# A Comparison Between Different Approaches of Solving Nonlinear Least Squares in the Case of Bundle Adjustment

Ariyan Zarei

University of Arizona

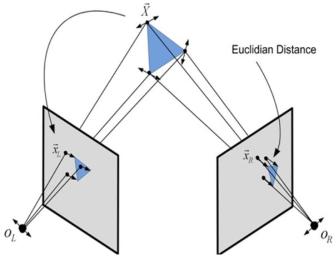
*ariyanzarei@email.arizona.edu*

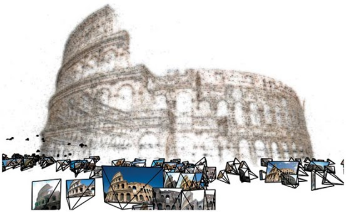
December 9, 2020

# Overview

# Bundle Adjustment Review

# Visual Reconstruction and Bundle Adjustment



# Definitions



- **Visual Reconstruction:** Recover a model of a 3D scene from multiple images.
- **Scene Model:** Collection of isolated 3D features, e.g., points, lines, etc.
- **Bundle Adjustment:** Problem of refining a visual reconstruction model to produce jointly optimal 3D structure and viewing parameter (camera pose/calibration) estimates.
- **jointly**: Solution is simultaneously optimal with respect to both structure and camera variations.
- **optimal**: Parameter estimates are found by minimizing some cost function that quantifies the model fitting error.
- **Bundle** in the name refers to the bundles of light rays leaving each 3D feature and converging on each camera center.

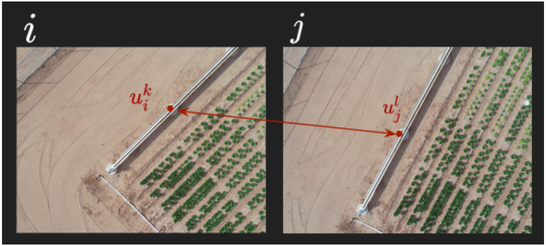
# Bundle Adjustment for Image Stitching



- Bundle Adjustment from Brown and Low paper (projection error)
- Minimizing Projection Error

$$e = \sum_{i=1}^{n} \sum_{j \in I(i)} \sum_{k \in K(i,j)} h(r_{ij}^k)$$

$$r_{ij}^k = u_i^k - H_{ij} u_j^k$$



# MegaStitch and Bundle Adjustment

# MegaStitch

- Large scale image stitching method
- Prevent drift and inconsistency
- Include all available sources of information
- Can be used on Drone and Gantry images
  - Translation/Similarity/Affine
  - Linear Least Squares
  - Proposed a new approach of bundle adjustment
- Can be used on other dataset with Homography
  - Nonlinear Least Squares
  - Main point of this presentation

# MegaStitch, Bundle Adjustment for Homography case



- Consider a reference image
- We estimate absolute Homographies between each image and the reference image
- Bundle Adjustment

$$e = p + \sum_{i=1}^{n} \sum_{j \in I(i)} \sum_{k \in K(i,j)} r_{ij}^{k}$$

$$r_{ij}^{k} = \sqrt{(u_{ir}^{k}[x] - u_{jr}^{k}[x])^2 + (u_{ir}^{k}[y] - u_{jr}^{k}[y])^2}$$

$$u_{ir}^{k}[x] = \frac{H_i^1 u_i^k}{H_i^3 u_i^k}, \quad u_{ir}^{k}[y] = \frac{H_i^2 u_i^k}{H_i^3 u_i^k}$$

- $p$ is a penalty term that enforces $H_r = I$ (for the reference image).

# MegaStitch, Bundle Adjustment for Homography case

$$e = p + \sum_{i=1}^{n} \sum_{j \in I(i)} \sum_{k \in K(i,j)} r_{ij}^k$$

$$r_{ij}^k = \sqrt{(u_{ir}^k[x] - u_{jr}^k[x])^2 + (u_{ir}^k[y] - u_{jr}^k[y])^2}$$

$$u_{ir}^k[x] = \frac{H_i^1 u_i^k}{H_i^3 u_i^k}, \quad u_{ir}^k[y] = \frac{H_i^2 u_i^k}{H_i^3 u_i^k}$$

- $H_i^1$: the first row of homography matrix for image $i$ (these are the parameters).
- $u_i^k$: location of the keypoint $k$ in image $i$.
- $u_{ir}^k$: projected keypoint $k$ from image $i$ into the reference image.

# Solving Nonlinear Least Squares



Python Scipy

- **leastsq** function: unconstrained nonlinear least squares solver.
  - callable function that calculates the residuals
  - starting point
  - optional callable function that calculates the jacobians
  - wrapper around the MINIPACK's lmdif and lmder functions (Fortran)
  - Levenberg-Marquardt algorithm

# Solving Nonlinear Least Squares



- **least_squares** function: nonlinear least squares solver with bounds on variables (newer).
  - callable function that calculates the residuals
  - starting point
  - method for estimating the jacobians: **2-point**, **3-point** or optional callable function that calculates the jacobians
  - minimization method
    - trf : Trust Region Reflective algorithm, large sparse problems with bounds.
    - dogbox : dogleg algorithm with rectangular trust regions, small problems with bounds.
    - lm : Levenberg-Marquardt algorithm as implemented in MINPACK, small unconstrained problems.

