109-2 DIP TERM PROJECT PROPOSAL

Reflection Removal

TEAM 11
B07209038 大氣三 邱曼甯
B07502165 機械三 賴昭蓉
B07902063 資工三 陳耕宇

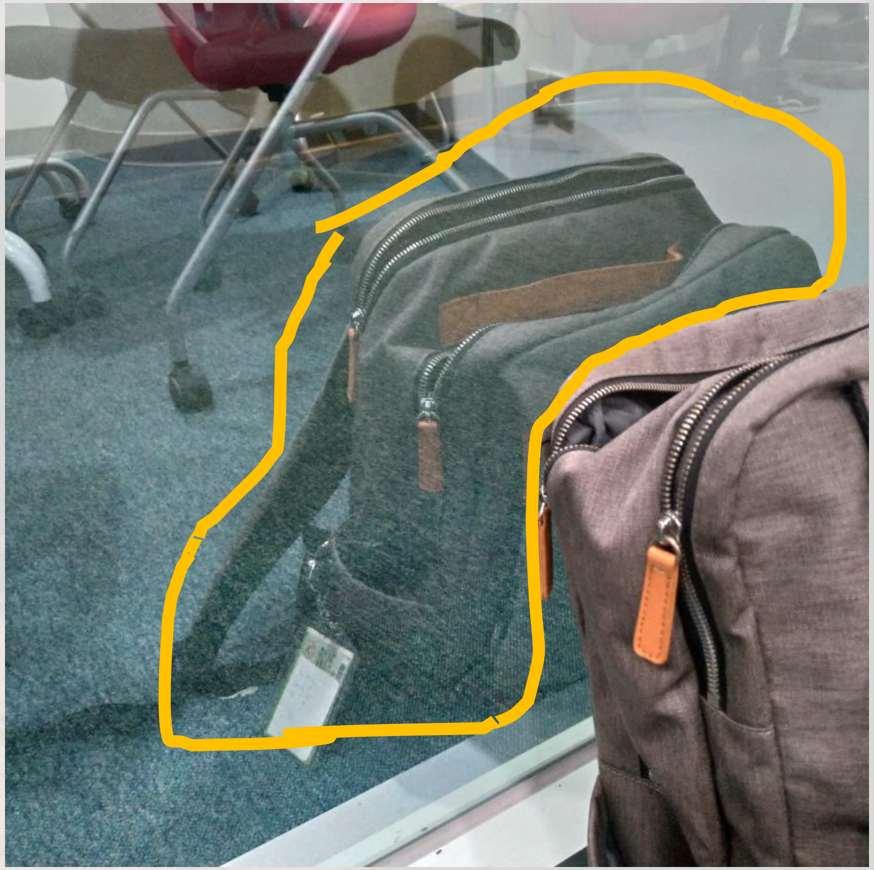
Outline

- Motivation
- Problem definition
- Algorithm
- Expected results
- Reference

Motivation

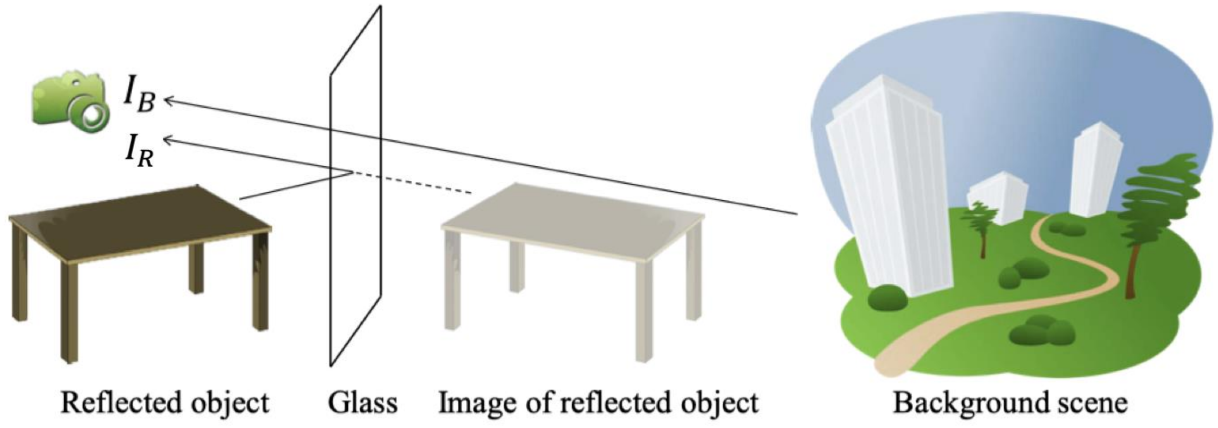
Reflection produced by glass windows is a bothering issue for photographers. Sometimes we only want the scene behind the window or object in front of the window.

We want to implement the Reflection removal method to get rid of undesired reflections.



Problem Definition

The figure shows the image formation model, in which the camera is taking a picture through the glass. The resulting image $\mathrm{I} \in R^n$ will contain two layers, one is the reflected object, denoted by $I_R \in R^n$, the other is the background scene, denoted by $I_B \in R^n$.



Problem Definition

Therefore, we have:

$$I = I_R + I_B$$

Our goal is to remove the reflected object part in the image.

We can remove the reflected object if we get I_R and I_B seperatedly.

Algorithm

Three papers and three methods

1. **Smoothness Approach**
   - Paper : Single Image Layer Separation Using Relative Smoothness
2. **Motion Approach**
   - Paper : A computational approach for obstruction-free photography.
3. **User-assisted Separation with Sparse Prior**
   - Paper : User assisted separation of reflections from a single image using a sparsity prior

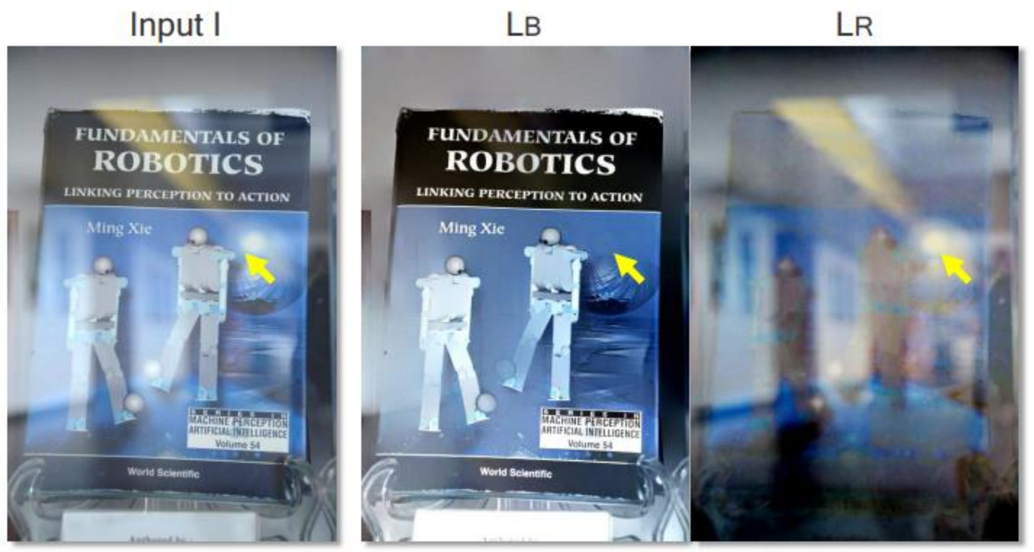
1 Smoothness Approach

LI, Yu; BROWN, Michael S. Single image layer separation using relative smoothness. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014. p. 2752-2759.

