

CSE185

Introduction to Computer Vision

Lab 05: Image Pyramid and
Template Matching

Instructor: Daniel Leung
TA: Mohammadkazem Ebrahimpour
Xueqing Deng

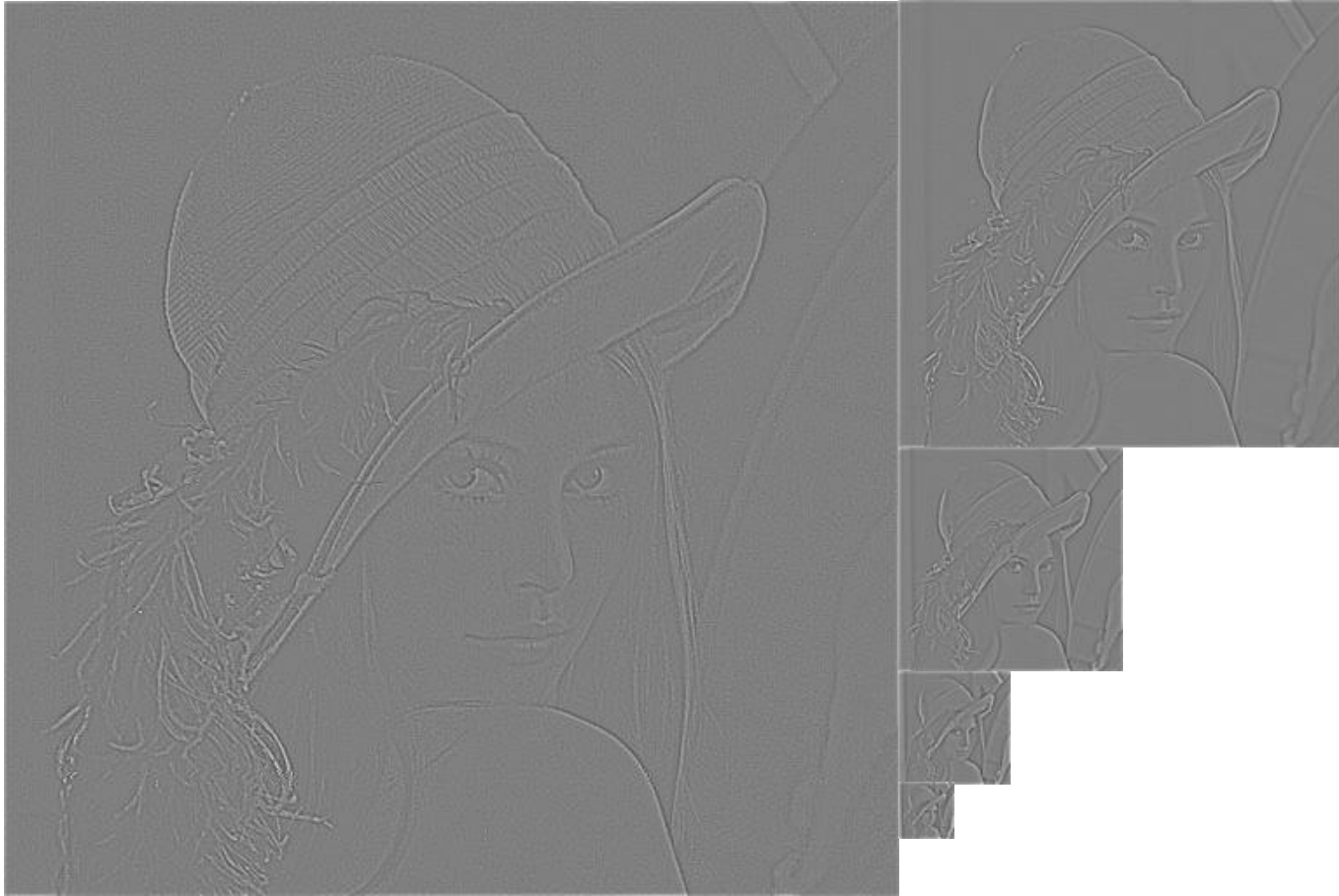
Overview

- Task 1: Gaussian Pyramid



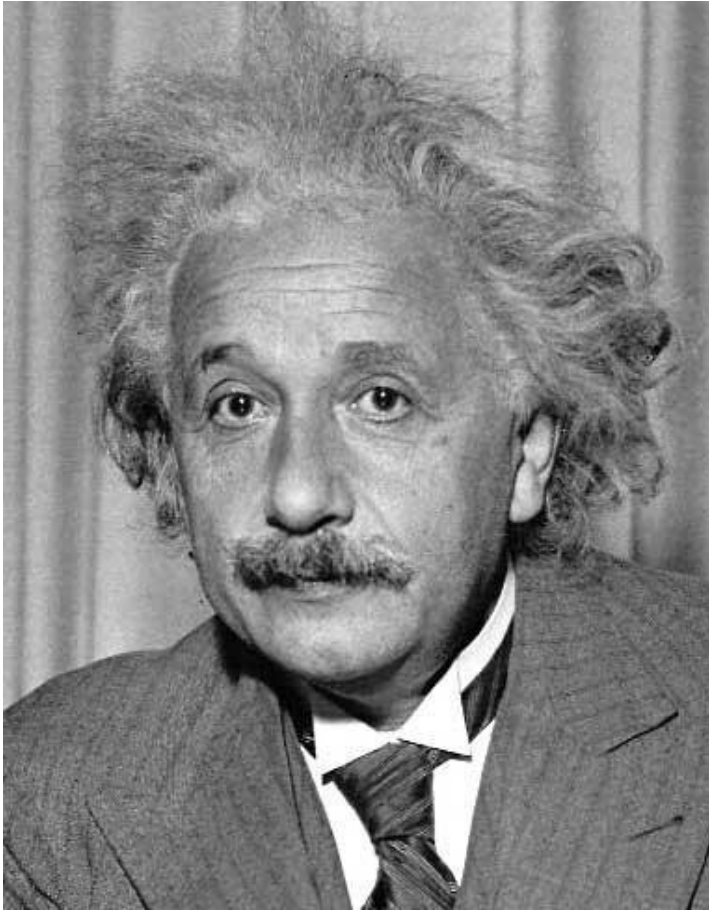
Overview

- Task 1: Laplacian Pyramid



Overview

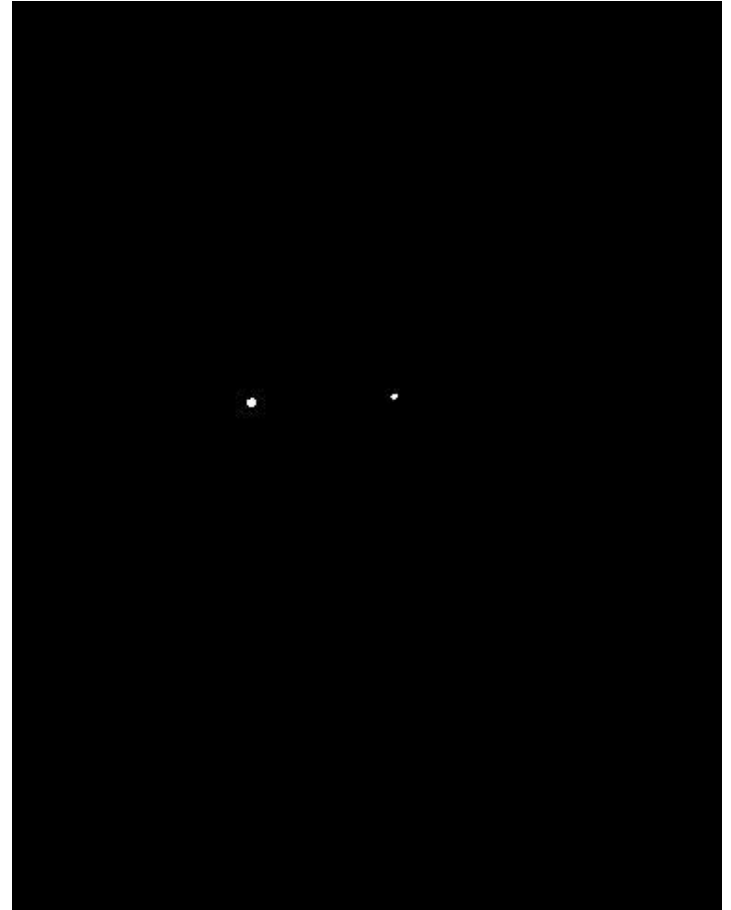
- Task 2: Template Matching