MGRAPH

Ariyan Zarei



# Comparing Optimization Approaches and Parameters

# Comparing Optimization Approaches



- Comparing the effect of calculating vs estimating (2 cases) the jacobians using the two mentioned functions on
  - Speed
  - Accuracy

# Jacobian Matrix



- Partial Derivative of each residual with respect to each variable

$$J = \frac{\partial r}{\partial H} = \begin{bmatrix} \frac{\partial r_1}{\partial H_1} & \frac{\partial r_1}{\partial H_2} & \cdots & \frac{\partial r_1}{\partial H_n} \\ \frac{\partial r_2}{\partial H_1} & \frac{\partial r_2}{\partial H_2} & \cdots & \frac{\partial r_2}{\partial H_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial r_m}{\partial H_1} & \frac{\partial r_m}{\partial H_2} & \cdots & \frac{\partial r_m}{\partial H_n} \end{bmatrix}$$

- Approximation
  - 2-point
  - 3-point
- Analytical Form

# MegaStitch Jacobian Matrix



Calculating Jacobians analytically

- 18 different types of equations
- Calculated manually

$$r_{ij}^k = \sqrt{(u_{ir}^k[x] - u_{jr}^k[x])^2 + (u_{ir}^k[y] - u_{jr}^k[y])^2}$$

$$\frac{\partial r_{ij}^k}{H_i^{11}} = \frac{1}{2}\frac{1}{\sqrt{r_{ij}^k}}[2(u_{ir}^k[x]\frac{\partial u_{ir}^k[x]}{\partial H_i^{11}} - u_{jr}^k[x]\frac{\partial u_{jr}^k[x]}{\partial H_j^{11}})+ 2(u_{ir}^k[y]\frac{\partial u_{ir}^k[y]}{\partial H_i^{11}} - u_{jr}^k[y]\frac{\partial u_{jr}^k[y]}{\partial H_j^{11}})]$$

# Experiments



1. leastsq + no jacobian
2. leastsq + analytical jacobian
3. least_squares + 2-point
4. least_squares + 3-point
5. least_squares + analytical jacobian

# Experiments



leastsq + no jacobian

- running time on 5 images: 43.19 *s*

# Experiments

leastsq + analytical jacobian

- running time on 5 images: 00.61 $s$

# Experiments



least_squares + 2-point

- running time on 5 images: 03.84 $s$

# Experiments



least_squares + 3-point

- running time on 5 images: 04.72 $s$

# Experiments

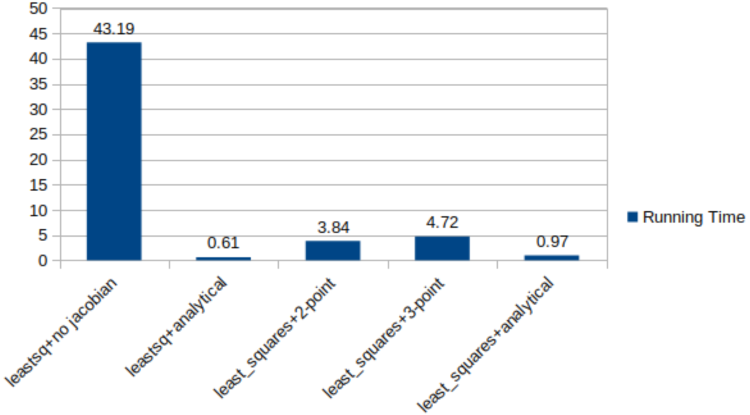


least_squares + analytical jacobian

- running time on 5 images: 00.97 $s$

# Results

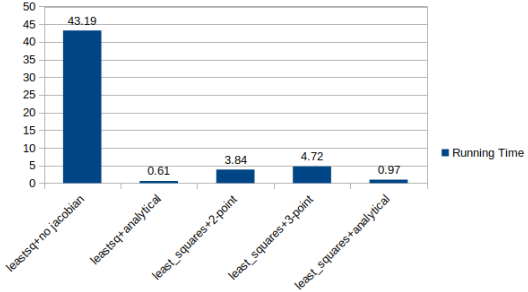



50
45
40
35
30
25
20
15
10
5
0
43.19
0.61
3.84
4.72
0.97
leastsq+no jacobian
leastsq+analytical
least_squares+2-point
least_squares+3-point
least_squares+analytical
Running Time

# Results

Figure: Left to right: leastsq, leastsq+analytical, least_squares+2, least_squares+3, least_squares+analytical

# Results


50
45
40
35
30
25
20
15
10
5
0
43.19
0.61
3.84
4.72
0.97
leastsq+no jacobian
leastsq+analytical
least_squares+2-point
least_squares+3-point
least_squares+analytical
Running Time

# Conclusions

- Calculating jacobians analytically helps a lot whenever possible ($\approx$ 80X faster for leastsq).
- Use leastsq when you don't have bounds.
- least_squares is generally faster compared to leastsq.
- probably we need to tune the parameters of least_squares with analytical jacobians to get better results.

# The End



Thank you Very much for you attention.

I will upload the slide to my homepage at http://vision.cs.arizona.edu/ariyanzarei/