JSPEC

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Reproduced by

Priyanka Reballi

Arun Kumar Subramanian

K L Bhanu Moorthy

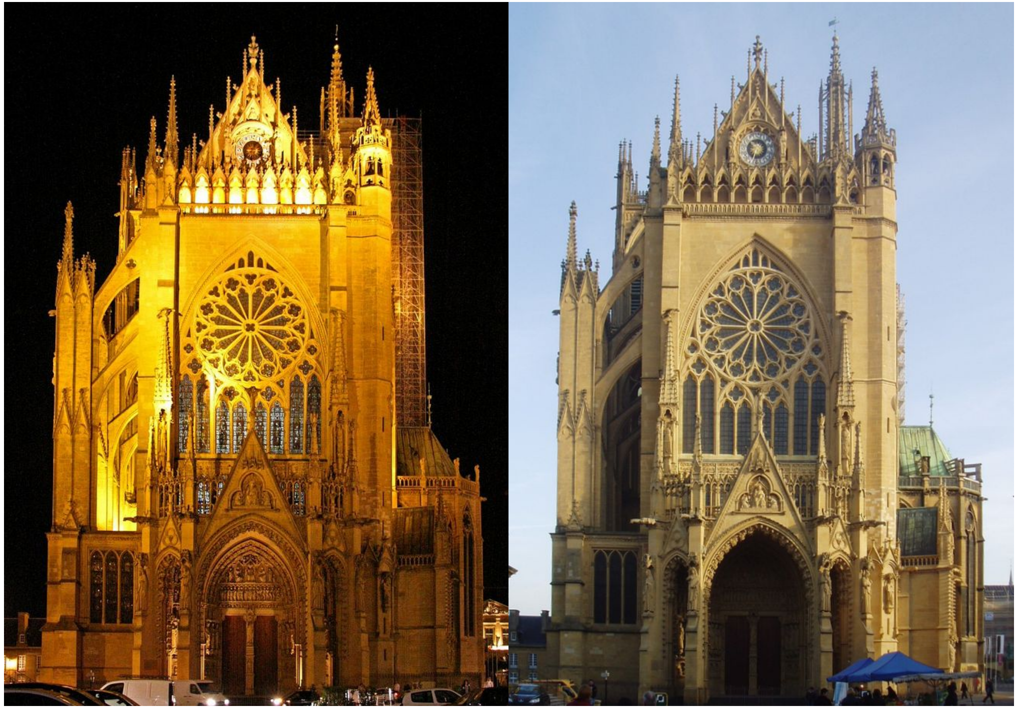
IIIT-Hyderabad

Apr 7th, 2019

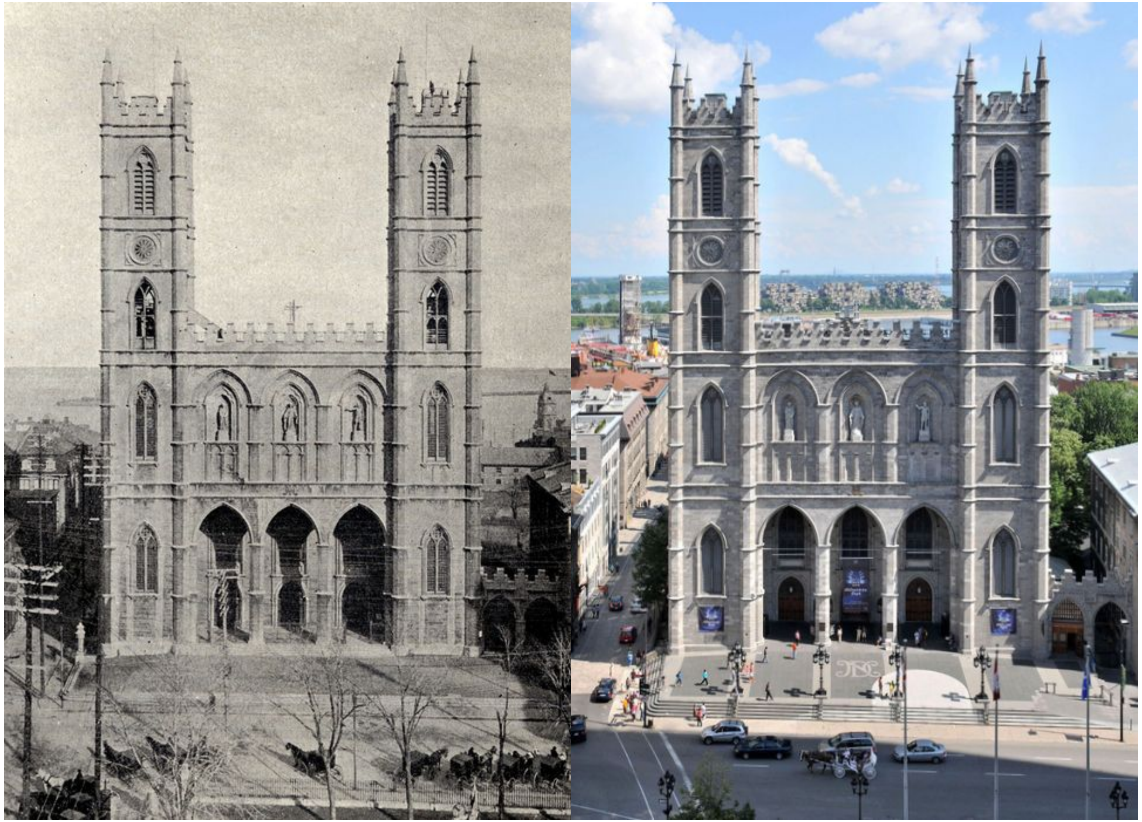
Introduction

- Problem:
  - To match images with disparate appearances
  - Neither intensity nor gradient distributions are locally comparable
  - SIFT is infeasible
- Solution:
  - Detecting and matching persistent features
  - Using eigen-spectrum of the joint image graph of two images.

Disparate Images



Disparate Images



Disparate Images

Introduction

- Spectral methods on Image graph Laplacians have been used earlier, in another form.
- Two significant contributions:
  - Joint image graph without considering proximity
  - New definition of persistent regions.

Methodology

- Image Graph
- Image Features and the Joint Spectrum
- Characterization of persistent regions
- Eigen-function feature matching
- JSPEC Algorithm

Theory

Theory

Image Graph

- Definition of Image Graph
- Function p

Laplacian

- Incident Matrix
- Laplacian Matrix

Obtaining optimum p

- Laplacian's relation to function p
- eigenvectors of L as p

Image Graph

- Image Graph is represented as $G(V, E, W)$
