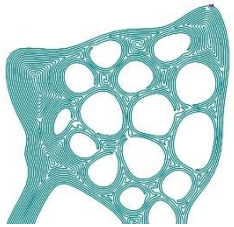
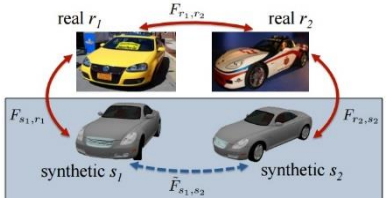
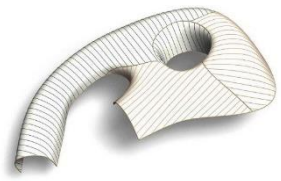
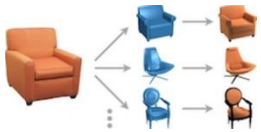
CS 395T
Lecture 12: Feature Matching and Bundle Adjustment



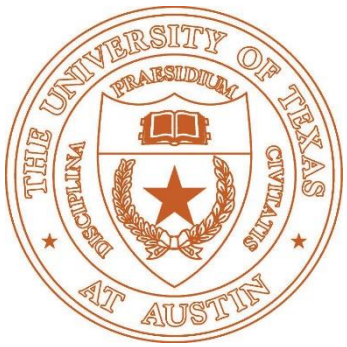


Qixing Huang

October 10st 2018



Lecture Overview

- Dense Feature Correspondences

- Bundle Adjustment in Structure-from-Motion