Input Image



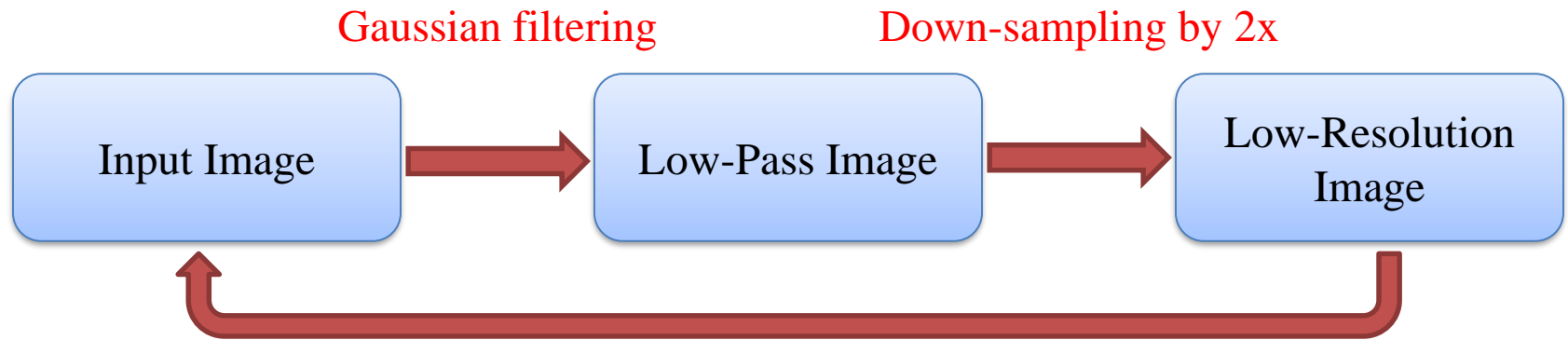
template



Match regions

Gaussian Pyramid

- Given an input image, a Gaussian kernel, construct a Gaussian Pyramid with N scales:



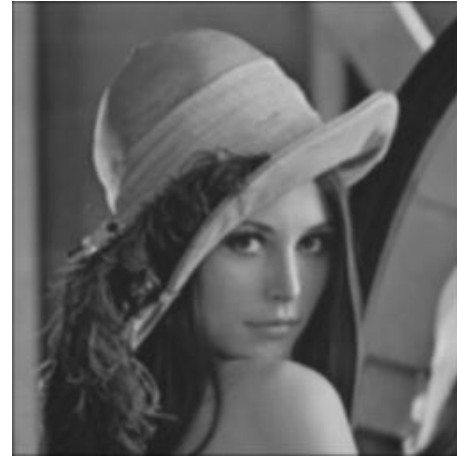
- Gaussian filter: use `imfilter` and `fspecial` or your `Gaussian_filter.m` in lab03.
- Down-sample: use `imresize` or your implementation in lab02.