- V contains all image pixels as vertices. If there are total n pixels in the image then $|V| = n$
- E contains all pairwise relationship between every pair of vertices(pixels) thus making G a complete graph. $|E| = \binom{n}{2}$ for undirected graph
- The weight $w_{ij} \geq 0$ associated with an edge $(v_i, v_j) \in E$ encodes the affinity between the pixels represented by vertices v_i and v_j. We can collect these weights into an $n \times n$ affinity matrix $W = (w_{ij})_{i,j=1,\dots,n}$

Function p

- We want to define a function $p : V \to \mathbb{R}$ such that it is a continuous function i.e. difference between $p(v_i)$ and $p(v_j)$ inversely follows w_{ij}
- It is equivalent to say that we want to minimize

$$\lambda = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(p(v_i) - p(v_j))^2$$

- Let Matrix P be defined as $\begin{bmatrix} p(v_1) \\ p(v_2) \\ \vdots \\ p(v_{|V|}) \end{bmatrix}$

Incident Matrix

- For any directed graph $G(V, E)$, consider

$$V = \{v_1, v_2, \dots v_{|V|}\}$$

$$E = \{e_1, e_2, \dots, e_{|E|}\}$$

- Incident Matrix ∇ is $|E| \times |V|$ matrix such that if k^{th} edge is from v_i to v_j with weight w_{ij} then
  - $\nabla_{ki} = +w_{ij}$
  - $\nabla_{kj} = -w_{ij}$
  - $\nabla_{km} = 0, \forall m \neq i, j$

