3D Reconstruction of Notre Dame Cathedral

BY KARAN AHUJA

UNDER THE GUIDANCE OF DR. MATHIEU AUBRY

1st Attempt - OpenCV

- OpenCV 3.1 has some 3D Scene Reconstruction and Structure from motion modules. But most of them require the camera intrinsic (f, cx and cy) to be known. These functions generate a sparse point cloud and a beta version of them is available.
- Therefore auto calibration is required before proceeding. This can be done from the Kruppa Equations for solving the fundamental matrix with N-View Constraints. More information about this can be found at this link. However, it is also stated that intrinsic parameters are recovered with fair but not excellent accuracy.
- Also most OpenCV methods currently available use only stereo block matching methods for 3D reconstruction using epipolar geometry techniques to generate depth maps.

Pipeline for 3D Reconstruction using OpenCV Stereo Matching

LEARNING THE CONCEPTS

- Start with multiple views of a scene.
  - Extract SURF/SIFT/ORB key points from images.
- Find interesting points.
  - Pair-wise matching of images.
- Simultaneously solve for good matches and F (fundamental matrix) using RANSAC.
- Incorporate additional knowledge (camera internal calibration, known angles, planes, etc.) to upgrade uncalibrated F to E.
  - Guess K matrix and then calculate essential matrix. [Hartley]
  - Assumptions made for calculating K:
    - No skew
    - Optical Center is at the center of the image
- Decompose E to find R and t.
- Use R and t, along with point correspondences, to triangulate the real-world 3D positions of the sparse feature points.

$$\boldsymbol{x}_1 \underbrace{\mathbf{K}_1^{-T}\,\mathbf{E}\,\mathbf{K}_0^{-1}}_{\mathbf{F}} \boldsymbol{x}_0 = 0$$

$$\mathbf{E} = \mathbf{K}_1^{-T}\mathbf{F}\mathbf{K}_0^{-1}$$

$$z_c \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{x} = \underbrace{\underbrace{\begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathrm{K}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \begin{bmatrix} \mathrm{R} & \mathrm{t} \\ 0 & 1 \end{bmatrix}}_{\mathrm{P}} \underbrace{\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}}_{p}$$

$$\boldsymbol{x}_0 \sim \mathbf{P}_0\, p$$
$$\boldsymbol{x}_1 \sim \mathbf{P}_1\, p$$

P corresponds to the epipole of the intersection of the epilines

- Bundle adjustment of pose and 3D points.
  - Optimize over the entire set by minimizing re-projection error:

$$\sum_i \sum_j \|x_{ij} - \hat{x}_{ij}\|^2 \quad \text{where} \quad x_{ij} \sim P_i \, p_j$$

- Repeat for multiple images and combine point clouds using Iterative Closest Point.
- Get desired dense depth map, using known epipolar constraints to constrain correspondence and use stereo (per-point triangulation).

OpenCV function Block Chain

Challenges

- Current openCV module only supports 3D reconstruction using disparity maps from stereo left and right images and require the images to be the same sizes. Although the images can be resized, it leads to considerable loss of information and aspect ratio for low resolution images. It use Block Matching Algorithm instead of a feature matching one.
- StereoRectifyUncalibrated calculates simply planar perspective transformation not rectification transformation in object space. It is necessary to convert this planar transformation to object space transformation to extract K matrix.
- OpenCV does not provide a module for ICP in it's stable version and ICP can introduce scale ambiguity which need to be refined using Bundle Adjustment.

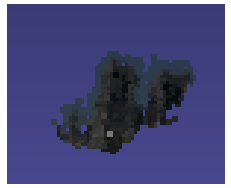
Solution: Instead of implementing these functions in OpenCV use a software tool that already performs it and implement later/ or contribute to openCV 3.1 Sfm module.

VisualSFM and CMVS

USING VISUALSFM FOR SPARSE RECONSTRUCTION AND CMVS FOR DENSE RECONSTRUCTION – USING MESHLAB TO VISUALIZE

Attempt 1 – Taking Recent Photos

- 8 pairs have two-view models
- 8 pairs have fundamental matrices
- 2 cameras used for reconstruction
- 1 model generated
- Total 670 3D vertices generated after Dense Reconstruction



Due to the low resolution of the images the reconstructed 3D image does not have many image points. This is because very less features are detected and matched.