Gaussian Pyramid

scale = 1



scale = 2



scale = 3



scale = 4



scale = 5



Gaussian Pyramid

- In lab05_task1.m:

```
img = im2double(imread('cameraman.jpg'));

sigma = 2.0;
hsize = 7;
scale = 5;

%% Gaussian Pyramid
I = img;
for s = 1 : scale

    % Gaussian filter

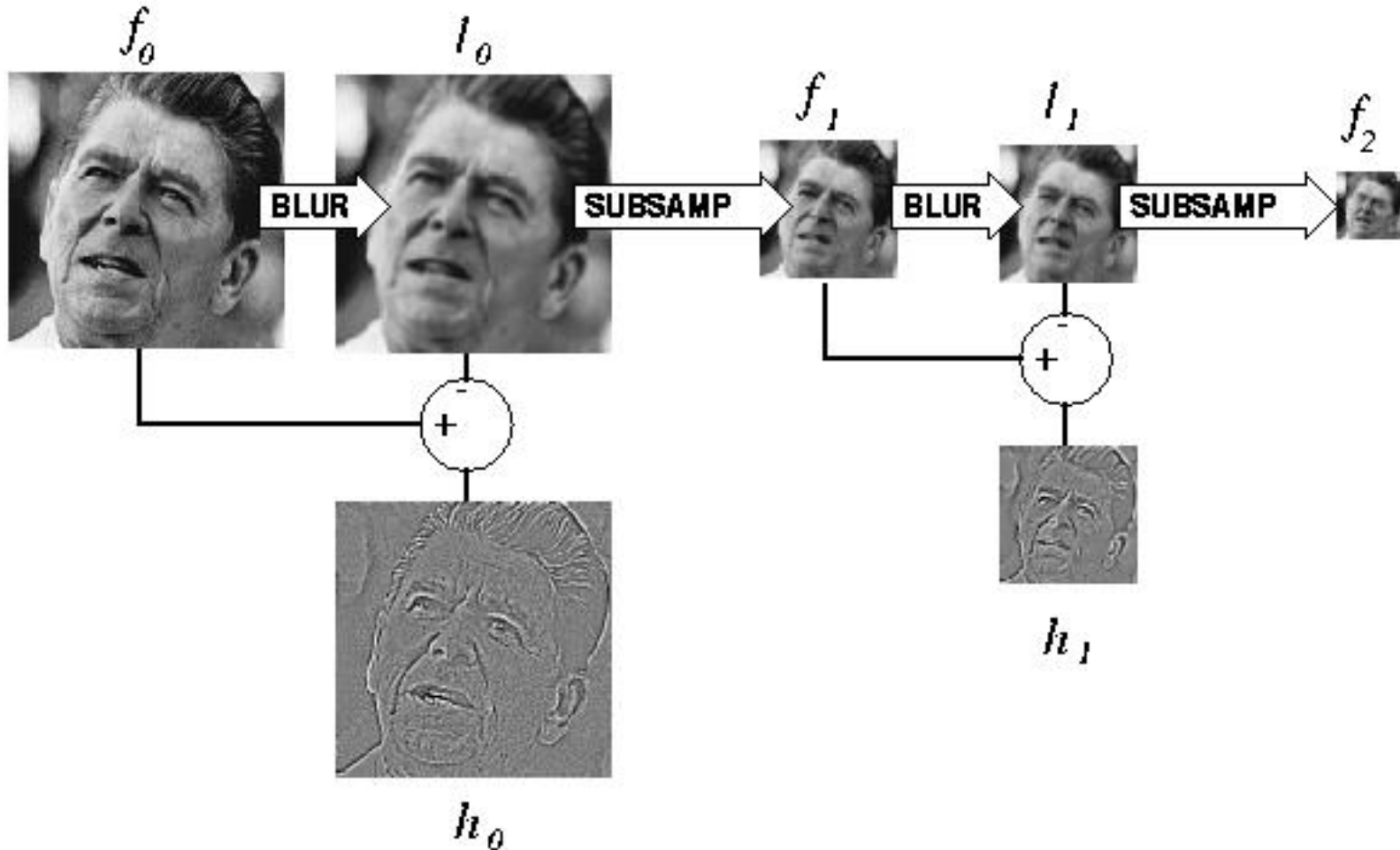
    % Save or show image
    imwrite(I, sprintf('Gaussian_scale%d.jpg', s));

    % Down-sampling

end
```

Laplacian Pyramid

- Laplacian filtering output = Input image – Gaussian filtering output



Laplacian Pyramid

- In lab05_task1.m:

```
%% Laplacian Pyramid
for s = 1 : scale

    % Gaussian filtering

    % Laplacian filtering

    % Save or show image
    imwrite(I + 0.5, sprintf('Laplacian_scale%d.jpg',
s));

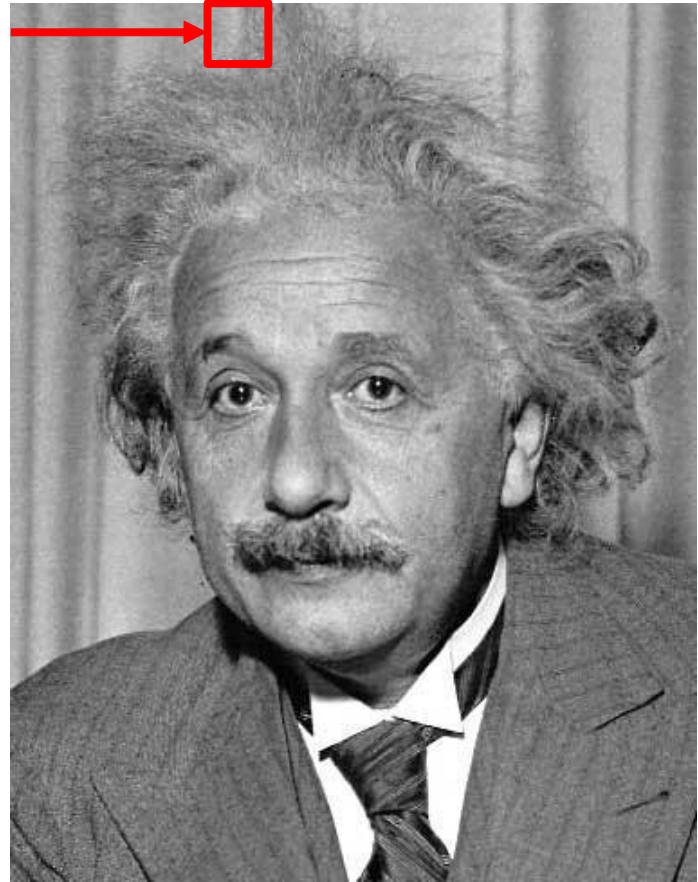
    % Down-sampling

end
```

Add 0.5 to better visualize
negative values

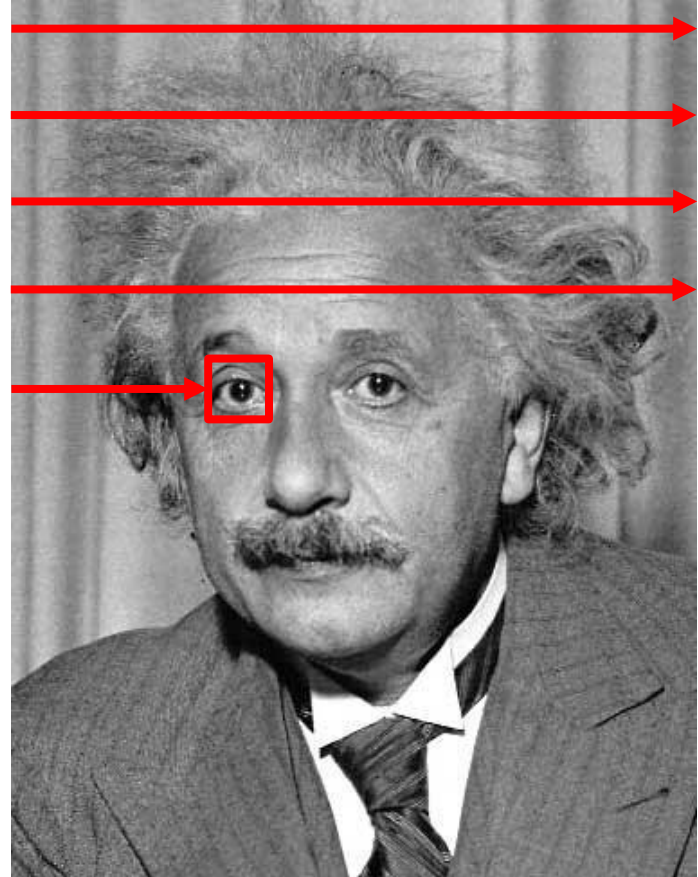
Template Matching

- Goal: given a template (patch) , find matched regions in the input image
- use sliding window:




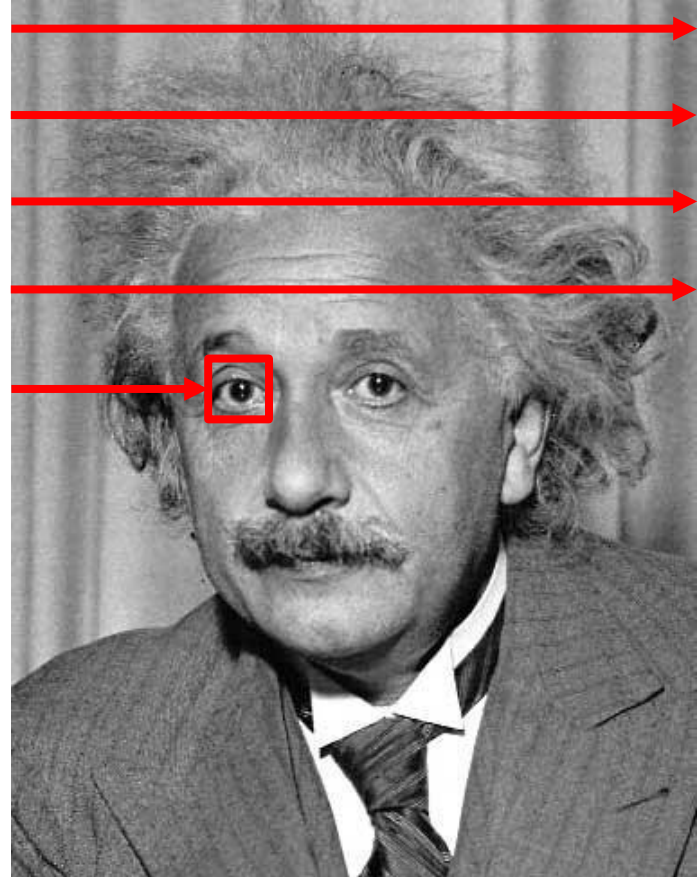
Template Matching

- Goal: given a template (patch) , find matched regions in the input image
- use sliding window:




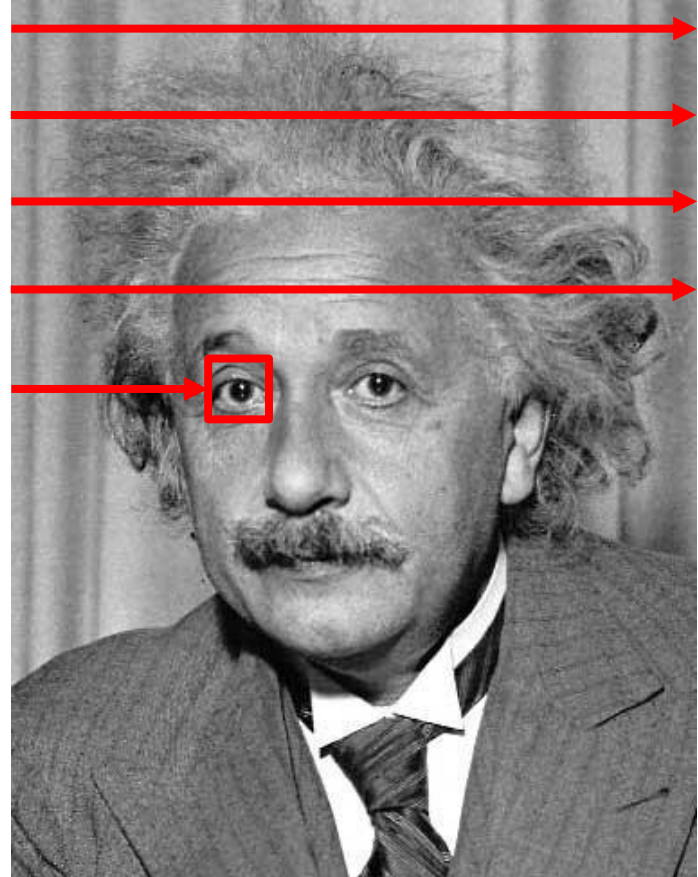
Template Matching

- Goal: given a template (patch) , find matched regions in the input image
- use sliding window:
 - similar to spatial filtering!




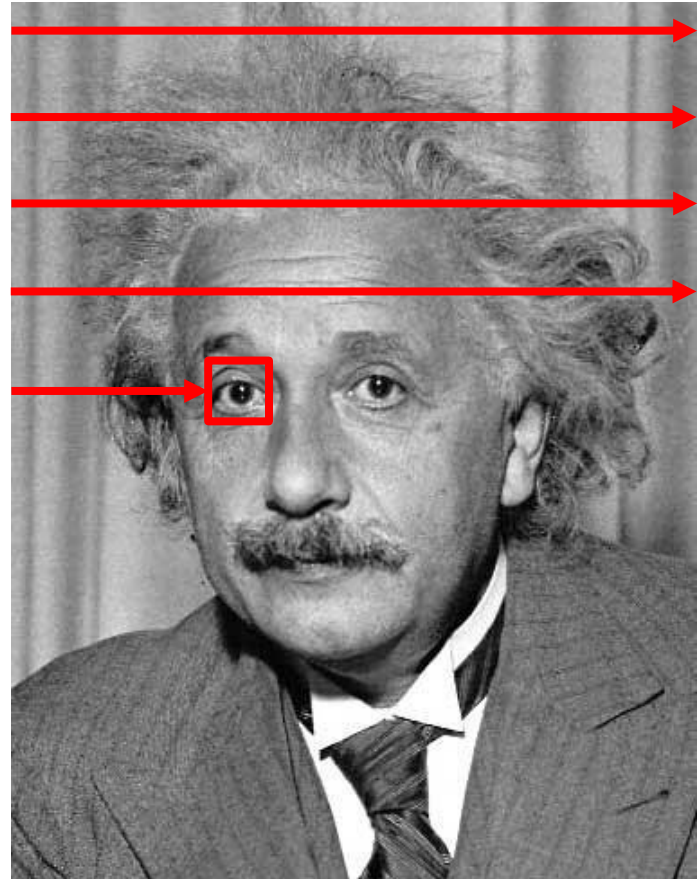
Template Matching

- Goal: given a template (patch) , find matched regions in the input image
- use sliding window:
 - similar to spatial filtering!
- Matching criteria:
 - correlation
 - zero-mean correlation
 - Sum of Square Difference (SSD)
 - Normalized Cross-Correlation (NormCorr)