Laplacian Matrix

- $L = \nabla^T \nabla$ is called laplacian of graph
- L is $|V| \times |V|$ matrix where

$$L_{ii} = \sum_{j=1}^{|V|} w_{ij}$$

$$\underset{i \neq j}{L_{ij}} = -w_{ij}$$

- $L = D - W$ where D is degree matrix and W is adjacency matrix

Conclusion

- For any image, a weighted graph is constructed considering each pixel a vertex.
- Edge weights are assigned according to affinity of vertices.
- Laplacian is obtained from adjacency matrix using formula $L = D - W$
- Normalized laplacian can be obtained by formula $L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$
- Eigen Decomposition of L gives v_1 as a trivial solution and $v_2, v_3, \cdots$ as desired solutions

Implementation Details

Outline

- Detect and compute SIFT features
- Affinity Matrix Construction
- Eigen functions
- MSER Detector and matching

Detect and compute SIFT features

- Purpose:
  - To extract sift feature for each key point at two scales with bin sizes 10 and 6 pixels to get 256D vector after concatenating
- Application:
  - Key points: cv2.KeyPoint
  - Description: cv2.xfeatures2d.SIFT_create(); .compute()

Detect and compute SIFT features

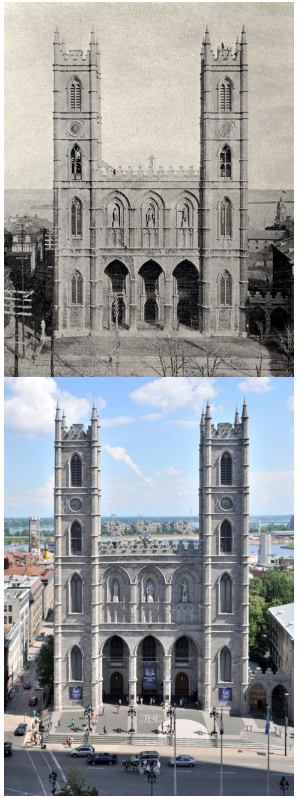
Affinity Matrix Construction

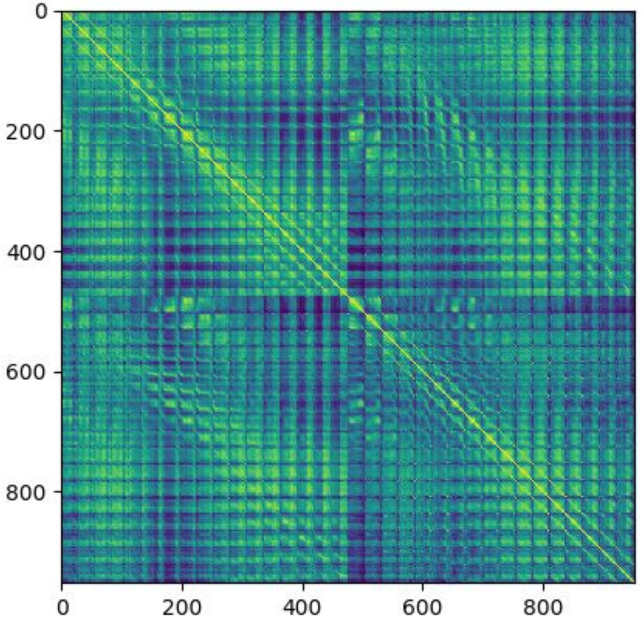
$$W = \begin{pmatrix} W_1 & C \\ C^t & W_2 \end{pmatrix}_{(n_1+n_2)\times(n_1+n_2)} \quad (1)$$

$$(W_i)_{x,y} = \exp\left(-\frac{\|f_i(x) - f_i(y)\|^2}{\sigma_f^2}\right) \quad (2)$$

$$(C)_{x,y} = \exp\left(-\frac{\|f_i(x) - f_j(y)\|^2}{\sigma_f^2}\right) \quad (3)$$