Increasing the number of 3D vertices detected by changing some VisualSFM settings:

- Changing minimum inlier matches for a pair to be considered correct from 15 to 10.
- Setting VisualSFM to output 1 Model and Merge two models if 3D matches is greater than 20

- 7 pairs have two-view models
- 7 pairs have fundamental matrices
- 2 cameras used for reconstruction
- 1 model generated
- Total 904 3D vertices generated after Dense Reconstruction

A better result is obtained, let's explore if tuning the parameters further gives a better result

Attempt 1 – Continued

- Changing threshold for the number of projections when adding a new camera from 20 to 12 and that for pose estimation from 64 to 40.
  - 7 pairs have two-view models
  - 7 pairs have fundamental matrices
  - 4 cameras used for reconstruction
  - Total 1462 3D vertices generated after Dense Reconstruction

All the methods to improve results till now have been done by changing VisualSfm parameters. Let's see how to further better them by:

- ❖Adding a better feature matching algorithm (VisualSFM uses Multi-threaded match). Let's see if using a Flann or Brute Force may give better results.
- ❖Increasing the data size for running SFM by manipulating the current one.

Attempt 2

- VisualSFM lets you add the key points you want to match for achieving Sparse Reconstruction. Let's try to a simple Flann or Brute Force Matcher on the images. You have to input the 0-based feature indices for customized matching. The inlier condition of greater than 10 is used.
  - 4 cameras used for reconstruction
  - Total 1358 3D vertices generated after Dense Reconstruction

Therefore, there is not much difference and the multithreaded feature matching of VisualSFM proves to be better and faster.

Attempt 2 - Continued

- Now to increase the data size I use Laplacian Pyramid to upscale the image and then I flip it to obtain it's mirror image. This is a common technique (mirroring of image) used to train Active Appearance Models with more sets.
  - 29 pairs have two-view models
  - 44 pairs have fundamental matrices
  - 3 cameras used for reconstruction
  - Total 5332 3D vertices generated after Dense Reconstruction

- **4 times increase** **compared to attempt 1**

Input:
Orig_Img
Flip(LaplacianUp(Orig_Img))

- 97 pairs have two-view models
- 174 pairs have fundamental matrices
- 14 cameras used for reconstruction
- Total 6924 3D vertices generated after Dense Reconstruction
- It’s evident that due to high scaling image quality decreases (blurred).

Input:
Orig_Img
LaplacianUp(Orig_Img))
Flip(LaplacianUp(Orig_Img))

This is because many features are not matched due to:

a) Change in scale and orientation
b) Change of illumination (night/day)
c) Detection of common non-essential repetitive patterns such as lamps and benches.
d) Various occlusions such as trees, people, etc….. .

Attempt 3 – Taking All Photos

Load the images in gray scale format and perform histogram equalization on the historical images set, to enhance the contrast. Now using Visual SFM for 3D reconstruction using old settings:

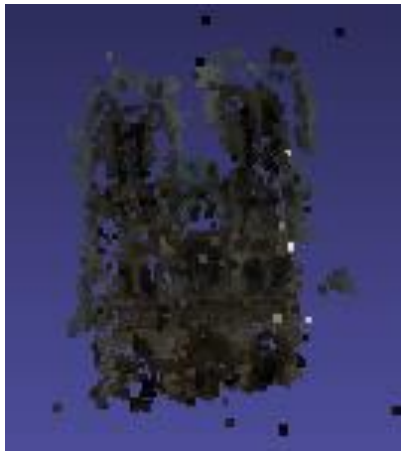
- 10 pairs have two-view models
- 10 pairs have fundamental matrices
- 4 cameras used for reconstruction
- Total 1325 3D vertices generated after Dense Reconstruction

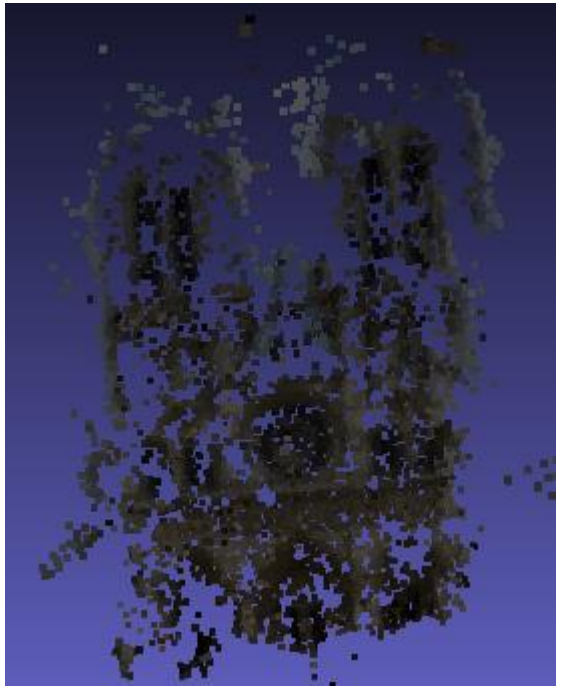
Without histogram equalization only 1050 points are generated