Template Matching

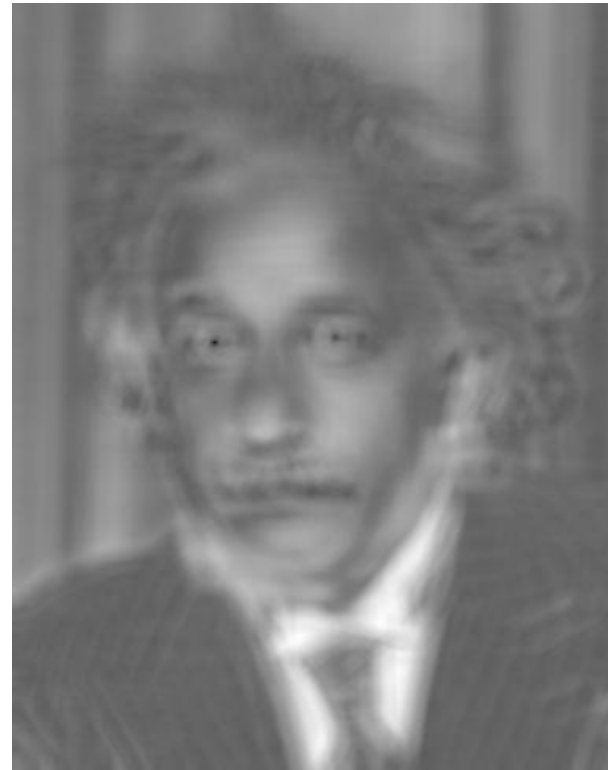
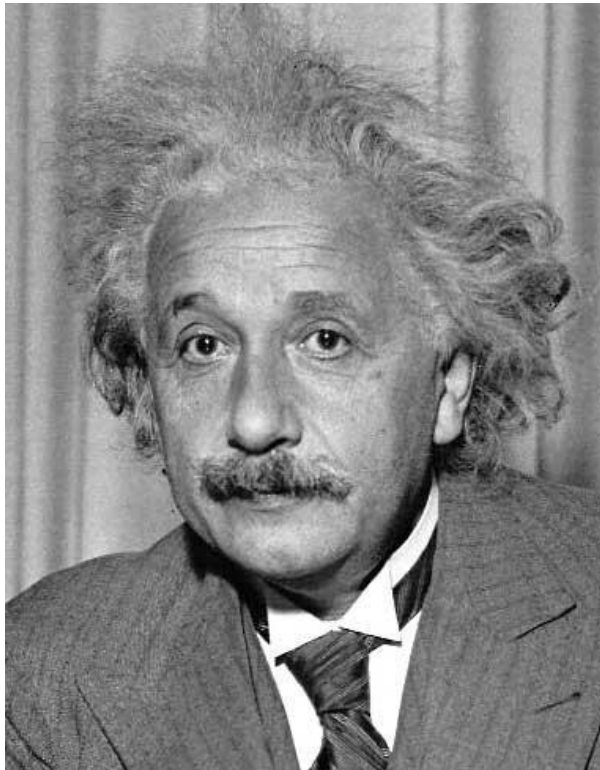
- Goal: given a template (patch) , find matched regions in the input image
- use sliding window:
 - similar to spatial filtering!
- Matching criteria:
 - correlation
 - zero-mean correlation
 - **Sum of Square Difference (SSD)**
 - **Normalized Cross-Correlation (NormCorr)**



Template Matching with SSD

- SSD: calculate the difference between the template and each image patch

$$I'(u, v) = \sum_i \sum_j [I(u + i, v + j) - H(i, j)]^2$$

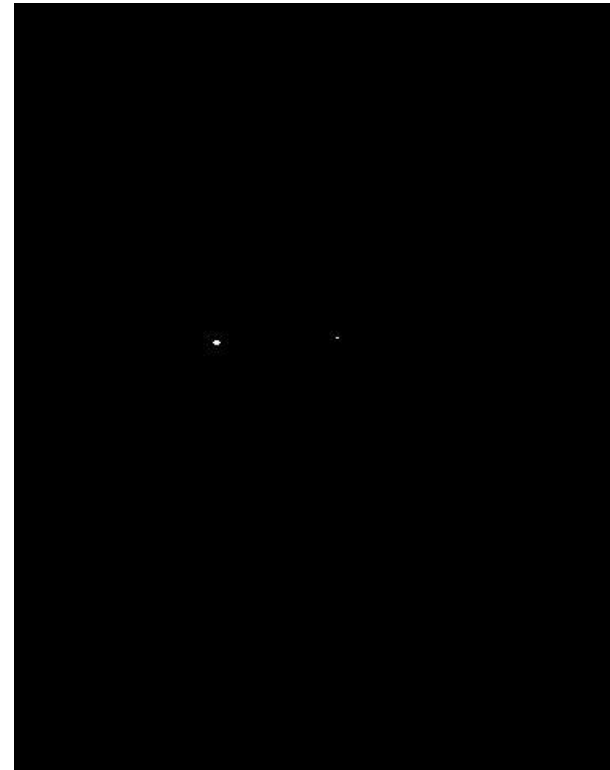
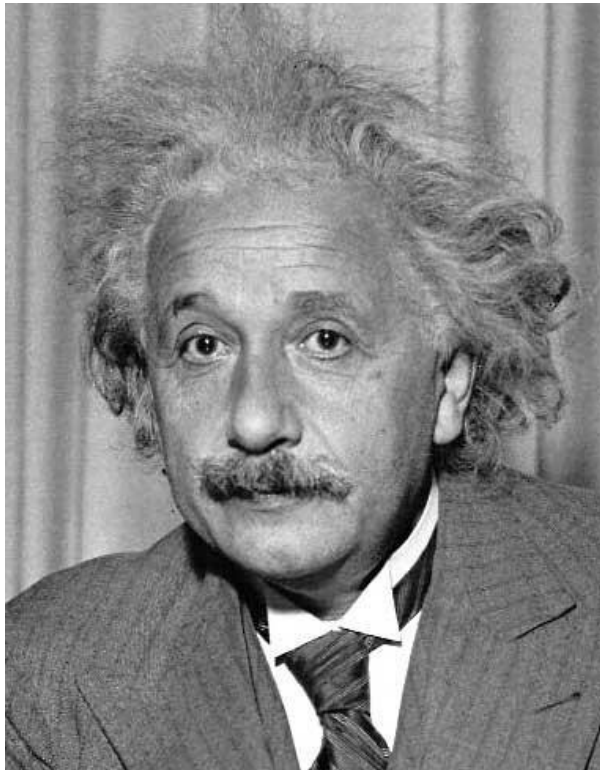