Affinity Matrix Visualization

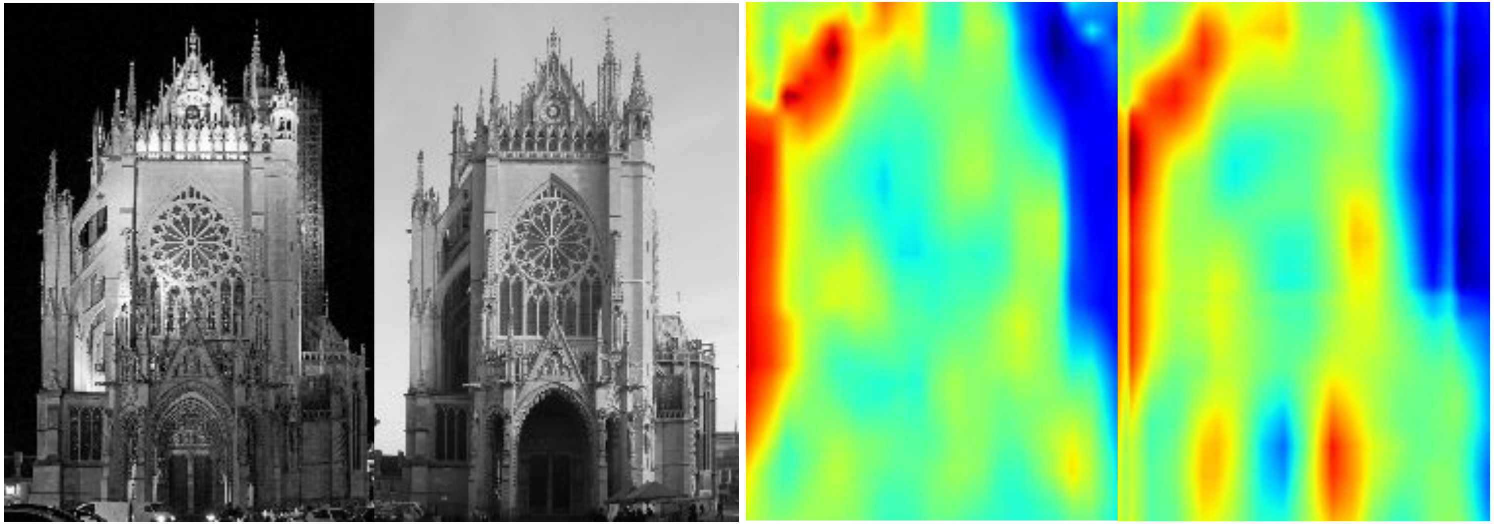




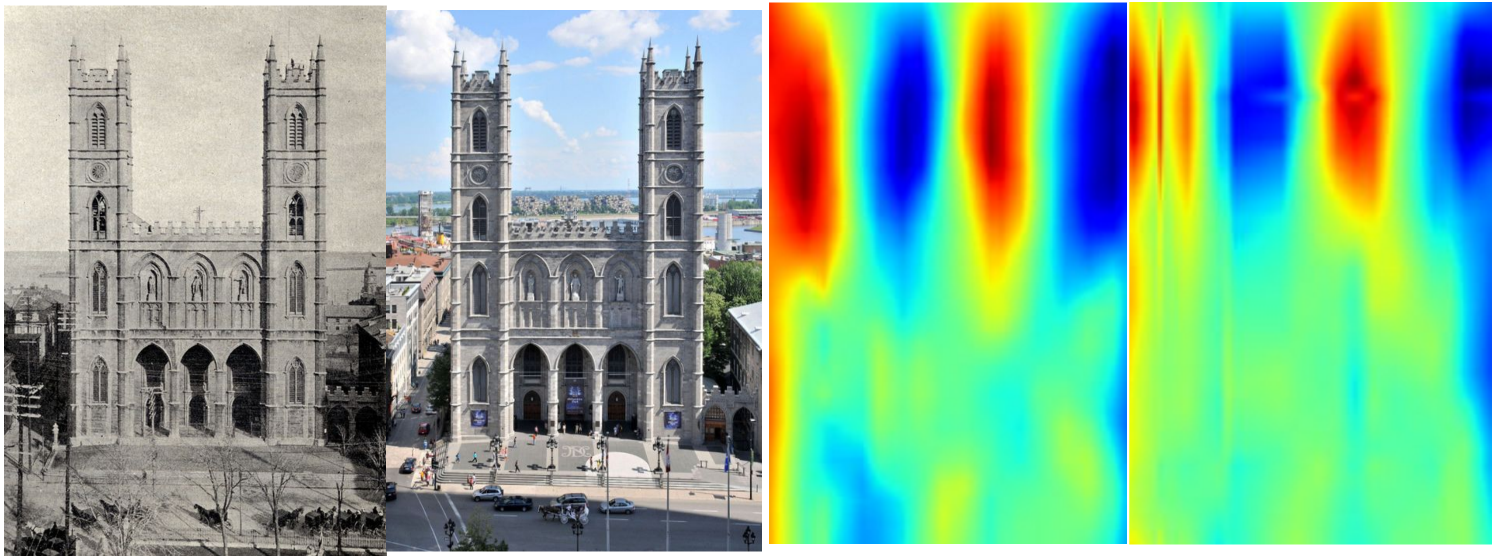
Eigen Functions

- Eigen Functions
  - Laplacian and its eigen decomposition
  - Image representation using eigenvectors
- Our Application:
  - Normalized Laplacian: csgraph.laplacian( ) from scipy.sparse
  - Eigenvectors and eigenvalues: linalg.eig( )