Input:
Orig_Img
Flip(LaplacianUp(Orig_Img))

Output:
35 pairs have two-view models
47 pairs have fundamental matrices
6 cameras used for reconstruction
Total 4682 3D vertices generated after Dense Reconstruction

- 88 pairs have two-view models
- 209 pairs have fundamental matrices
- 14 cameras used for reconstruction
- Total 5176 3D vertices generated after Dense Reconstruction





It should be noted that adding of historical images does not bring about much change in the number of 3D vertices generated.
In fact the number of points were more for the recent images as compared to the combination of the two. This occurs because many of the historical images contribute less/none to the matching of key points. Hence, a proper mechanism is needed to extract these points.

<u>Input:</u>
Orig_Img
LaplacianUp(Orig_Img))
Flip(LaplacianUp(Orig_Img))

Attempt 4 – Taking Only Historical Photos

Load the images in gray scale format and perform histogram equalization on the historical images set, to enhance the contrast.

Input:
Orig_Img
LaplacianUp(Orig_Img))
Flip(LaplacianUp(Orig_Img))

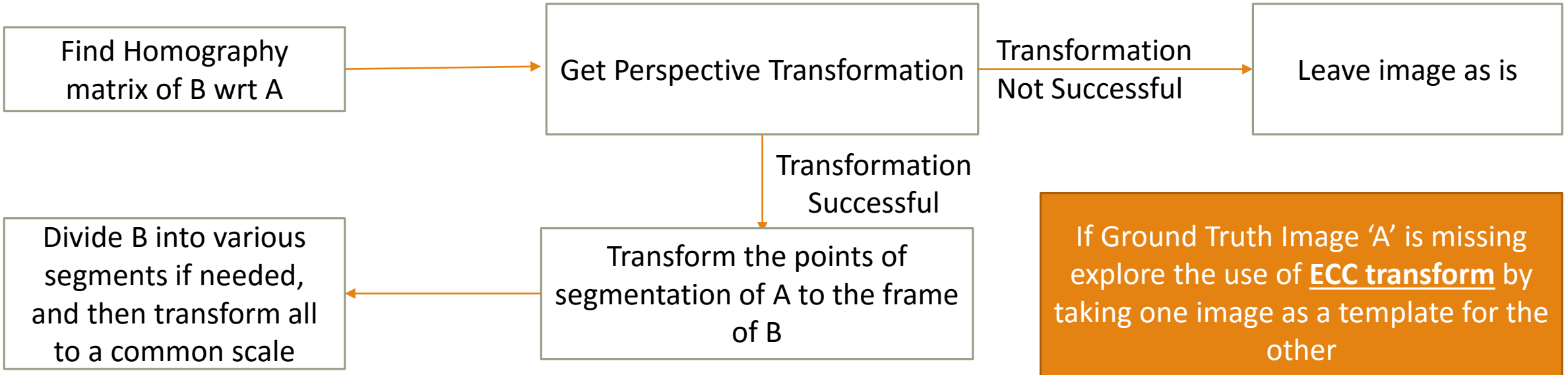
31 pairs have two-view models
112 pairs have fundamental matrices
5 cameras used for reconstruction
Total 2983 3D vertices generated after Dense Reconstruction

Problem: Lower part of reconstruction missing!! Not enough feature matches due to low image quality and blurring of image.

Note: Without proper pre-processing and contrast equalization very few vertices initialized (200).

Proposed Method for Image Segmentation and Normalization

Keep a ground truth image ‘A’ which is of decent resolution and has the points of segmentation of the historical building. It should have a near frontal view for ease of comparison. The image to be detected will be referred to as ‘B’.



If Ground Truth Image ‘A’ is missing explore the use of **ECC transform** by taking one image as a template for the other

Sample Example showcasing Proposed Method

As HoG based Detector has not yet been trained I showcase the example using Homography to find Perspective Transform.