An image with reflection can be expressed as

$$I = L_1 + L_2 = L_B + L_R \otimes \mathrm{h}$$

L1 : sharper component
L2 : smoother component
h : Gaussian kernel



Convolution Kernel

Sharp

$$f_1 = \begin{bmatrix} -1 & 1 \end{bmatrix} \qquad f_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Smooth

$$f_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Objective function

$$\min_{L_1,L_2} \sum_{i,j} \left(\rho(L_1 * f_j)_i + \lambda\rho(L_2 * f_j)_i^2\right)$$

Goal

Find the $\boldsymbol{L_1}$ and $\boldsymbol{L_2}$ to satisfy

1. Maximize $\boldsymbol{L_1 \otimes f_1 + L_1 \otimes f_2}$
2. Maximize $\boldsymbol{L_2 \otimes f_3}$

Algorithm Layer Separation using Relative Smoothness
Input: image I; smoothness weight λ; initial β_0; iterations number i_{max}; increasing rate η
Initialization: $L_1 \leftarrow I$; $\beta \leftarrow \beta_0$; $i \leftarrow 0$
while $i < i_{max}$ do
upgrade g_i^j using Eqn. 7
compute L_1 using Eqn. 8
normalize L_1 using Eqn. 9
$\beta = \eta * \beta$
$i \leftarrow i + 1$
end while
$L_2 = I - L_1$
Output: The estimation of two layers L_1 and L_2

Algorithm

$$g_i^j = \begin{cases} F_i^j L_1, & (F_i^j L_1)^2 > \frac{1}{\beta} \\ 0, & \text{otherwise.} \end{cases}$$

$$L_1 = \mathcal{F}^{-1}(A),$$
$$A = \frac{\beta \sum_j (\mathcal{F}(F^j)^{\star} \mathcal{F}(g^j)) + \lambda \mathcal{F}(F^3)^{\star} \mathcal{F}(F^3) \mathcal{F}(I)}{\beta \sum_j (\mathcal{F}(F^j)^{\star} \mathcal{F}(F^j)) + \lambda \mathcal{F}(F^3)^{\star} \mathcal{F}(F^3) + \tau}$$

$$\min_t \sum_i m_i((L_1)_i + t - lb_i)^2 + \sum_i n_i((L_1)_i + t - ub_i)^2,$$

Algorithm Layer Separation using Relative Smoothness

Input: image I; smoothness weight λ; initial β_0; iterations number i_{max}; increasing rate η

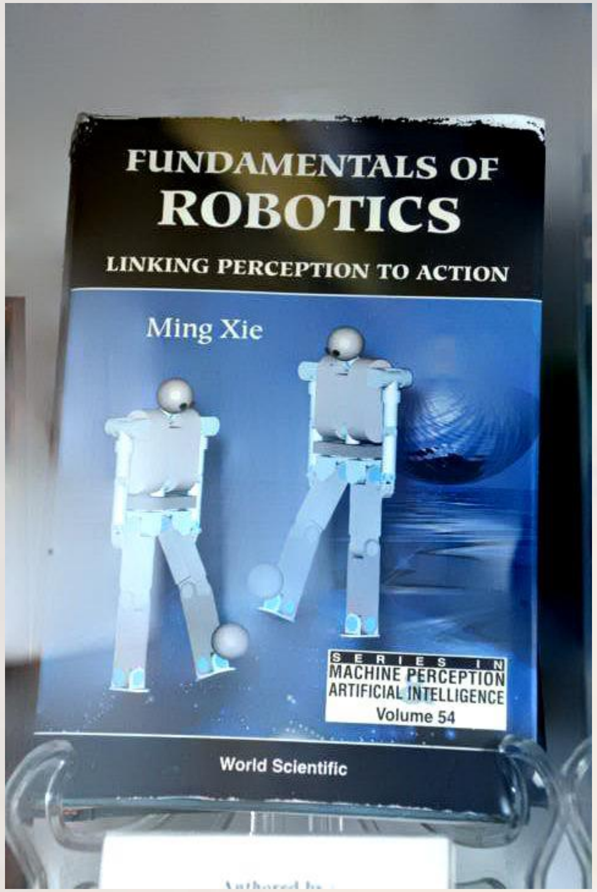
Initialization: $L_1 \leftarrow I$; $\beta \leftarrow \beta_0$; $i \leftarrow 0$

- while $i < i_{max}$ do
  - upgrade g_i^j using Eqn. 7
  - compute L_1 using Eqn. 8
  - normalize L_1 using Eqn. 9
  - $\beta = \eta * \beta$
  - $i \leftarrow i + 1$
- end while
- $L_2 = I - L_1$

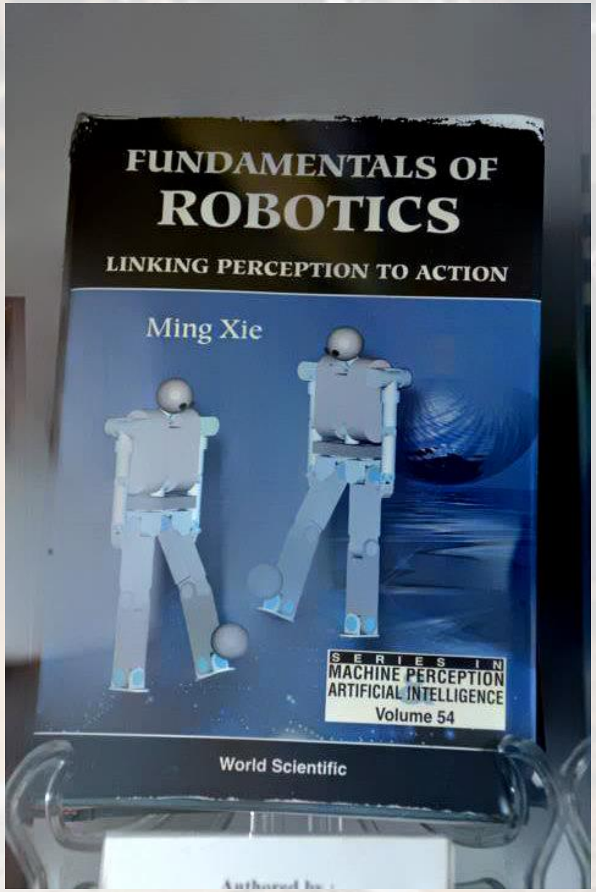
Output: The estimation of two layers L_1 and L_2

Experiment result

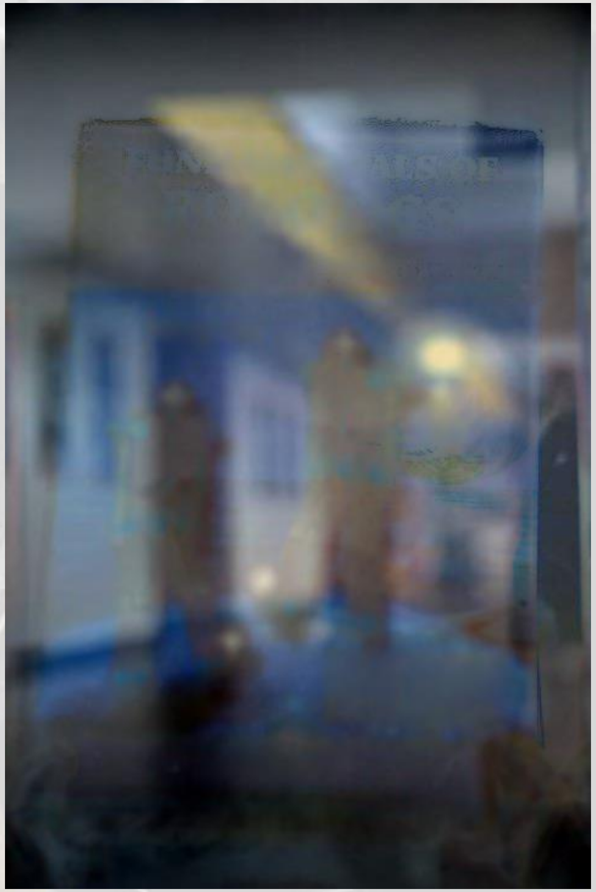
Paper test photo (725*480, 6 sec)