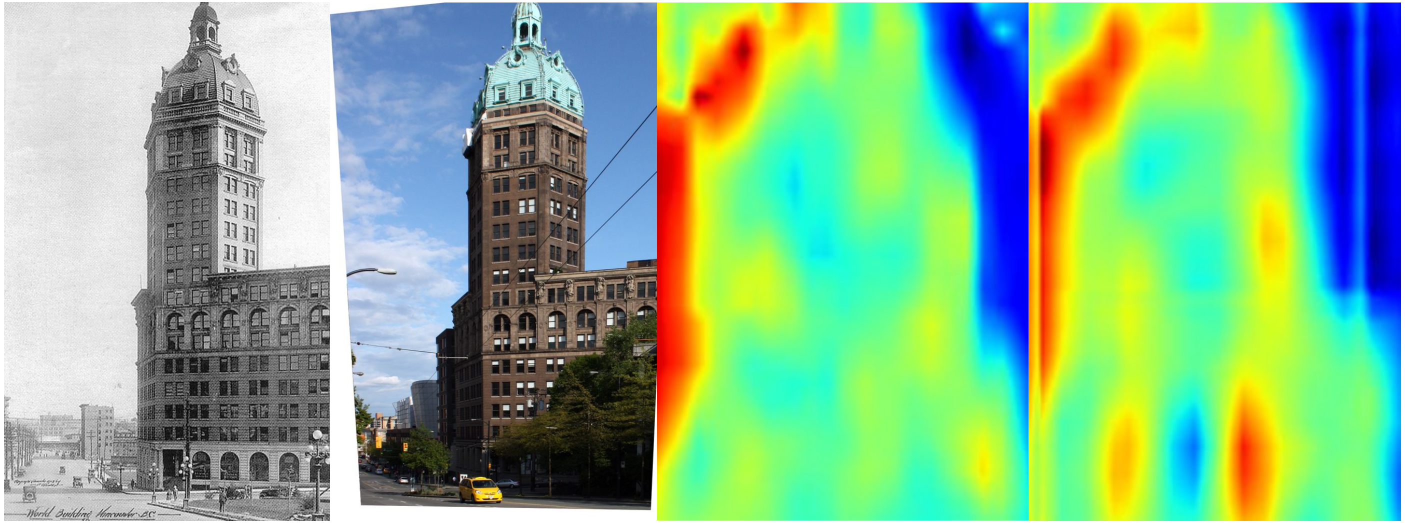
Eigen Functions



Eigen Functions



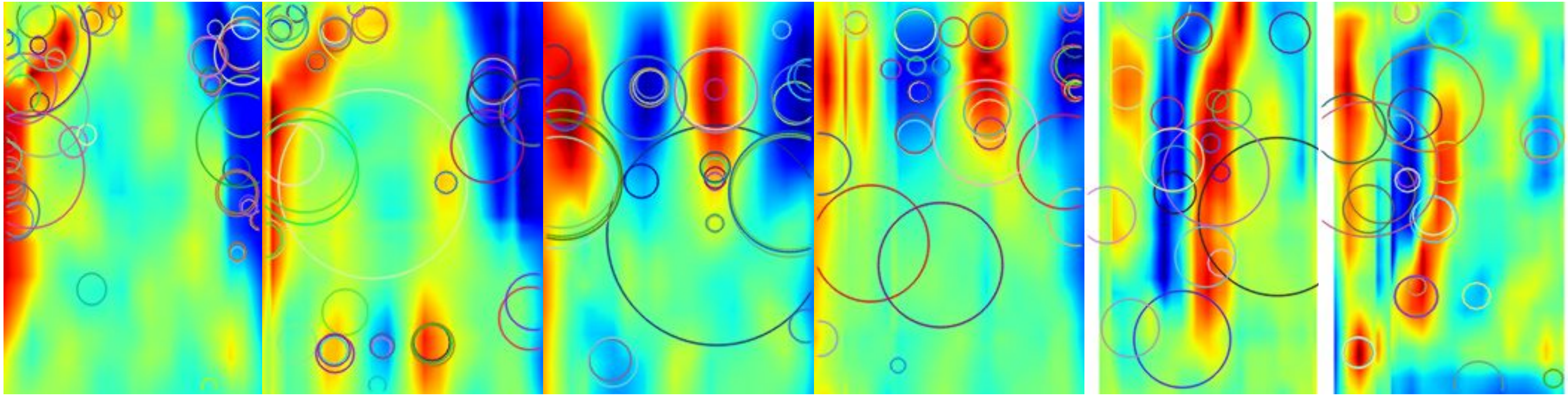
Eigen Functions



MSER Detector

- Purpose:
  - To detect affine-covariant regions in an image
- Application:
  - Detection: cv2.MSER_create(); mser.detect()
  - Description: SIFT match