Template Matching with SSD

- SSD: calculate the difference between the template and each image patch

Match Regions = (output < threshold)

threshold = 25



Template Matching with SSD

- In `template_matching_SSD.m`

```
for u = 1 + shift_u : size(I1, 2) - shift_u
    for v = 1 + shift_v : size(I1, 1) - shift_v

        x1 = ???; x2 = ???;
        y1 = ???; y2 = ???;
        patch = I1(y1:y2, x1:x2);

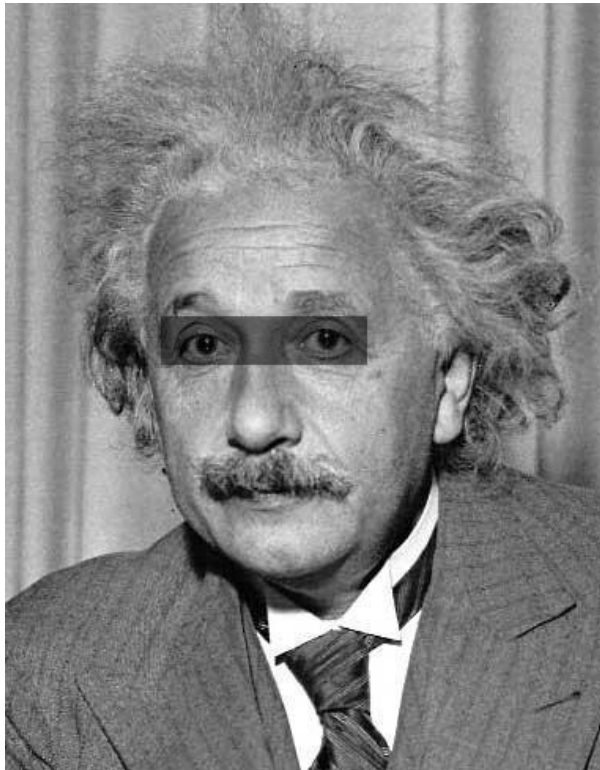
        % SSD
        value = ???;
        output(v, u) = value;

    end
end

match = (output < threshold);
```

Template Matching with SSD

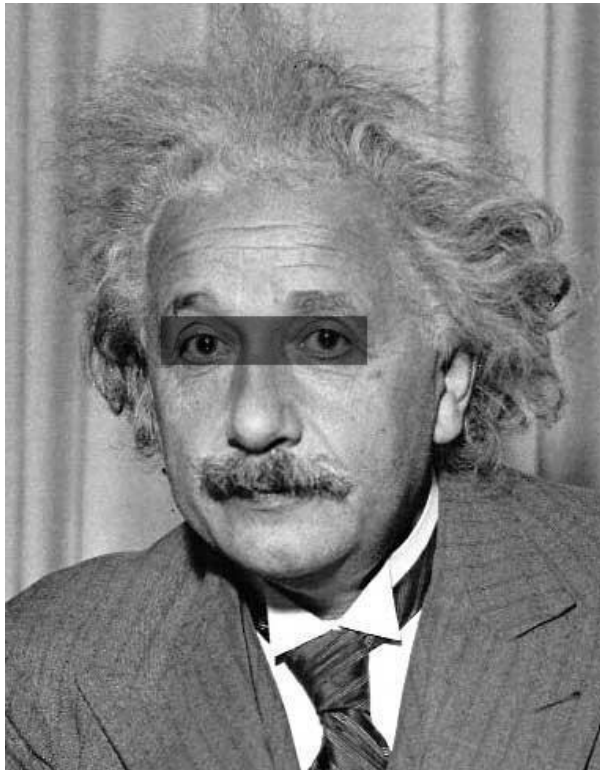
- However, SSD is sensitive to intensity change



Template Matching with SSD

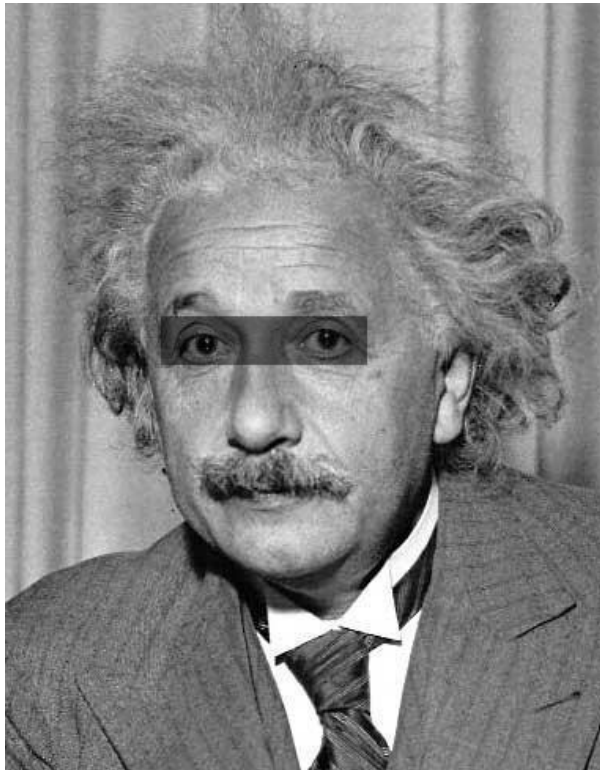
- However, SSD is sensitive to intensity change
 - hard to define the threshold value

threshold = 25



Template Matching with SSD

- However, SSD is sensitive to intensity change
 - hard to define the threshold value



threshold = 36



Normalized Cross-Correlation

- Normalized cross-correlation:
 - assume P_{uv} is a local image patch at (u, v) , H is the template patch

$$\text{normcorr} = \frac{\sum_{i,j} (P_{uv}(i,j) - \bar{P}_{uv})(H(i,j) - \bar{H})}{(\sum_{i,j} (P_{uv}(i,j) - \bar{P}_{uv})^2 \sum_{i,j} (H(i,j) - \bar{H})^2)^{0.5}}$$

$$\begin{aligned}\bar{P}_{uv} &= \text{mean}(P_{uv}) \\ \bar{H} &= \text{mean}(H)\end{aligned}$$

