
# Computer Vision

Lecture 6: Matching, Harris Corner Detector

**Dr. Esmaeil Najafi**

**MSc. Javad Khoramdel**

# How to panorama

- Imagine that we have a pair of images, and we want to stick them together to form a panorama image.

- As a human, how would you combine these two images, and why?

# How to panorama

- As a human, how would you combine these two images, and why?
  - We find “**unique patches**” in these two images, …

# How to panorama

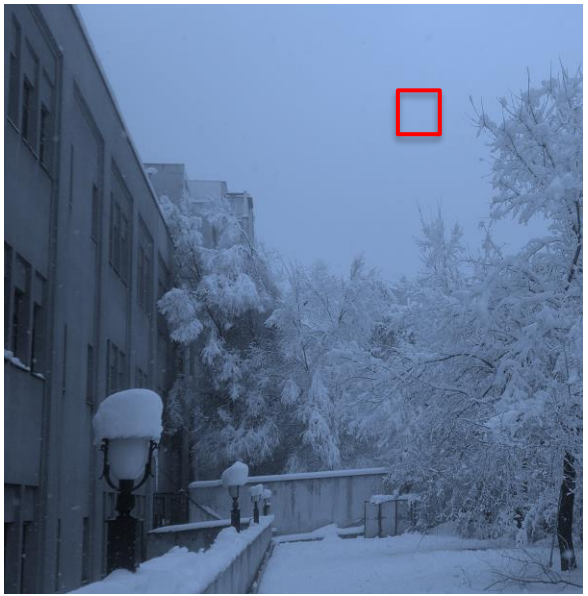
- As a human, how would you combine these two images, and why?
  - We find "**unique patches**" in these two images, "**match**" them based on their similarities, …

# How to panorama

- As a human, how would you combine these two images, and why?
  - We find “**unique patches**” in these two images, “**match”** them based on their similarities, then “**transform**” the images to form the panorama image.

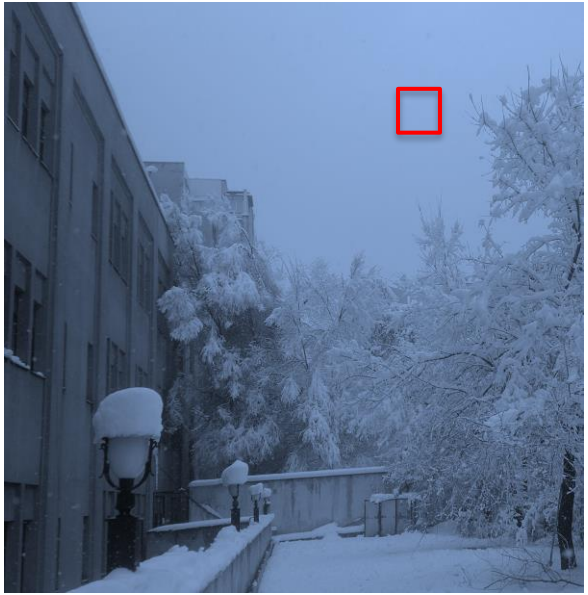
# How can we find unique patches?

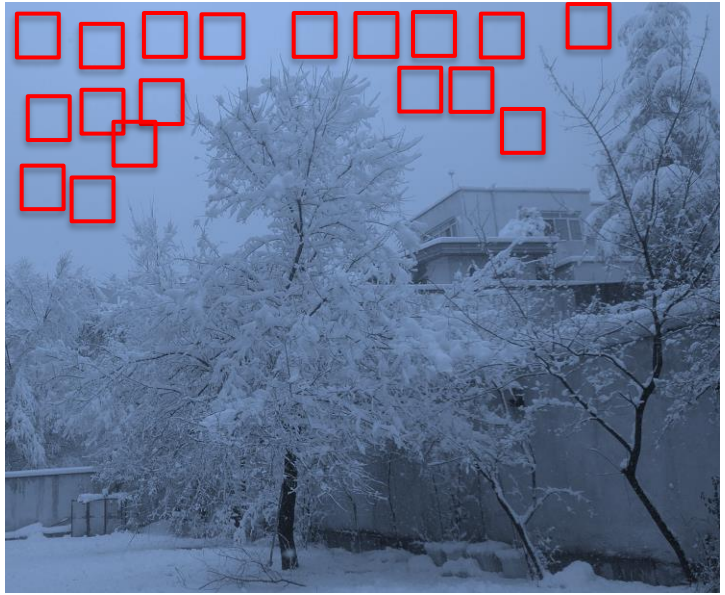
- Why didn’t we choose this patch in the first image?