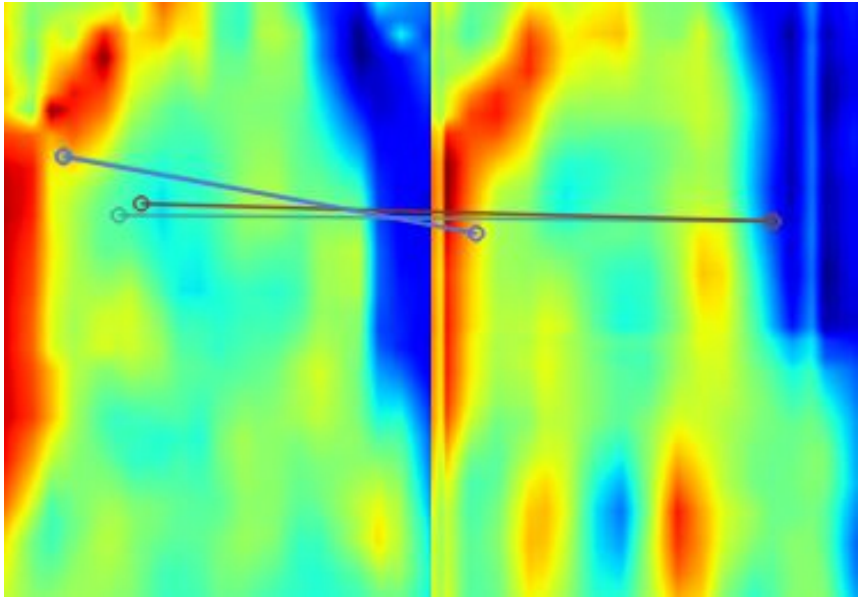
MSER Detector



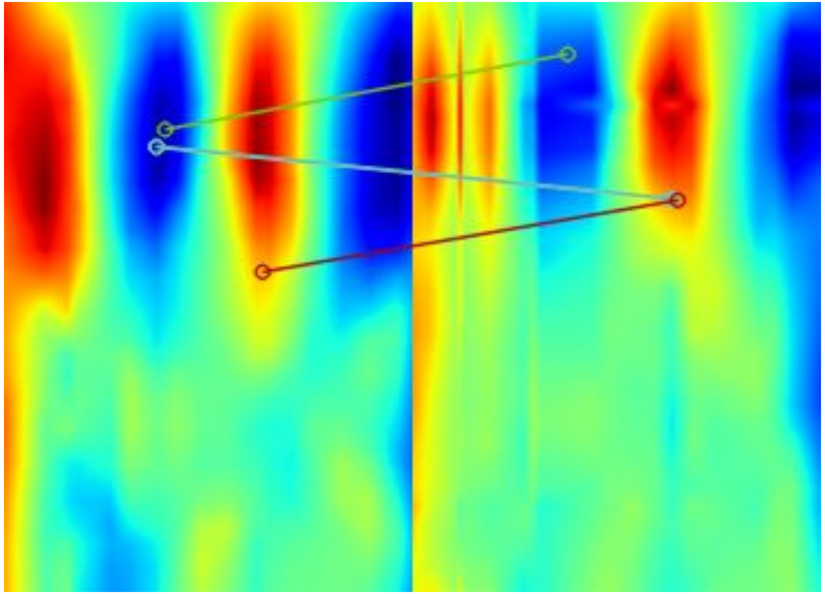
MSER matching

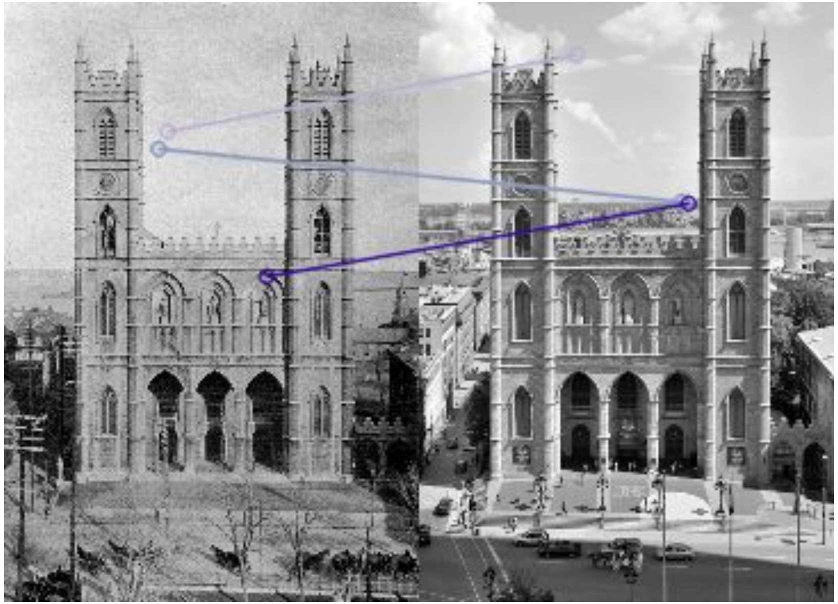
- Purpose:
  - To get good matches from eigen functions
- Our Application:
  - Matcher: cv2.BFMatcher()
  - Good matches: .knnMatch()