=



+

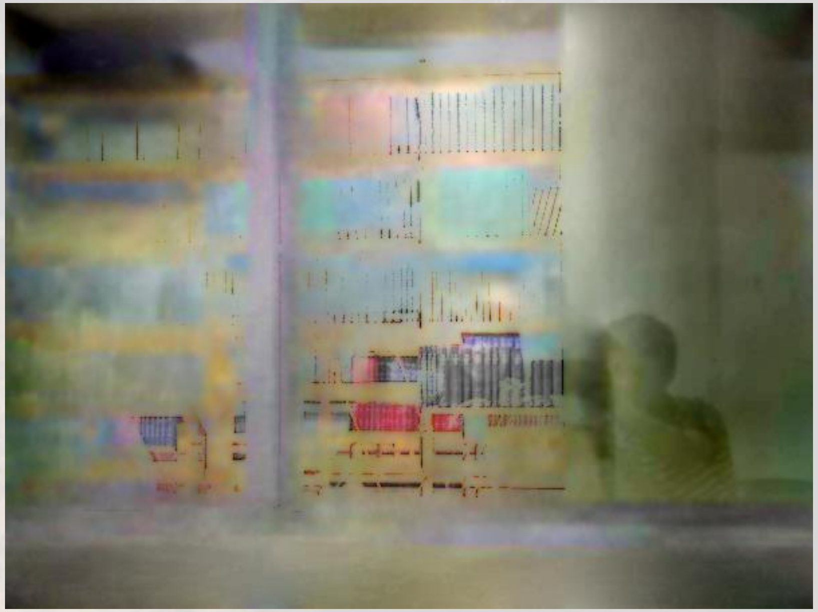


Experiment result

Our photo (451*602 , 8 sec)

=

+



2 Motion Approach

XUE, Tianfan, et al. A computational approach for obstruction-free photography. ACM Transactions on Graphics (TOG), 2015, 34.4: 1-11.

User should take a short image sequence

$$I^t = W(V_R^t) I_R + W(V_B^t)\, I_B$$

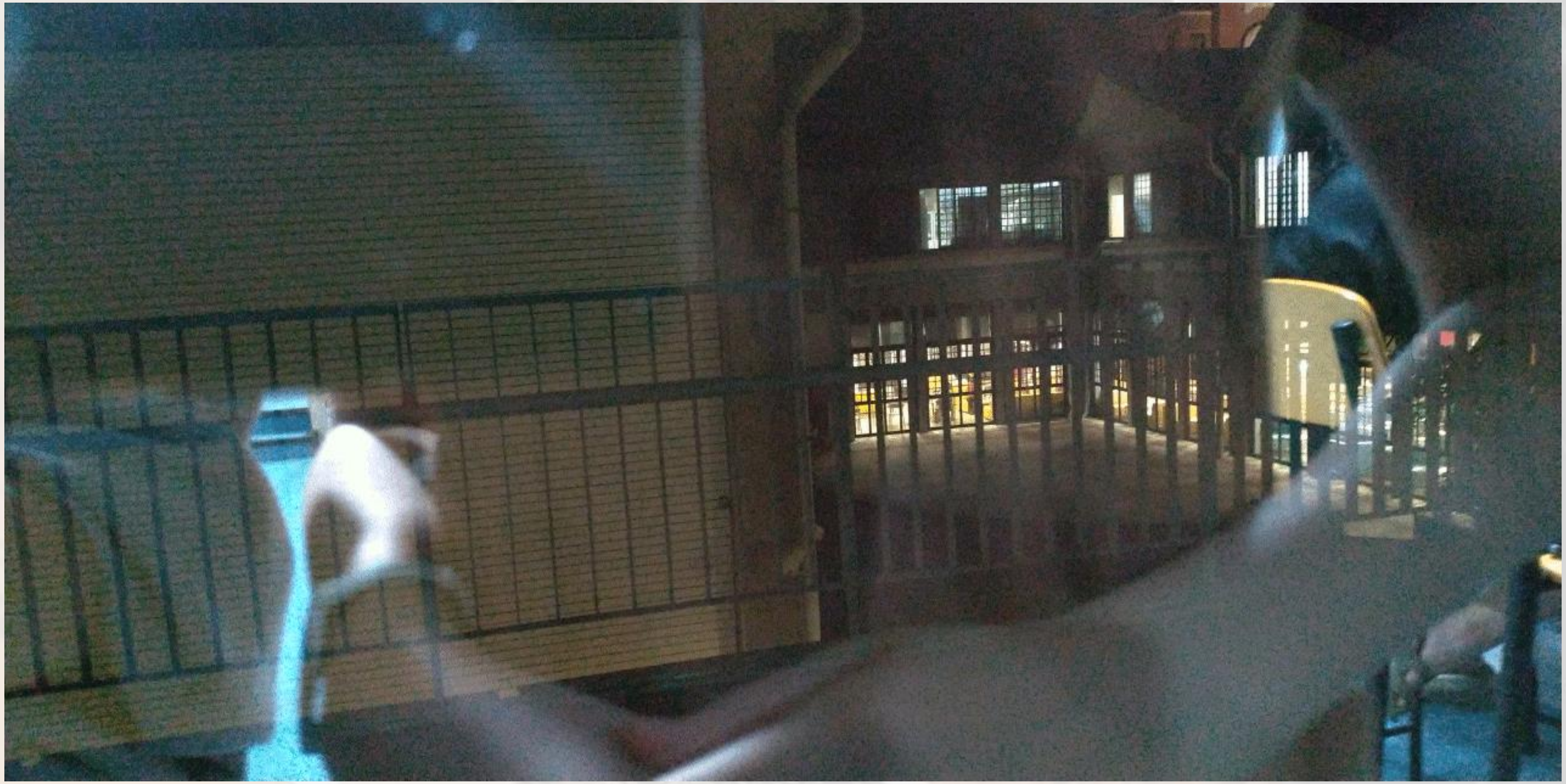
I^t is the image at frame t

V_B^t denotes the motion field for the background layer from reference frame t_0 to the frame t. Similarly, V_R^t is the motion field for the reflected-object layer.

$W(V_B^t) \in R^{n \times n}$ is a warping matrix such that $W(V_B^t) I_B$ is the warping background component I_B according to the motion field V_B^t

User should take a short image sequence

$$I^t = W(V_R^t) I_R + W(V_B^t)\, I_B$$



The goal is to recover the background layer $\mathrm{I_B}$ and the reflected-object image $\mathrm{I_R}$ for the reference frame I^{t_0}, from an input image sequence $\{\mathrm{I^t}\}$ without knowing the motion fields $\mathrm{V_R^t}$ and $\mathrm{V_B^t}$.

We define an optimization problem with the objective function:

$$\min_{I_R, I_B, \{V_R^t\}, \{V_B^t\}} \sum_t || I^{\mathrm{t}} - \mathrm{W(V_R^t)I_R} + \mathrm{W(V_B^t)\, I_B} ||$$

Algorithm

The original algorithm consists of two steps: **Initialization** and **Iterative optimization**

1. Initialization

- Edge map: Canny edge detector
- Sparse motion field: Solving the equation

$$\min_{V} \sum_{x \in Edge(I^1)} NCC\left(I^1(x), I^2(x+V(x))\right) + \sum_{x,x' \in Edge(I^1) and (x,x') \in N} S(V(x), V(x'))$$

- Classfication: separate sparse motion field V into two layers by fitting two perspective transforms to the edge motion and assign each pixel to either the background layer or the object-reflected layer using RANSAC.

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping
6. Optimization

Algorithm

- Dense motion field: Interpolate dense flow fields from sparse edge flows for both background and reflection.

- Warping: Warping to the reference frame using the dense flow field. We take the initial estimation of the background image to be the minimum intensity across the warped frames.

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping
6. Optimization

Algorithm

2. Optimization

Use an alternating **gradient descent method** to solve the objective function. That is, alternate between the decomposition and motion estimation until convergence.

