers it ignores small gradients that are due to texture. In other words, small texture gradients can be assigned to any layer without any change in the cost. Our reconstruction process, however, prefers to put all small gradients in a patch in a single layer. Thus in the picture of the man's back seen through a window, both the back and the reflection contained small gradients but in the output of our algorithm all the small gradients were assigned to the reflection layer and the man's back appears artificially smooth.

Figure 8 shows the output on an image without reflections. There are still many possible decompositions of this image, and even though the same parameters are used, the algorithm decided not to break the image into two layers, since a lower cost is obtained with a one layer solution.

How important is the cost function $cost_2$ in getting these results? When we set the local potentials to be uniform in the graphical model (i.e. we completely ignore the minimization of edges and corners) and run BP as before, the result is determined by the pairwise agreement between patches. Without the minimization of edges and corners, BP always finds one layer solutions: apparently these have the highest agreement in the overlap between patches. When we artificially remove the one layer decompositions from the range of possible decompositions (by removing the 20 candidate patches that correspond to one layer decompositions from the discretization) we still obtain solutions that are quite wrong (see figure 9). Thus it really is the minimization of edges and corners that allow us to obtain the "correct" decompositions as shown in figure 7.

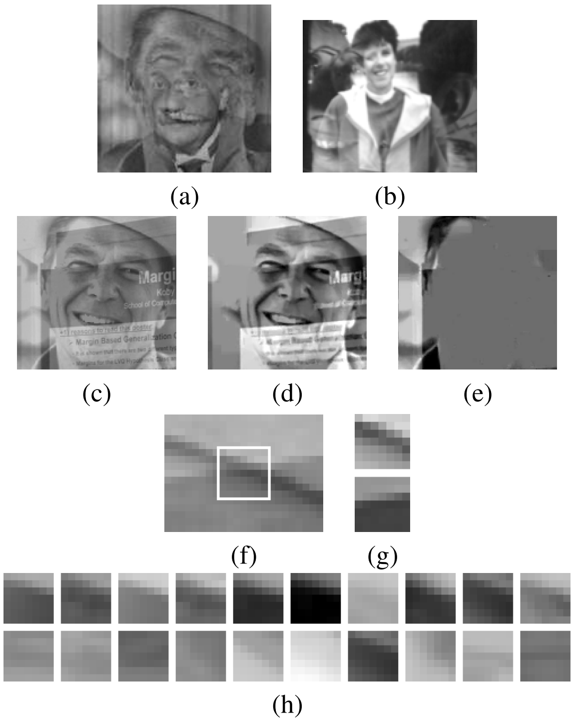


Figure 10: Failure of our decomposition algorithm. (a,b) two images for which our algorithm prefers a one layer decomposition. (c) a synthetic image formed by summing two real images. Our algorithm outputs the wrong decomposition in (d,e) even though the cost prefers the right decomposition. (f) Zooming into a x-junction on the forehead and its expected decomposition (g). (h) The 10 candidate decomposition found by database search. None of them seem to be "good".

5 Discussion

The problem of separating reflections from a single image is massively ill-posed: we are given one image as input and need to generate two images as output. One might think that very high level prior knowledge is needed to perform this task. In this paper we have shown that very simple low-level knowledge: that edges and corners are rare, can give surprisingly good decompositions on many real images.

Our current approach is only a first step towards separating arbitrary reflection images from a single image. Our algorithm often fails to separate reflections correctly, even when the correct decomposition is perceptually obvious (e.g for the input images in figures 10(a-b), our algorithm prefers a one layer decomposition).

To gain understanding into the sources of these failures we constructed the synthetic image in figure 10(c) by summing two real images. We examined a one dimensional family of solutions and found that the correct decomposition is indeed favored by our cost. Nevertheless, when we run our algorithm on this input image, we obtain the decomposition shown in figure 10(d-e). This decomposition has *worse* cost compared to the correct one.