Normalized Cross-Correlation

- Let's simplify it:
 1. convert P_{uv} and H to vectors

Normalized Cross-Correlation

- Let's simplify it:
 1. convert P_{uv} and H to vectors
 2. subtract mean: $P'_{uv} = P_{uv} - \bar{P}_{uv}$ and $h' = H - \bar{H}$

Normalized Cross-Correlation

- Let's simplify it:

1. convert P_{uv} and H to vectors
2. subtract mean: $P'_{uv} = P_{uv} - \bar{P}_{uv}$ and $h' = H - \bar{H}$
3. normalize length to 1: $P''_{uv} = \frac{P'_{uv}}{\|P'_{uv}\|_2}$ and $h'' = \frac{h'}{\|h'\|_2}$

$$normcorr = dot(P''_{uv}, h'')$$

Simple dot product of two
normalized vectors

Normalized Cross-Correlation

- Let's simplify it:

1. convert P_{uv} and H to vectors

2. subtract mean: $P'_{uv} = P_{uv} - \bar{P}_{uv}$ and $h' = H - \bar{H}$

3. normalize length to 1: $P''_{uv} = \frac{P'_{uv}}{\|P'_{uv}\|_2}$ and $h'' = \frac{h'}{\|h'\|_2}$

$$\text{normcorr} = \text{dot}(P''_{uv}, h'')$$

Simple dot product of two normalized vectors

- $\|x\|_2$: the norm (length) of the vector

- $\|x\|_2 = \sqrt{\sum_i x_i^2}$

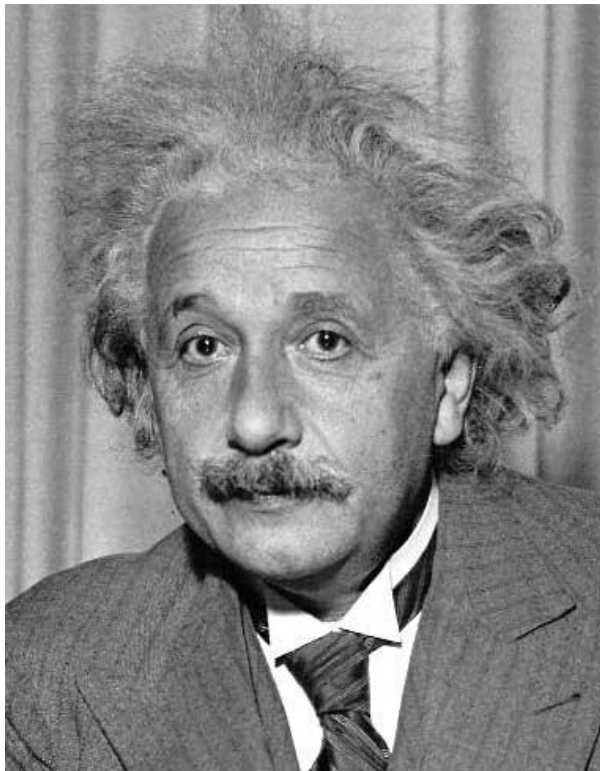
- use `norm(x)` in MATLAB (**x must be a vector!**)

- norm of vector \neq norm of matrix

Normalized Cross-Correlation

- Normalized cross-correlation:

$$\text{normcorr} = \text{dot}(P''_{uv}, h'')$$



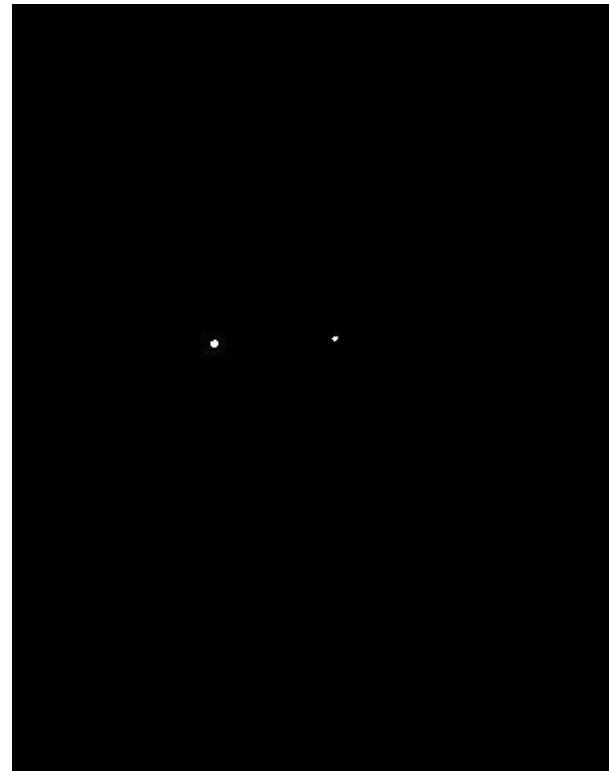
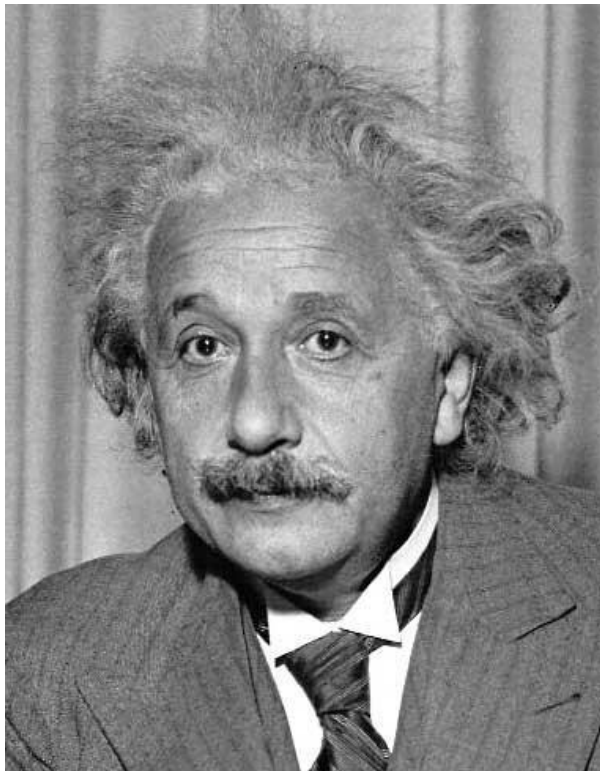
Normalized Cross-Correlation