Algorithm Motion Approach

Input: sequence of images $\{I^t\}_t$; initial guess of $I_O, I_B, A, \{V_O^t\}$ and $\{V_B^t\}$
  for *Scale* $s = 1$ **to** n_s **do**
    $\{\widehat{I^t}\} \leftarrow$ downsampling input image sequence $\{I^t\}$ to scale s
    $I_O, I_B, A, \{V_O^t\}, \{V_B^t\} \leftarrow$ downsampling/upsample I_O, I_B
    $A, \{V_O^t\}$, and $\{V_B^t\}$ to scale s
    for $i = 1$ **to** n_s **do**
      $I_O, I_B, A \leftarrow Decompose(\{\widehat{I^t}\}, \{V_O^t\}, \{V_B^t\})$
      $\{V_O^t\}, \{V_B^t\} \leftarrow EstimateMotion(\{\widehat{I^t}\}, I_O, I_B, A)$
    end for
  end while
Output: $I_O, I_B, A, \{V_O^t\}$ and $\{V_B^t\}$

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping
6. Optimization

Implementation

Input image sequence:

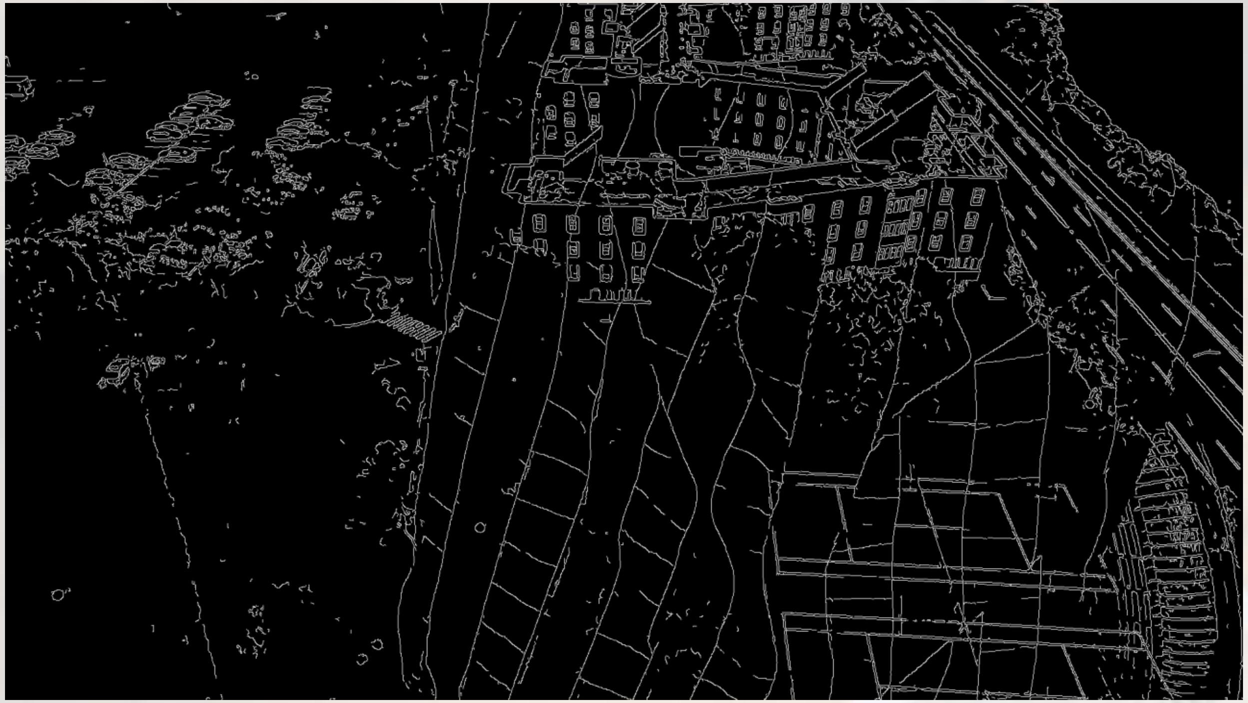
1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping

Implementation

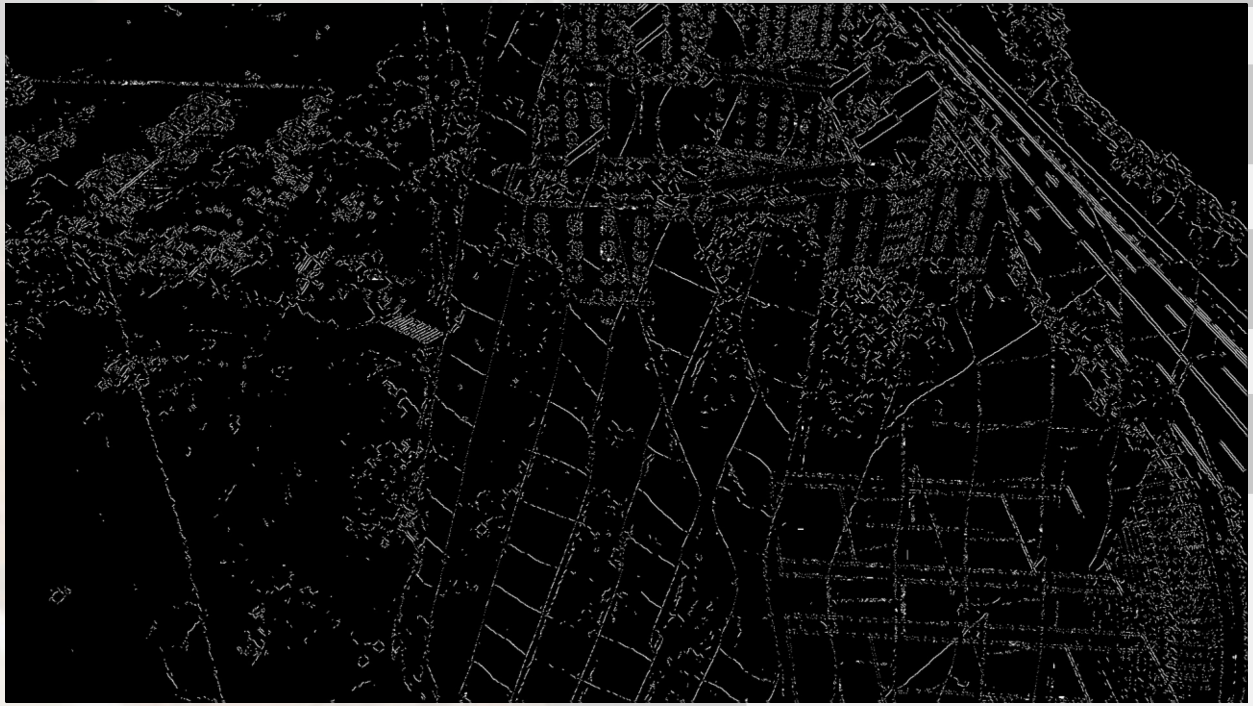
1. Edge Map

2. Sparse Motion Field

1. Create edge map by canny edge detection



opencv Canny edge detection



My Canny edge detection implementation

Implementation

2. Extract sparse motion field:

- original solution in the paper:

$$\min_{V} \sum_{x \in Edge(I^1)} NCC\left(I^1(x), I^2\left(x + V(x)\right)\right) + \sum_{x,x' \in Edge(I^1) and (x,x') \in N} S\left(V(x), V(x')\right)$$

- my solution:
  use **"Lucas-Kanade Optical Flow algorithm"** to find motion field.

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping

Implementation

Lucas–Kanade method

the local image flow (velocity) vector (u, v) must satisfy:

$$I_x(p_1)u + I_y(p_1)v = -I_t(p_1)$$

$$I_x(p_2)u + I_y(p_2)v = -I_t(p_2)$$

$$\vdots$$

$$I_x(p_{25})u + I_y(p_{25})v = -I_t(p_{25})$$

where $p_1, p_2, \ldots p_{25}$ are the pixels inside the window,

$I_x(p_i) = \frac{\partial I}{\partial x}$ partial derivatives of the image I wrt position x

$I_y(p_i) = \frac{\partial I}{\partial y}$ partial derivatives of the image I wrt y

$I_t(p_i) = \frac{\partial I}{\partial t}$ partial derivatives of the image I wrt t

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping

$$I_x u + I_y v + I_t = 0$$

$$\mathrm{u} = \frac{\partial \mathrm{x}}{\partial \mathrm{t}} \quad \mathrm{v} = \frac{\partial \mathrm{y}}{\partial \mathrm{t}}$$

Implementation

Write the equations in matrix form $Av = b$

$$A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \quad v = \begin{bmatrix} \mathrm{u} \\ \mathrm{v} \end{bmatrix} \quad b = \begin{bmatrix} -I_t(p) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_{25}) \end{bmatrix}$$