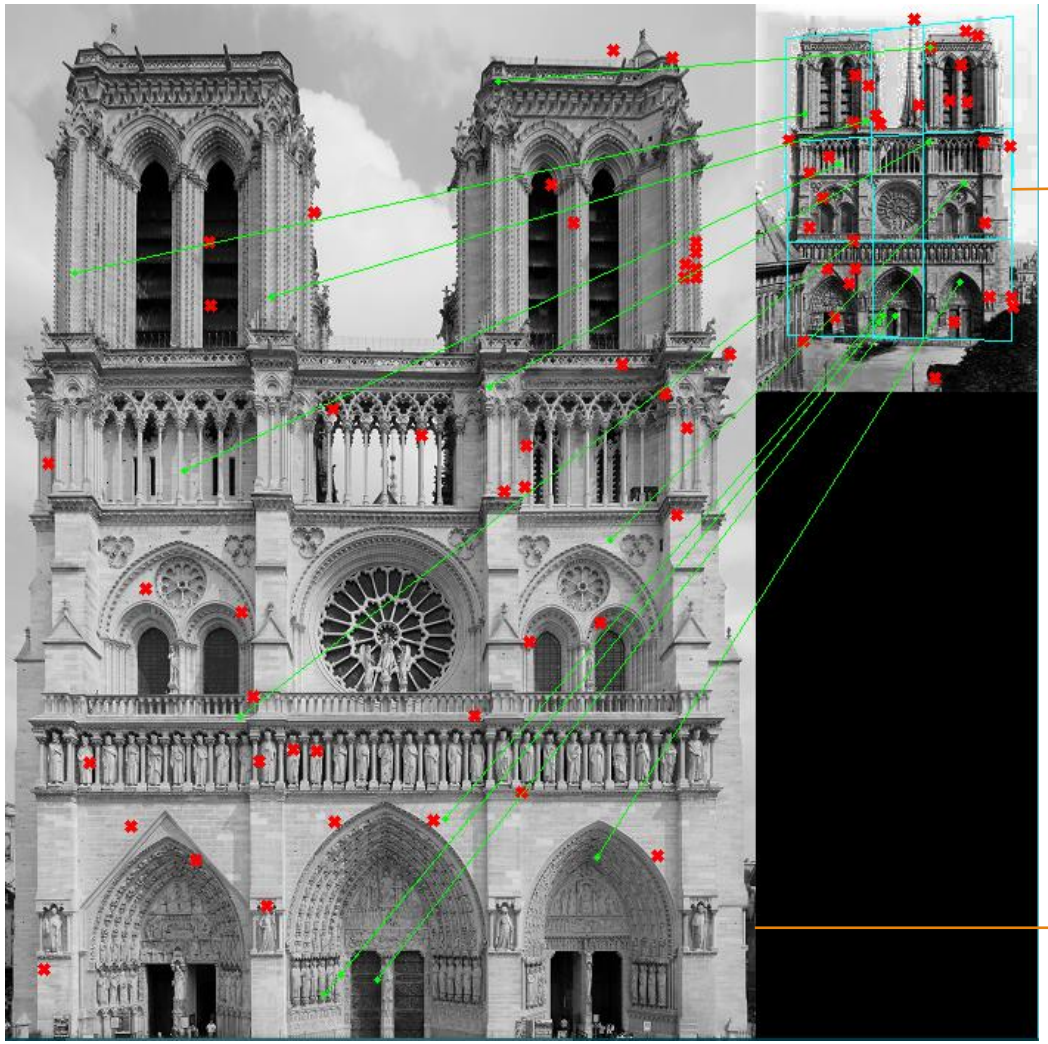


Image A. The bounding box showcases the segmentation and perspective transform of points from B

Image A after normalization to common scale and orientation. The various segmented parts can also be normalized in this manner.

Image B which contains the annotated points of segmentation

The proposed method may help to overcome some of the challenges listed in Slide 12.

References

http://www.theia-sfm.org/sfm.html

http://grail.cs.washington.edu/projects/mvscpc/

http://www.cs.cornell.edu/~snavely/bundler/

http://phototour.cs.washington.edu/

http://ccwu.me/vsfm/

http://www.di.ens.fr/cmvs/

Structure from Motion – Bryan S. Morse (CS 650)

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_epipolar_geometry/py_epipolar_geometry.html

http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

http://openmvg.readthedocs.org/en/latest/software/SfM/SfM/

http://stackoverflow.com/questions/21855335/structure-from-motion-from-multiple-views