# How can we find unique patches?

- Why didn’t we choose this patch in the first image?
  - This patch is not unique; it would match lots of irrelevant patches.
  - Very little variation.





# How can we find unique patches?

- Why didn’t we choose this patch in the first image?
  - This patch is not even unique in the first image.
    - ✓ So, it may be better to find patches that are unique compared to the other patches in the same image.

# How can we find unique patches?

- How about this one?

# How can we find unique patches?

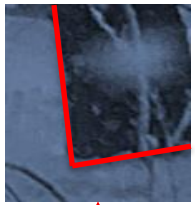
- How about this one?
  - The target patch contains “edge”.
  - Variation in one directions.
  - This patch is different from lots of patches but it is similar to the patches alongside the edge

# How can we find unique patches?

- This one?

# How can we find unique patches?

- This one?
  - This target patch contains “corner”.
    - ✓ Corner: contains two (or more strong edges) with different directions.
  - Since there are two edges, the patches will be different from the target patch if we move in any direction.
  - Corners are perfect!



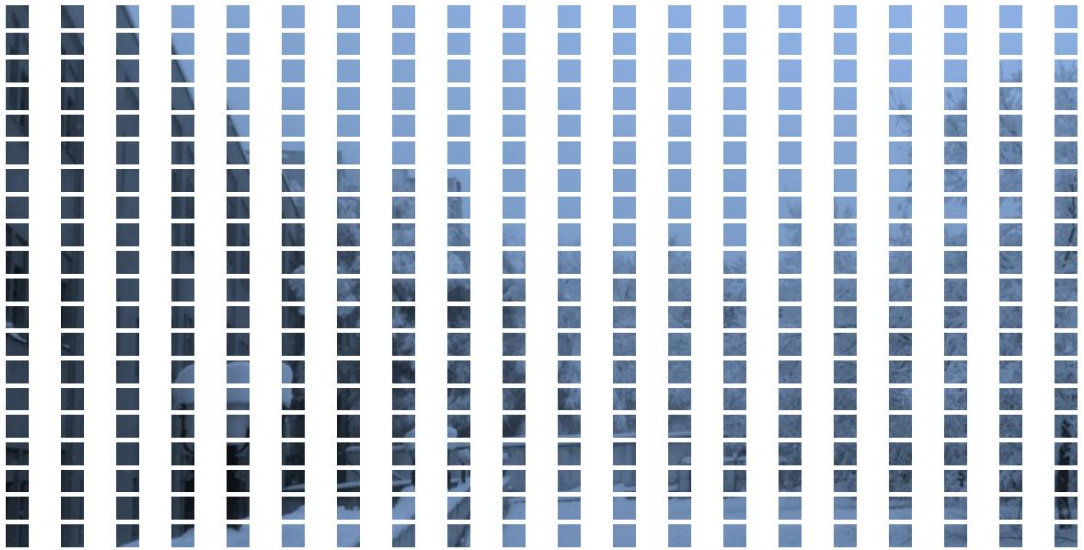
There are two strong edges with different directions

# How similar are two patches?

- Maybe sum squared difference?
  - Given patches I & J:
    - ✓ $Distance = \Sigma_{x,y}(I(x,y) - J(x,y)$
    - ✓ If $Distance$is high, patches are not similar.
    - ✓ If $Distance$is low, patches are similar

# How can we find unique patches?

- We want patches that are unique in the image.
- We can calculate the distance between the patch and every other patch.
- If the patch is similar to many patches, it is not unique; otherwise, it is.



There are 400 patches, it would be computationally very expensive to compare each patch with all the other patches

Note that these patches do not overlap, only for simplicity and visualization. In other words, when we consider overlapping patches (which we should do!), there will be much more patches.

# Features are local

- If a patch is different from its neighbors, it is probably also different from other patches.
- We don't need to compare a patch to all other patches.

# Raw pixel values are not informative

- We know that the highlighted patch is not unique at all!
  - All the patches on the floor, matches this patch.
- This patch is different from its neighbors (their colors slightly differ), so the proposed method will consider it a unique patch while it is not!
- We need another method…

This patch doesn't contain useful information like corners or even edges!. We know that this patch is not unique!