Solve the equation by least squares approximation

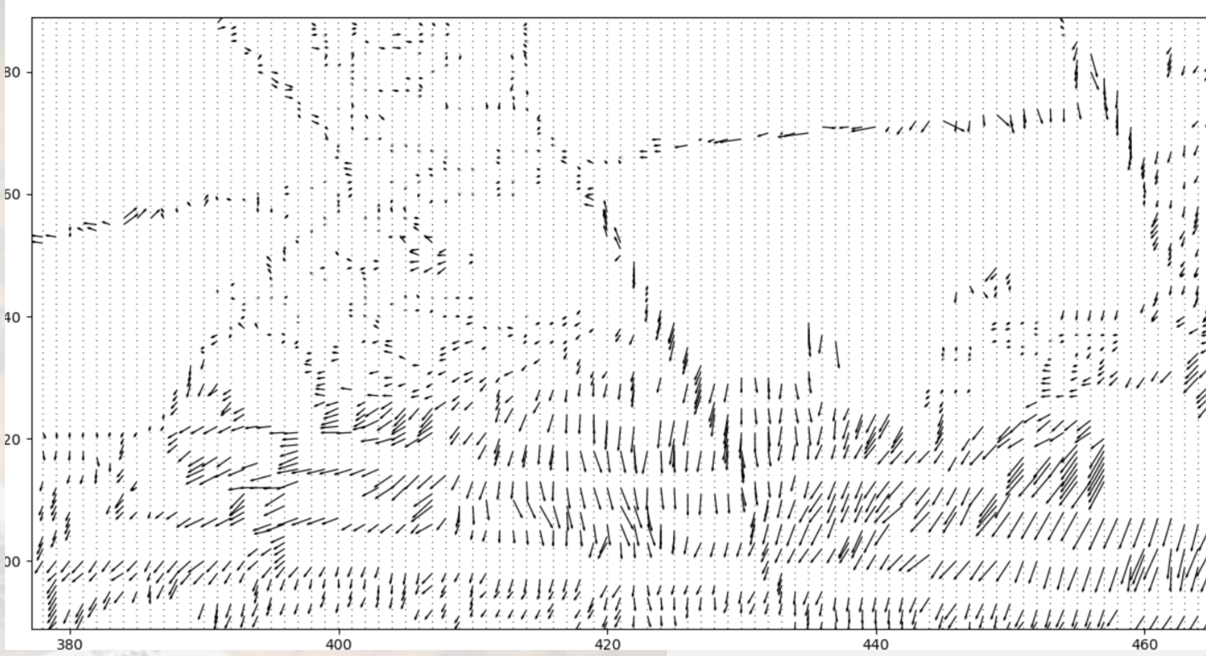
$\hat{x} = argmin_x ||Ax - b||^2$ equivalent to solving $A^\top A\hat{x} = A^\top b$

To obtain $I_x(p_i), I_y(p_i), I_t(p_i)$, use pixel domain convolution

$$I_x(p_i) = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \otimes I_t \; ; \quad I_y(p_i) = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \otimes I_t,$$

$$I_t(p_i) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \otimes I_t - \begin{bmatrix} -1 & -1 \\ -1 & -1 \end{bmatrix} \otimes I_{t-1}$$

1. Edge Map
2. Sparse Motion Field
3. Classification

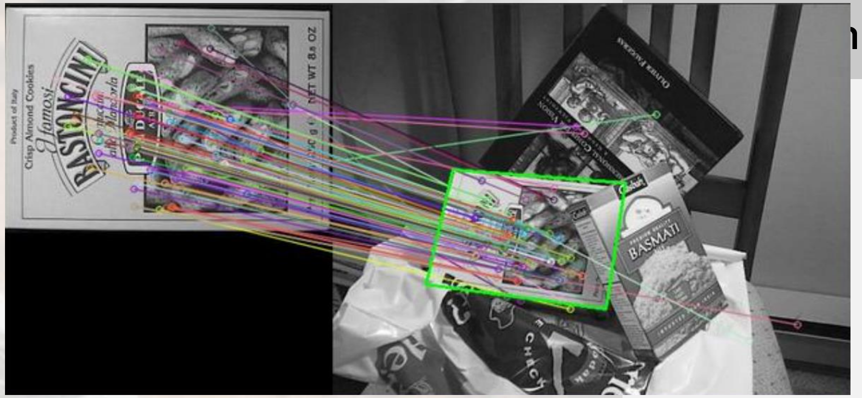


Implementation

3. classfication:

- original solution in the paper:
  separate sparse motion field V into two layers by fitting two perspective transforms to the edge motion and assign each pixel to either the background layer or the object-reflected layer using RANSAC.

- my implementation:
  use cv2.findHomography() with RANSAC method



1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
ng

Implementation

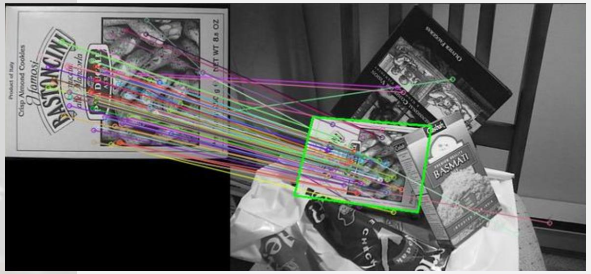
- Treat the points in edge map as first planes, the new points moved according to their corresponding motion vectors as the second planes.

- To separate the original sparse motion points into background motion points and reflection motion points, I call `cv2.findHomography()` twice:

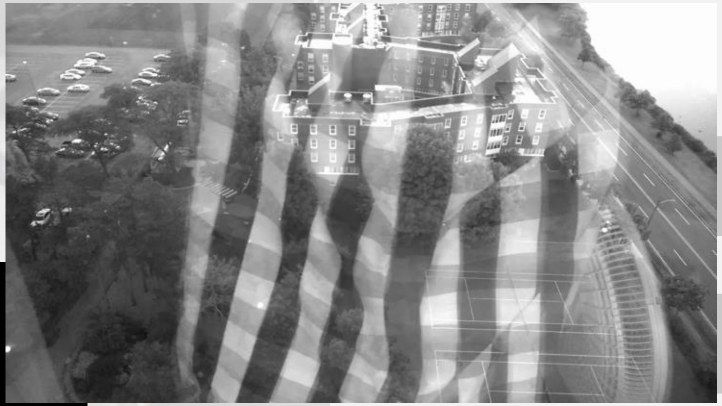
(1) input: the original motion points
get: use the output inlier points as background motion points.
(assume background dominate)

(2) input: the oulier points from the output of the first call
get: output inlier points as reflection motion points.