MSER matching

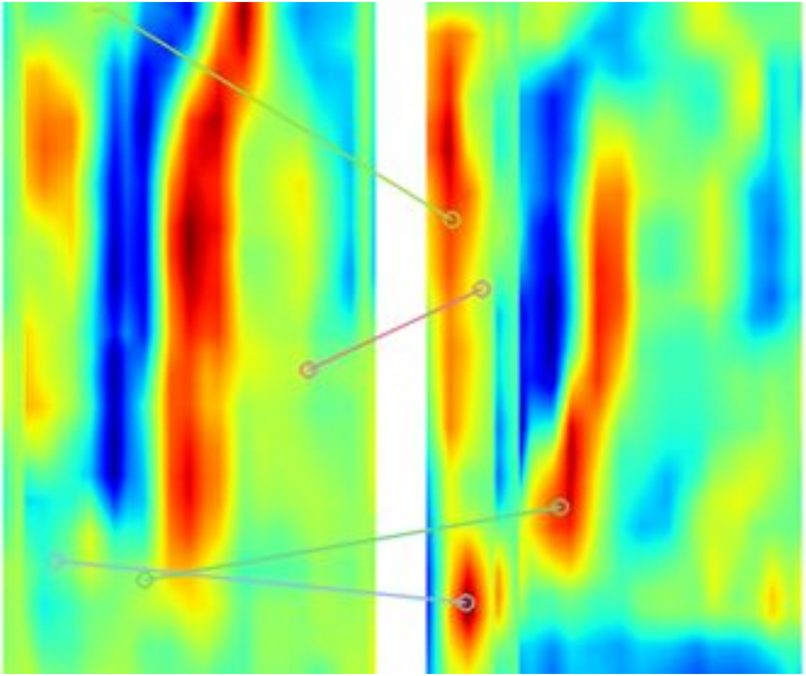
 

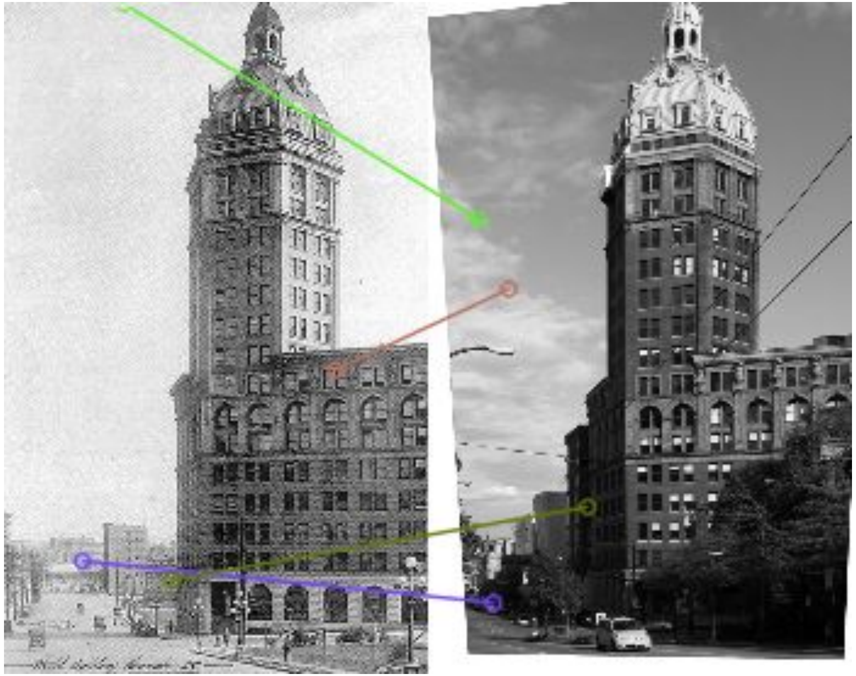
MSER matching





MSER matching





Challenges faced

- Faced issues with NaNs. Resolved by debugging and noticing 0 valued SIFT.
- Faced issues with complex Eigenvalues. Resolved by debugging - Ensuring width vs height of image graph, is used when reordering Eigen map.

Challenges faced

- Reduced the code execution time by using more matrix operations than iterative operations.
- MSER ellipse creation and match;

Code Walkthrough

https://github.com/priya55612/-Joint-Spectral-Correspondence-for-Disparate-Image-Matching/blob/master/eigenMapMatcher.py

Github link

https://github.com/priya55612/-Joint-Spectral-Correspondence-for-Disparate-Image-Matching

Thank you