- Normalized cross-correlation:

$$\text{normcorr} = \text{dot}(P''_{uv}, h'')$$

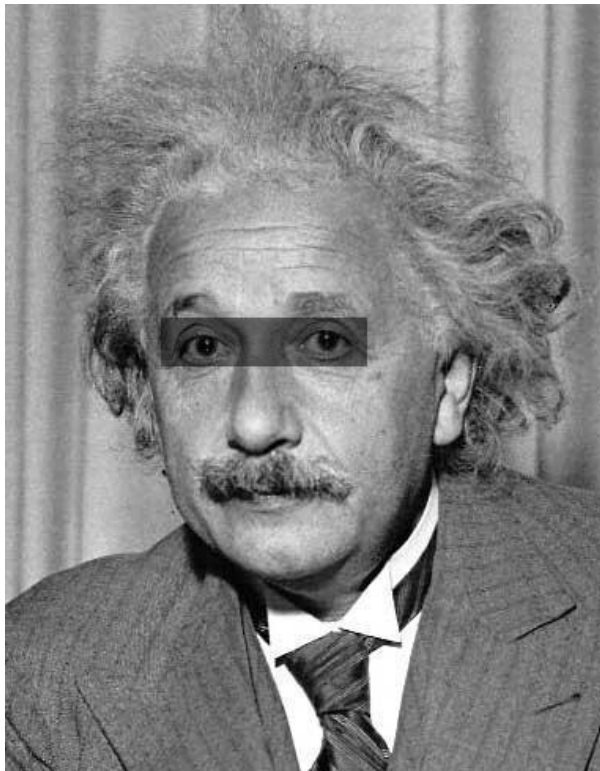
$$\text{Match Regions} = (\text{output} > \text{threshold})$$

$$\text{threshold} = 0.5$$



Normalized Cross-Correlation

- Normalized cross-correlation is invariant to intensity/contrast change

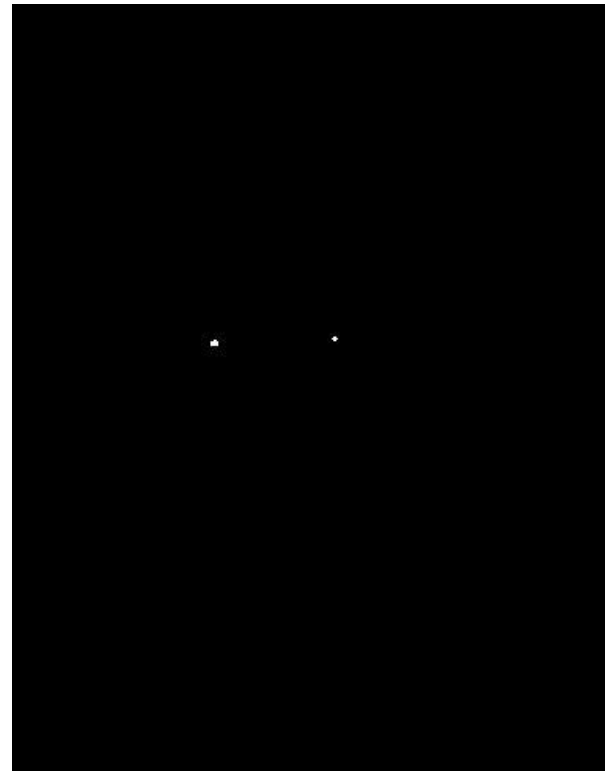
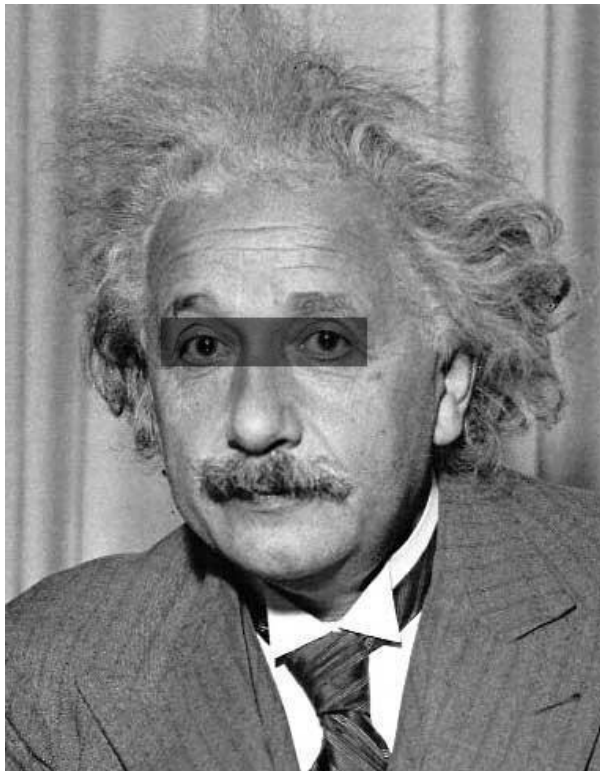


Normalized Cross-Correlation

- Normalized cross-correlation is invariant to intensity/contrast change

Match Regions = (output $>$ threshold)

threshold = 0.5



Template Matching with NormCorr

- In `template_matching_normcorr.m`

```
for u = 1 + shift_u : size(I1, 2) - shift_u
    for v = 1 + shift_v : size(I1, 1) - shift_v

        x1 = ???; x2 = ???;
        y1 = ???; y2 = ???;
        patch = I1(y1:y2, x1:x2);

        % Normalized Cross-Correlation
        value = ???;
        output(v, u) = value;

    end
end

match = (output > threshold);
```

TODO

1. Implement Gaussian Pyramid and Laplacian Pyramid in *lab05_task1.m* (5pt for Gaussian, 5pt for Laplacian)
2. Implement *template_matching_SSD.m*, and try to find the best threshold value for *einstein1.jpg* and *einstein2.jpg*. Use *lab05_task2.m* to test it. (5pt)
3. Implement *template_matching_normcorr.m*, and try to find the best threshold value for *einstein1.jpg* and *einstein2.jpg*. Use *lab05_task2.m* to test it. (5pt)
4. Upload all output images and *lab05_task1.m*, *lab05_task2.m*, *template_matching_SSD.m*, and *template_matching_normcorr.m*