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping



Implementation



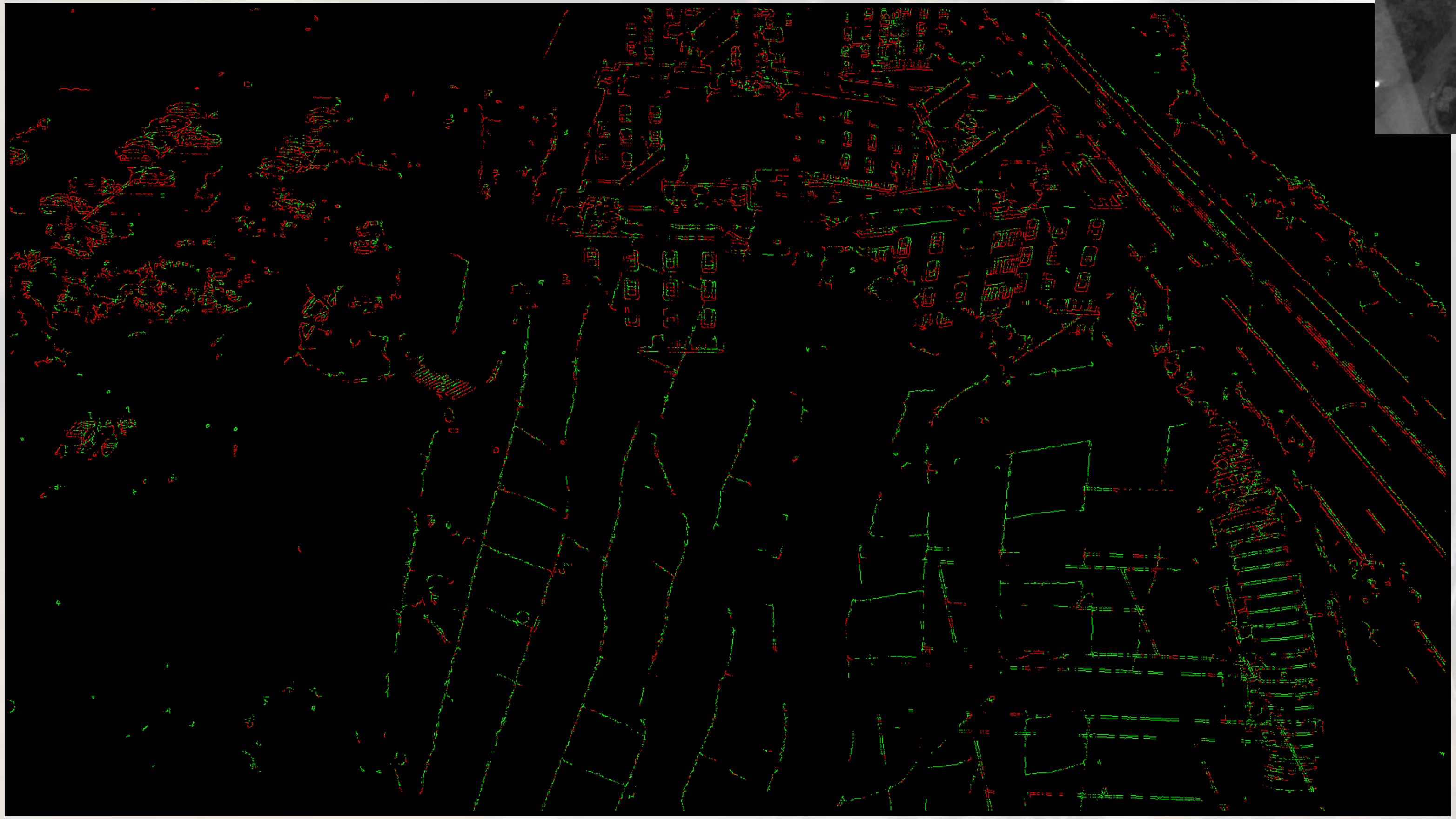
Field

3. Classification

4. Dense Motion Field

5. Warping

6. Optimization



Classified sparse motion field
(motion calculate from frame #3-4, red points are background motion, green points are reflection motion)

Implementation

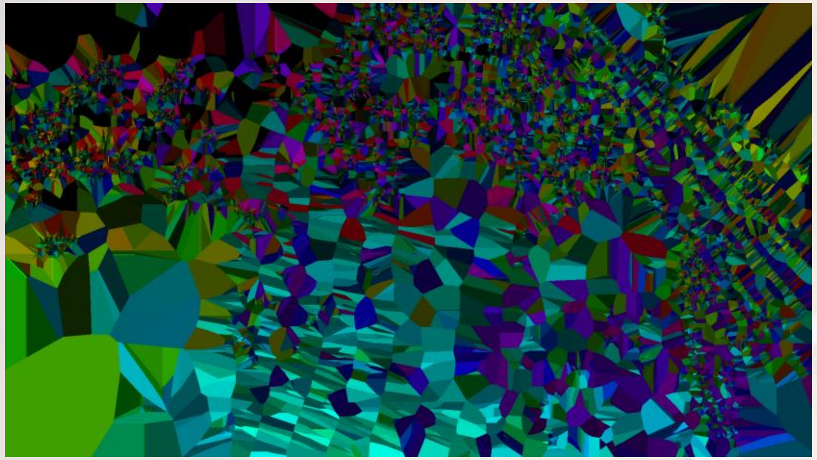
4. Dense motion field

- interpolate dense flow fields from sparse edge flows for both background and reflection motion.
- My implementation:

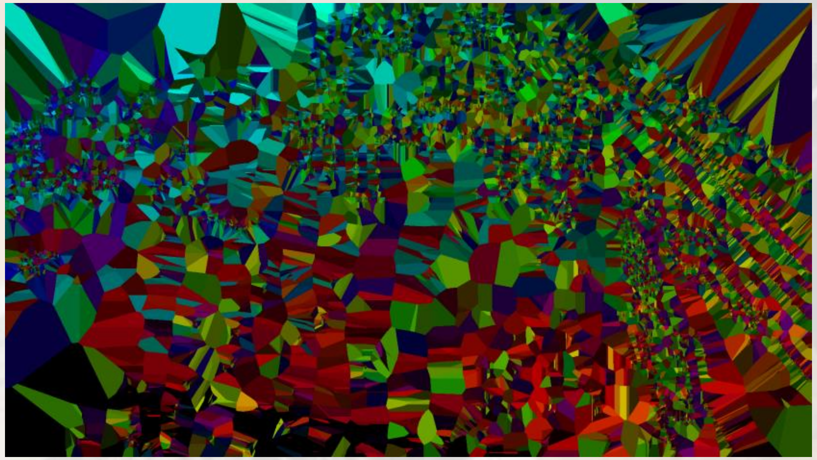
  use scipy.interpolate.griddata() to interpolate for both y-axis motion vector and x-axis motion vector.

Motion direction: 0 deg — *365 deg*

Magnitude: low — *high*



dense motion field of reflection



dense motion field of background

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping

Implementation

5. Warping

- Warping to the reference frame using the dense flow field.

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping

Implementation

1. Edge Map
2. Sparse Motion Field
3. Classification

ield

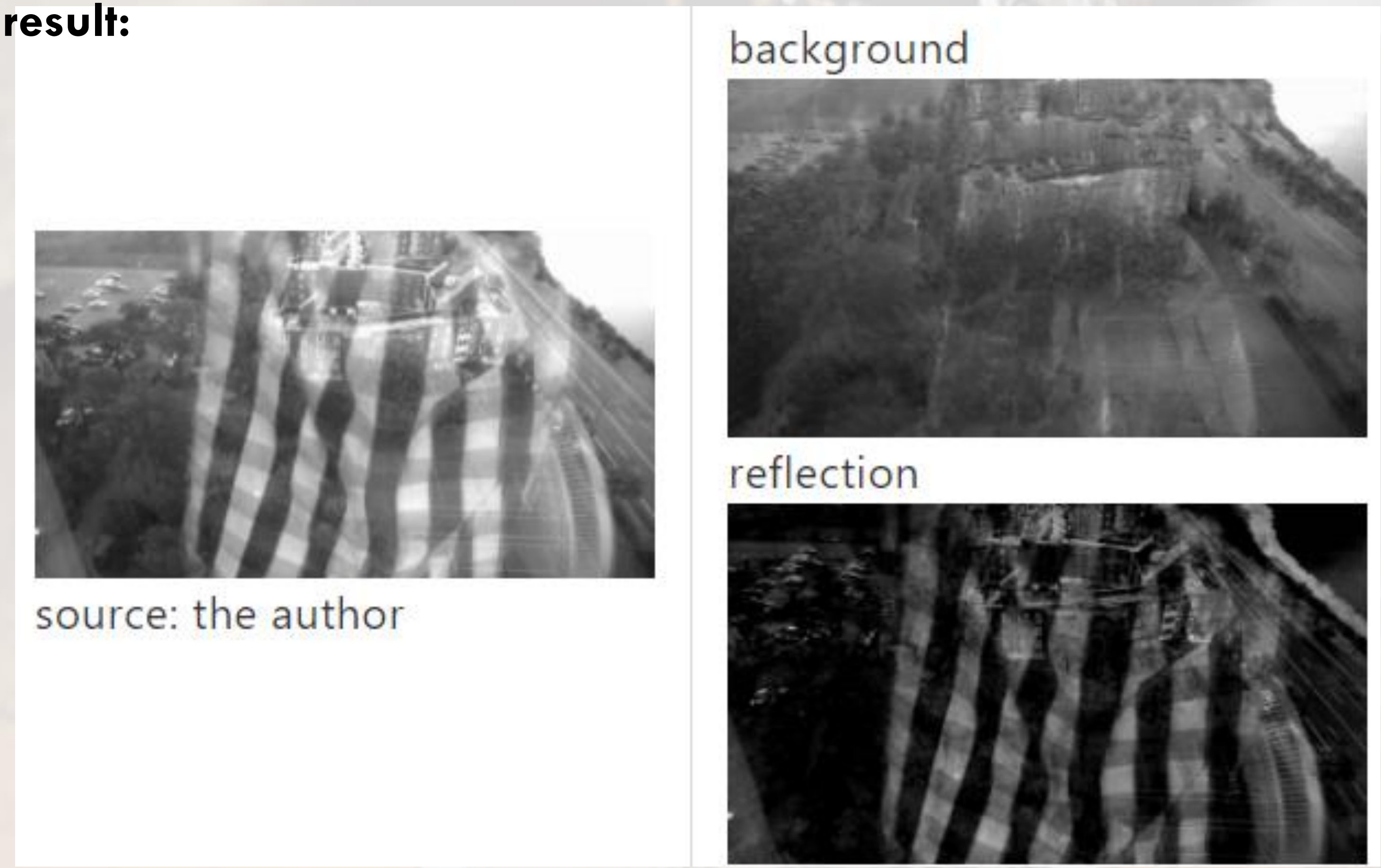
Result:

I_R

I_B

Implementation

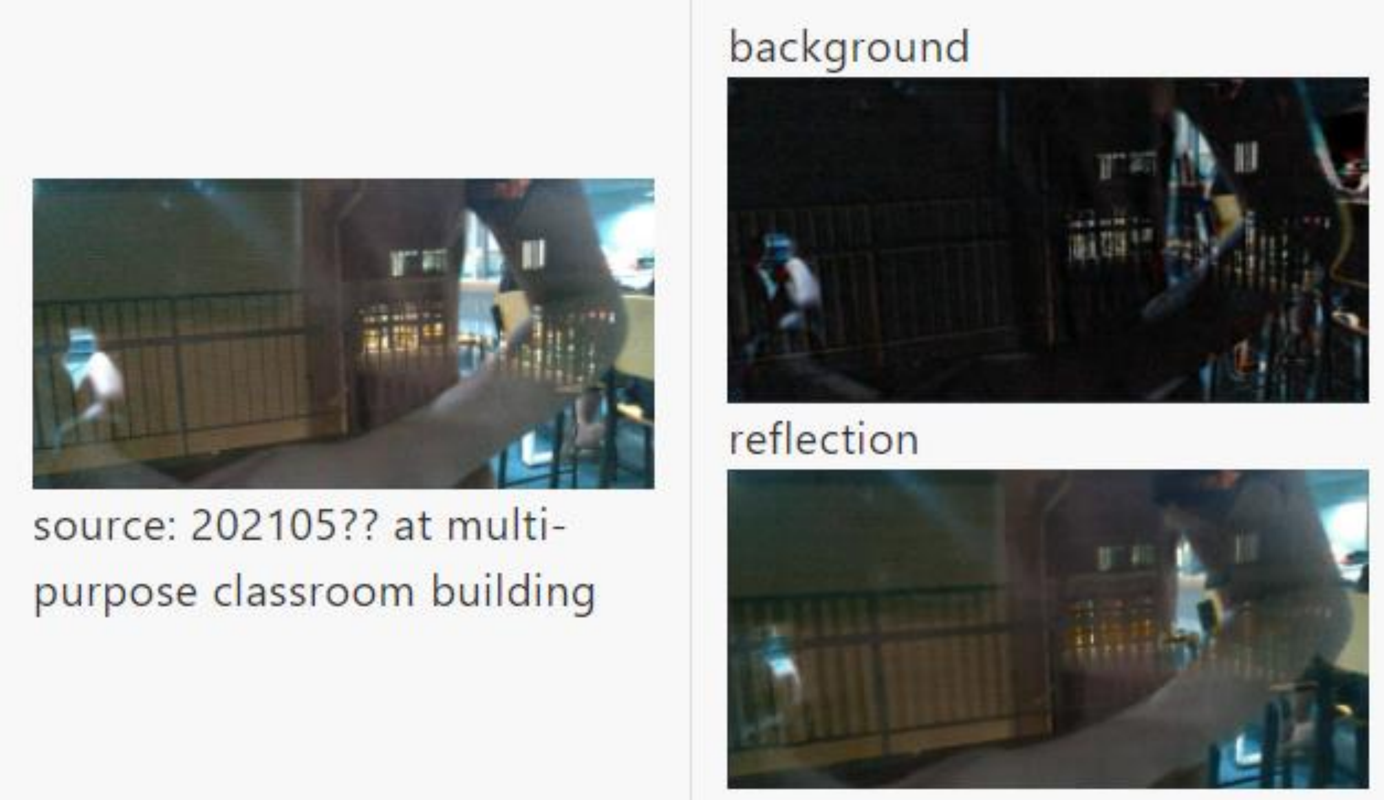
Other result:



- ➢ The result is good because the black-white plaid shirt acts as a good edge information.

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping

Implementation



- Most of the light outside of the windows are classified correctly. However, because the unfocused reflection image is too blurred to act correctly as an edge. Therefore, some pixels near the arm aren’t classified correctly.

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping

Implementation

- The result isn't very well because the movement of sequence is too subtle to cover every "background pixel" that is blocked by the lightbulb.

1. Edge Map
2. Sparse Motion Field
3. Classification
4. Dense Motion Field
5. Warping

User-assisted Separation with Sparse Prior

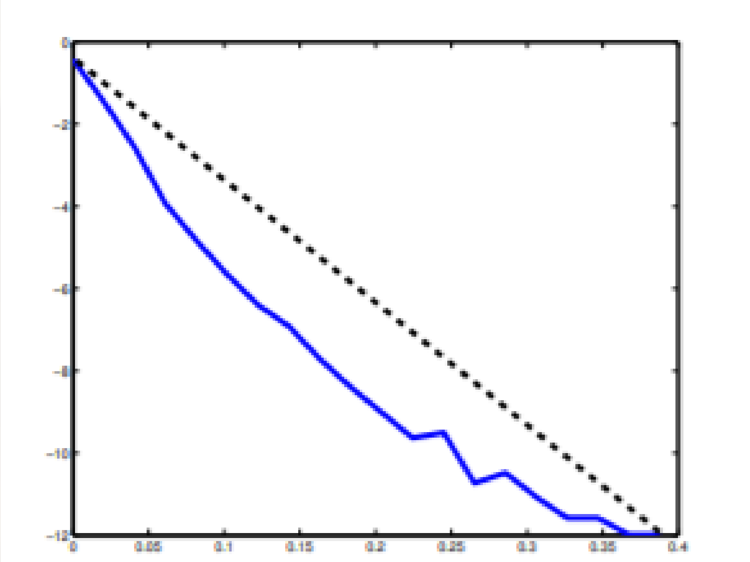
LEVIN, Anat; WEISS, Yair. User assisted separation of reflections from a single image using a sparsity prior. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29.9: 1647-1654.

We use the assumption that the user can mark where the edge of the reflection or the transmitted image is.

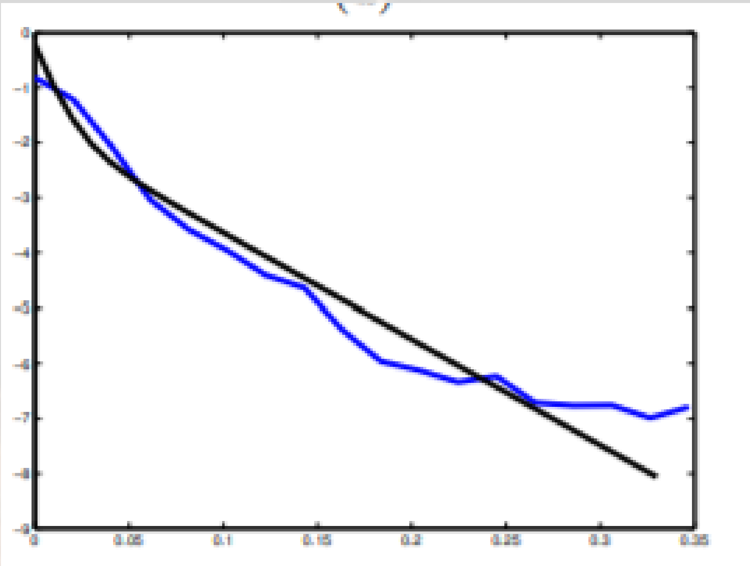
Original Image

User-marked Image

The method is based on the fact that, the gradient of a natural image is sparse in the sense that the distribution peak at zero and have heavy tails, so we can model the distribution by some sparse prior.