# We need another approach…

- We know that:
  - Corners are useful for finding the unique patches.
  - Corners consist of two (or multiple) strong edges.
  - Image gradients help us in finding the edges
- So…

# Gradients might help

- We know that:
  - Corners are useful for finding the unique patches.
  - Corners consist of two (or multiple) strong edges.
  - Image gradients help us in finding the edges

- Look at nearby gradients components $Grad_x$ and $Grad_y$
  - If gradients are mostly zero, not a lot going on
    - ✓ No edges -> Low variation
  - If gradients are mostly in one direction, edge
    - ✓ One edge -> Still low variation
  - If gradients are in two directions, corner!
    - ✓ Two edges -> good patch!

# How do we tell what's going on with gradients?

- At first, calculate the gradient in x and y directions.
- For each pixel, calculate these values:
  - $Grad_x^2$
  - $Grad_x . Grad_y$
  - $Grad_y^2$

$Grad_x$ is a 2D matrix (like a gray image) that contains the gradient component in x direction. We can $Grad_x$ by applying $Sobel_x$ filter on the image. $Grad_x^2$ is an element-wise operation on $Grad_x$ matrix. So, $Grad_x^2$ is also a 2D matrix (like a gray image).

- Apply gaussian filter on each of these three images.
  - This will combine the information from nearby gradients.

- For each pixel, make the following matrix:
  - $S(i,j) = \begin{bmatrix} (Grad_x^2(i,j) & Grad_x(i,j).Grad_y(i,j) \\ Grad_x(i,j).Grad_y(i,j) & Grad_y^2(i,j) \end{bmatrix}$

This matrix is called "structure matrix"

# Structure matrix

- $$S(i,j) = \begin{bmatrix} (Grad_x^2(i,j) & Grad_x(i,j).Grad_y(i,j) \\ Grad_x(i,j).Grad_y(i,j) & Grad_y^2(i,j) \end{bmatrix}$$

- Find the eigenvalues of this matrix.
  - Eigenvalues of this matrix summarize the distribution of the gradients nearby
    - ✓ if $\lambda_1$ and $\lambda_2$ both are small: There is no gradients, no useful information.
    - ✓ If $\lambda_1 \gg \lambda_2$: Gradients in one direction, probably there is an edge.
    - ✓ If $\lambda_1$ and $\lambda_2$ are similar: Multiple gradient directions, corner!

The assumption is that the larger eigenvalue is $\lambda_1$ and the smaller eigenvalue is $\lambda_2$. ($\lambda_1 \geq \lambda_2$)

# Estimating smaller eigen value

- From linear algebra:
  - $trace(S) = \lambda_1 + \lambda_2$
  - $det(S) = \lambda_1 . \lambda_2$
- Therefore:
  - $\det(S) - \alpha . trace(S) = \lambda_1 . \lambda_2 - \alpha(\lambda_1 + \lambda_2)$
  - $\frac{\det(S)}{trace(S)} = \frac{\lambda_1 . \lambda_2}{\lambda_1 + \lambda_2}$
- So, instead of calculating $\lambda_1$ and $\lambda_2$ directly, we can find one of these two values based on the determinant and trace of S.
- If these two values are high, both eigenvalues are high; otherwise, one or two of them are low.

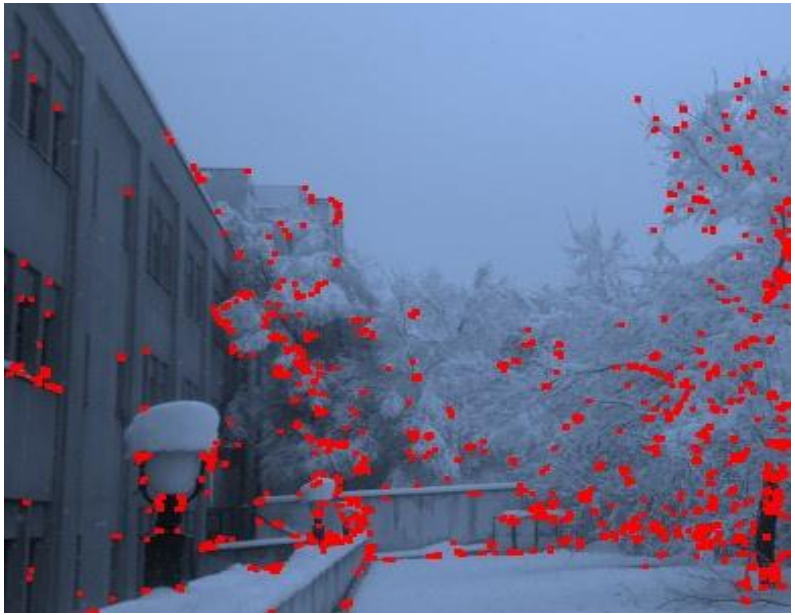
$\alpha$ is a small positive value (for example 0.4, 0.6, …)