Why does our optimization algorithm fail? In this case, the discretization seems to be at fault. Figure10 shows an "x" junction on the forehead and the desired decomposition. The candidates found by our database search do not include the decomposition. In future work we will experiment with other ways to find candidate decompositions, perhaps similar to the approach used in [14]. Even when correct candidates exist, the discrete optimization is so difficult that BP may fail to find the best one. In future work we will explore alternative optimization techniques such as graph cuts [1].

We were surprised with the power of the simple low-level prior we are using. We are optimistic that more sophisticated local features may enable separating reflections from arbitrary images.

6 Appendix: implementation details

For the anisotropic diffusion we used Malik and Perona's algorithm with $\sigma = 0.06$ (image gray levels were in $[0, 1]$). We measured cornerness by dividing the determinant of the Harris matrix (equation 3) by its trace. The gradients used in the Harris matrix had magnitude based on the gradients of the anisotropically smoothed layers but their orientations were determined by anisotropically smoothing the orientations of the original image and not the orientation of the gradients in the diffused layers.

To find candidate patches $p \approx p_1 + p_2$ we searched the database for the patches q minimizing $\sum_i |f_i * p - f_i * q \cdot s||f_i * q \cdot s|$ where f_i is a collection of directional filters at different locations, orientations and phases [8][2]. For any candidate patch q the optimal contrast s was found by minimizing this expression.

The pairwise potentials in the graphical model to enforce consistency was set to zero if the average L_1 distance (after accounting for DC differences) between the two patches in the overlap region was greater than 4.6 gray values and it was set to 1 if the two patches agreed completely. For intermediate levels of agreement the potentials varied smoothly. We increased the threshold so that for any two neighboring locations, at least 20% of the pairs have nonzero potentials.

References

[1] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, 1999.

[2] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of SIGGRAPH*, pages 341–346, August 2001.

[3] H. Farid and E.H. Adelson. Separating reflections from images by use of independent components analysis. *Journal of the optical society of america*, 16(9):2136–2145, 1999.

[4] G. D. Finlayson, S. D. Hordley, and M. S. Drew. removing shadows from images. In *ECCV*, 2002.

[5] W.T. Freeman and E.C. Pasztor. Learning to estimate scenes from images. In M.S. Kearns, S.A. Solla, and D.A. Cohn, editors, *Adv. Neural Information Processing Systems 11*. MIT Press, 1999.

[6] M. Irani and S. Peleg. Image sequence enhancement using multiple motions analysis. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, pages 216–221, Champaign, Illinois, June 1992.

[7] A. Levin, A. Zomet, and Y. Weiss. Learning to perceive transparency from the statistics of natural scenes. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, 2002.

[8] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. In K.L. Boyer and S. Sarkar, editors, *Perceptual Organization for artificial vision systems*. Kluwer Academic, 2000.

[9] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. PAMI*, 8(5):565–593, 1989.

[10] Y. Shechner, J. Shamir, and N. Kiryati. Polarization-based decorrelation of transparent layers: The inclination angle of an invisible surface. In *Proceedings ICCV*, pages 814–819, 1999.

[11] E.P. Simoncelli. Statistical models for images:compression restoration and synthesis. In *Proc Asilomar Conference on Signals, Systems and Computers*, pages 673–678, 1997.

[12] R. Szeliksi, S. Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In *Proceedings IEEE CVPR*, 2000.

[13] M. Tappen, W.T. Freeman, and E.H. Adelson. Recovering intrinsic images from a single image. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, 2002.

[14] M. F. Tappen, B. C. Russell, and W. T. Freeman. Exploiting the sparse derivative prior for super-resolution and image demosaicing. In *IEEE Workshop on Statistical and Computational Theories of Vision*, 2003.

[15] Y. Tsin, S.B. Kang, and R. Szeliski. Stereo matching with reflections and translucency. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, pages 702–709, 2003.

[16] M. Zibulevsky, P. Kisilev, Y. Zeevi, and B. Pearlmutter. Blind source separation via multinode sparse representation. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, 2001.

[2] We thank S. Belongie for the filtering code