Sparse Distribution



Non-sparse Distribution

Now we state the problem formally.

Assume we are given an input image I with two sets of image locations S_R, S_B, such that the gradients in location S_R belong to one layer and gradients in location S_B belong to the other layer. We then wish to find the two layers I_R, I_B such that:

1. Sum of two layers form the input image $I = I_R + I_B$.

2. Gradients of I_R at locations in S_R agree with the gradients of I, and similarly the gradients of I_B at locations in S_B agree with the gradients of I.

On these constraints, we aim to maximize the probability of the two layers

$$\mathcal{P}(I_R, I_B) = \mathcal{P}(I_R)\mathcal{P}(I_B) = \prod_{i,k} \mathcal{P}\left(f_{i,k} \cdot I_R\right) \cdot \mathcal{P}(f_{i,k} \cdot I_B)$$

We will use the Laplacian prior to approximate the probability function by

$$\log \mathcal{P}(I) = \sum_{i,k} \rho\left(f_{i,k} \cdot I\right) \qquad \rho(x) = \log\left(\frac{\pi_1}{2s_1} e^{-\frac{|x|}{s_1}} + \frac{\pi_2}{2s_2} e^{-\frac{|x|}{s_2}}\right)$$

$f_{i,k} \cdot I$ is the convolution of the k'th filter with image I centered at pixel i.

The filters include the first and second order derivatives, as its selection is of our interests.

To sum all, we aim to minimize the cost

$$J(I_R, I_B) = \sum_{i,k} \rho\left(f_{i,k} \cdot I_R\right) + \rho(f_{i,k} \cdot I_B)$$

One can write it into the form

$$J(v) = \sum_{j} \rho_j(A_j v - b_j)$$

where v is the image vector, A has rows that correspond to the filters, and b is the derivatives.

We use the method of Iterative reweighted least squares optimization(IRLS) to approximate the solution.

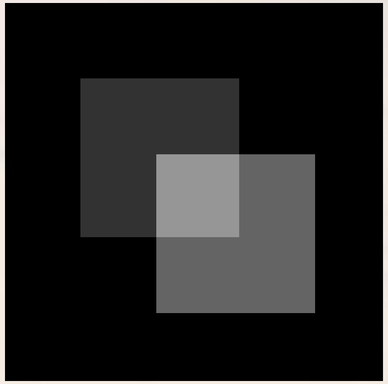
- Initialization: Set initial state $\psi_j^0 = 1$ for all j indexed from 1 to image size.
- Repeat:
    1. Let $\bar{A} = \sum_j A_j^T \psi_j^{t-1} A_j$ and $\bar{b} = \sum_j A_j^T \psi_j^{t-1} b_j$. Let x^t be the solution of $\bar{A}x = \bar{b}$
    2. Set $u_j = A_j x^t - b_j$, and then $\psi_j^t(u_j) = \frac{1}{|u_j|}\rho'(u_j)$.

$$\rho'(u_j) \sim (\frac{\pi_1}{2s_1^2} e^{-\frac{|x|}{s_1}} + \frac{\pi_2}{2s_2^2} e^{-\frac{|x|}{s_2}}) / ((\frac{\pi_1}{2s_1} e^{-\frac{|x|}{s_1}} + \frac{\pi_2}{2s_2} e^{-\frac{|x|}{s_2}})$$

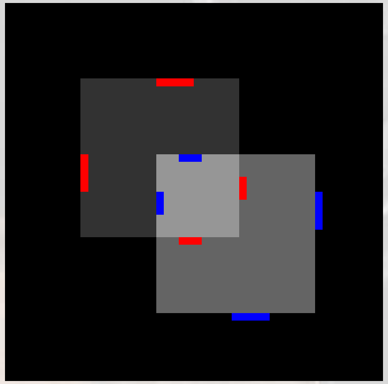
The experimented result made by python code consumes lots of memory overhead.

Regular images, e.g. size 300×300, will crash the process.

For a simple test image (size 50×50) as below:



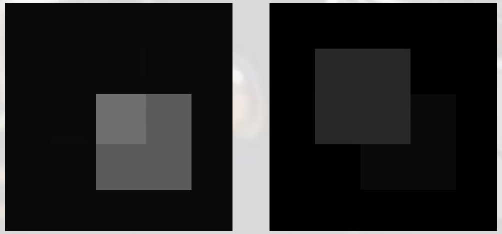
Test Image



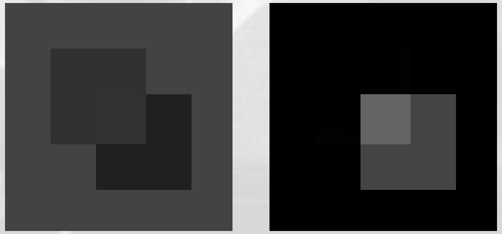
Test Image with User-assisted edges

Fix repeat time to 10 times:

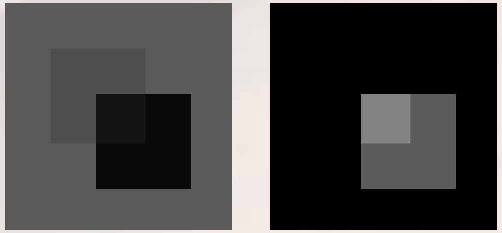
Iteration 1



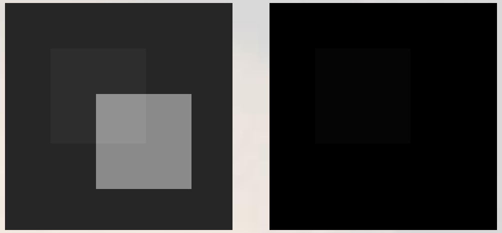
Iteration 2



Iteration 3

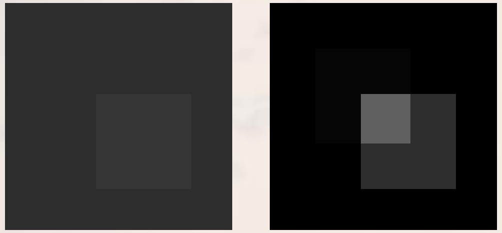


Iteration 4



Iteration 5

Iteration 6



Iteration 7

Iteration 8

Iteration 9

For 10 times iterations, the program can separate the reflection to some extent.

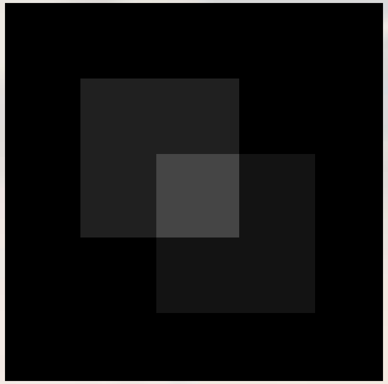


Image I_R

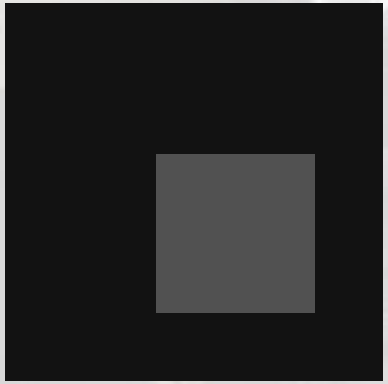


Image I_B

- The program seem not to converge as iteration time increases.
- The result takes 7 minutes on CSIE Workstation.

Reference

[1] LI, Yu; BROWN, Michael S. Single image layer separation using relative smoothness. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014. p. 2752-2759.

[2] XUE, Tianfan, et al. A computational approach for obstruction-free photography. ACM Transactions on Graphics (TOG), 2015, 34.4: 1-11.

[3] LEVIN, Anat; WEISS, Yair. User assisted separation of reflections from a single image using a sparsity prior. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007, 29.9: 1647-1654.