# Harris corner detector

- Calculate the derivatives $Grad_x$ and $Grad_y$.
- Calculate three measures:
  - $Grad_x^2, Grad_x.Grad_y, Grad_y^2$
    - ✓ For each pixel, we have these three values.
    - ✓ So, it's like we have three images.
- Calculate the weighted sum
  - In other words, apply gaussian filter on each of these three images.
- Find the structure matrix for each pixel.
- Estimate the response based on the smallest eigenvalues.
- Non-max suppression (like Canny edge detector)

# Harris corner detector

- Lots of corners found in these two images!
- In other words, we know the coordinates of “**pixels**” that are located on corners.

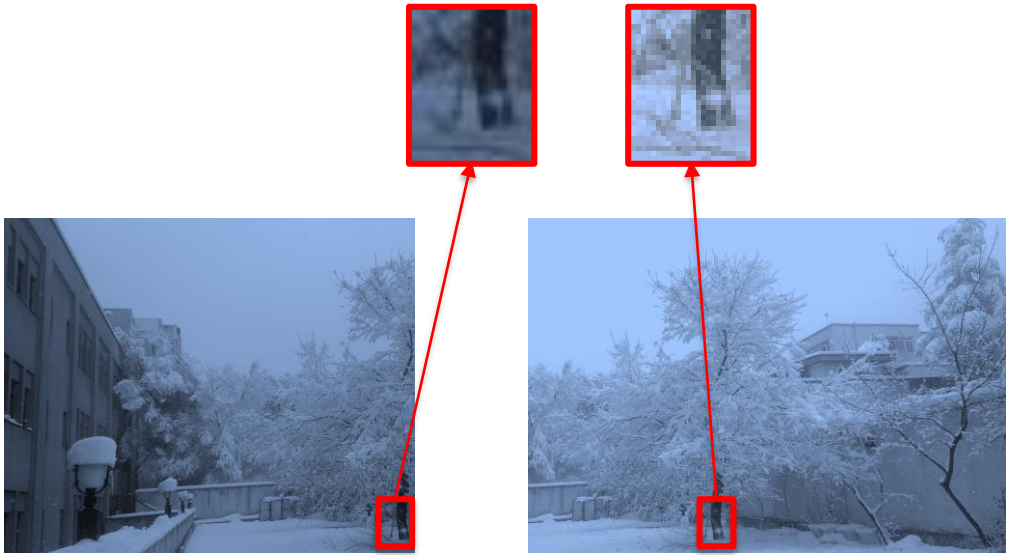


# Now what?

- Need to match image patches to each other.
  - What is a patch? How do we look for matches? Pixels?
- Need to figure out transform between images
  - How can we transform images?
  - How do we solve for this transformation given matches?

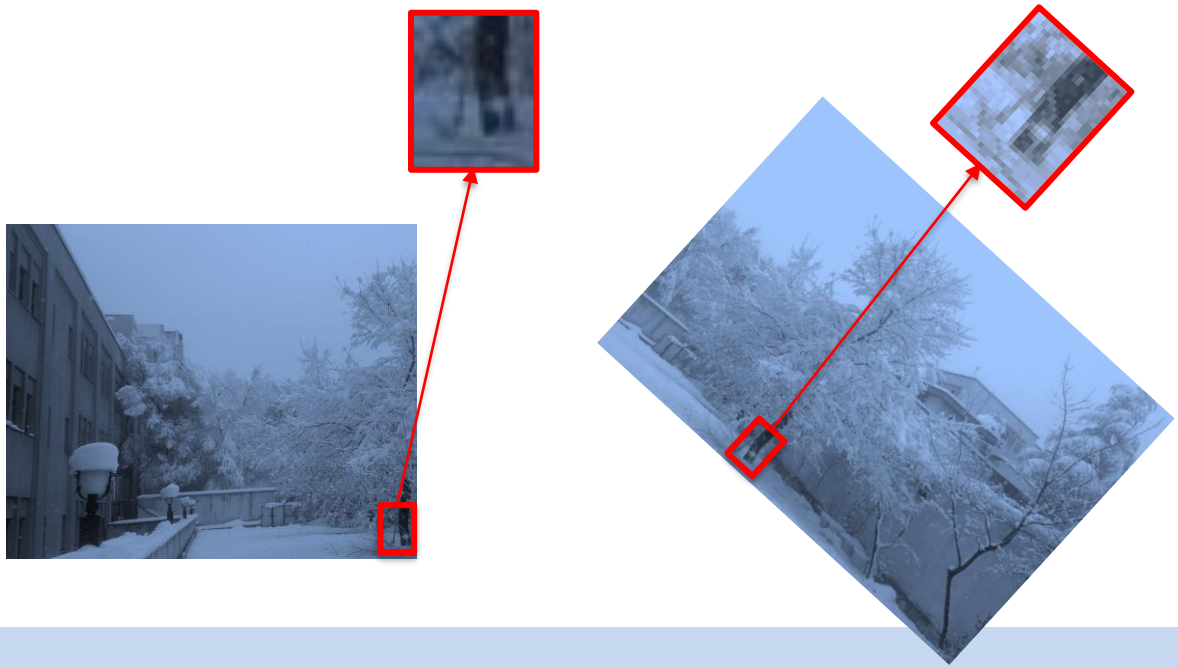
# Matching patches: descriptors!

- What problems are there with just using pixels?
  - Pixel values are not informative!
    - ✓ During taking a panorama picture, the light intensity may change due to the light conditioning of the environment, camera flash, …



# Matching patches: descriptors!

- What problems are there with just using pixels?
  - Pixel values are not informative!
    - ✓ During taking a panorama picture, there might be unwanted rotations



# Matching patches: descriptors!

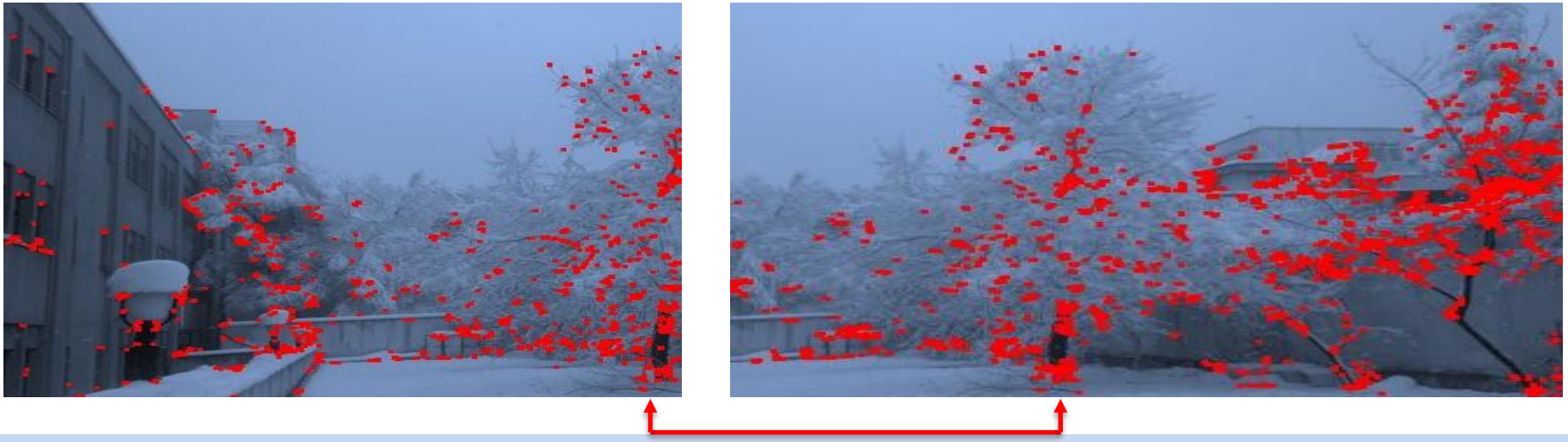
- Instead of using raw pixel values, we can use higher level information from the area of interest in the image like:
  - Gradient information
  - Edges
  - Etc.

- These are region descriptors.

- We'll talk more about descriptors later!

# Matching patches: descriptors!

- Already have our patches that are likely “unique”
- We can loop over good patches in one image
  - Find best match in other image



# We are not done yet!

- We find "**unique patches**" in these two images, "**match**" them based on their similarities, **then "transform" the images to form the panorama image.**

- To be continued…