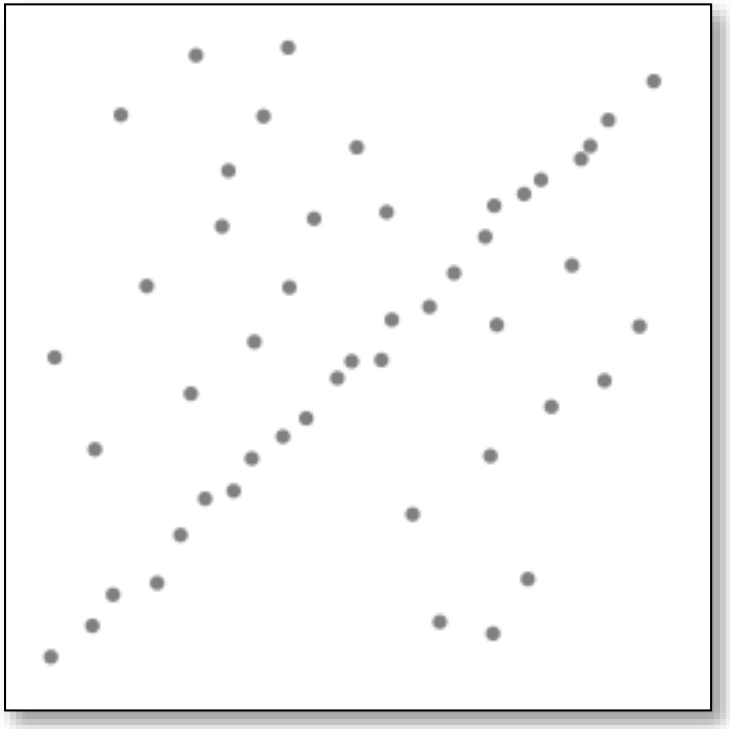
Image Matching Algorithm

- Given images A and B

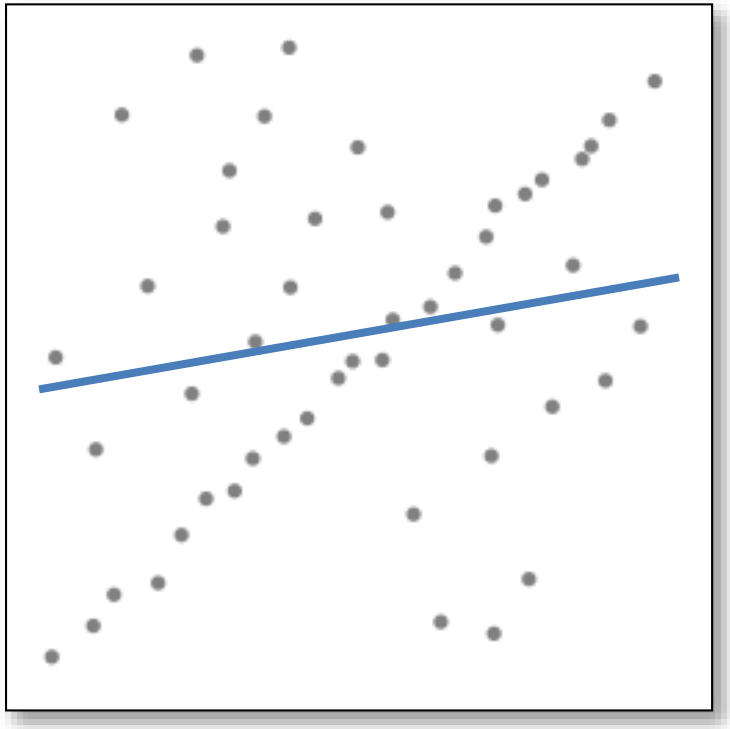
- Compute image features for A and B
- Match features between A and B
- Estimate the essential matrix

Robustness

- Let's consider a simpler example… linear regression



Problem: Fit a line to these datapoints



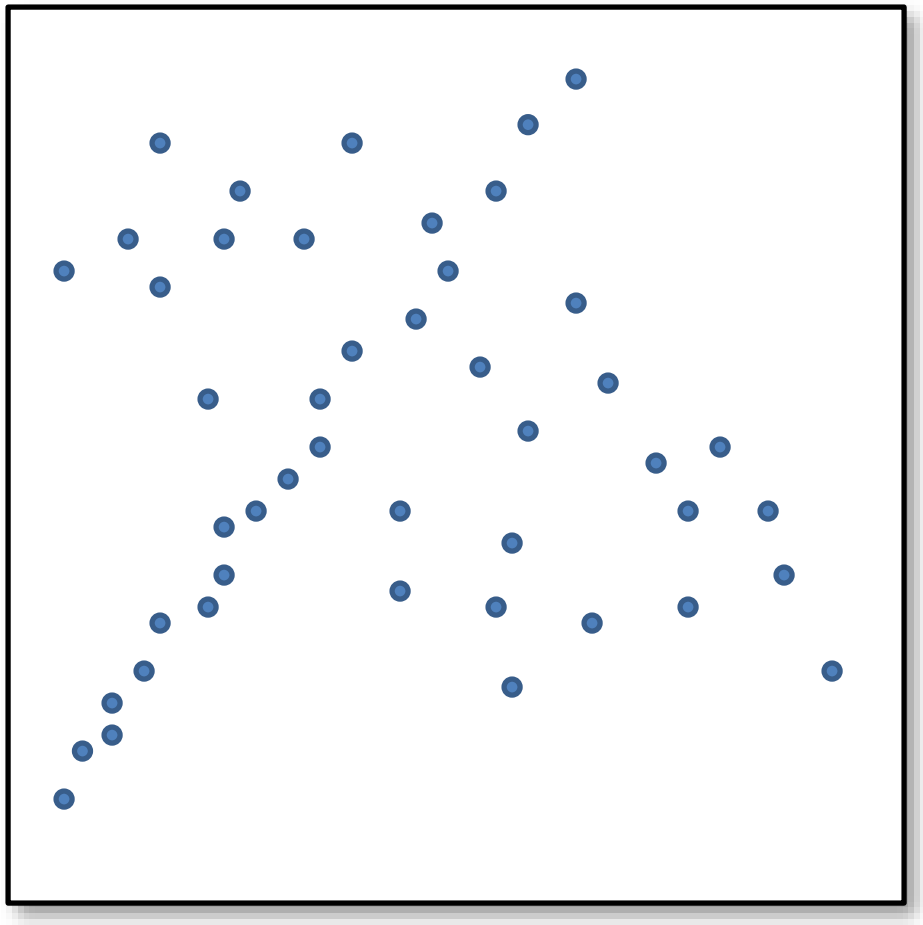
Least squares fit

Potential Fix!

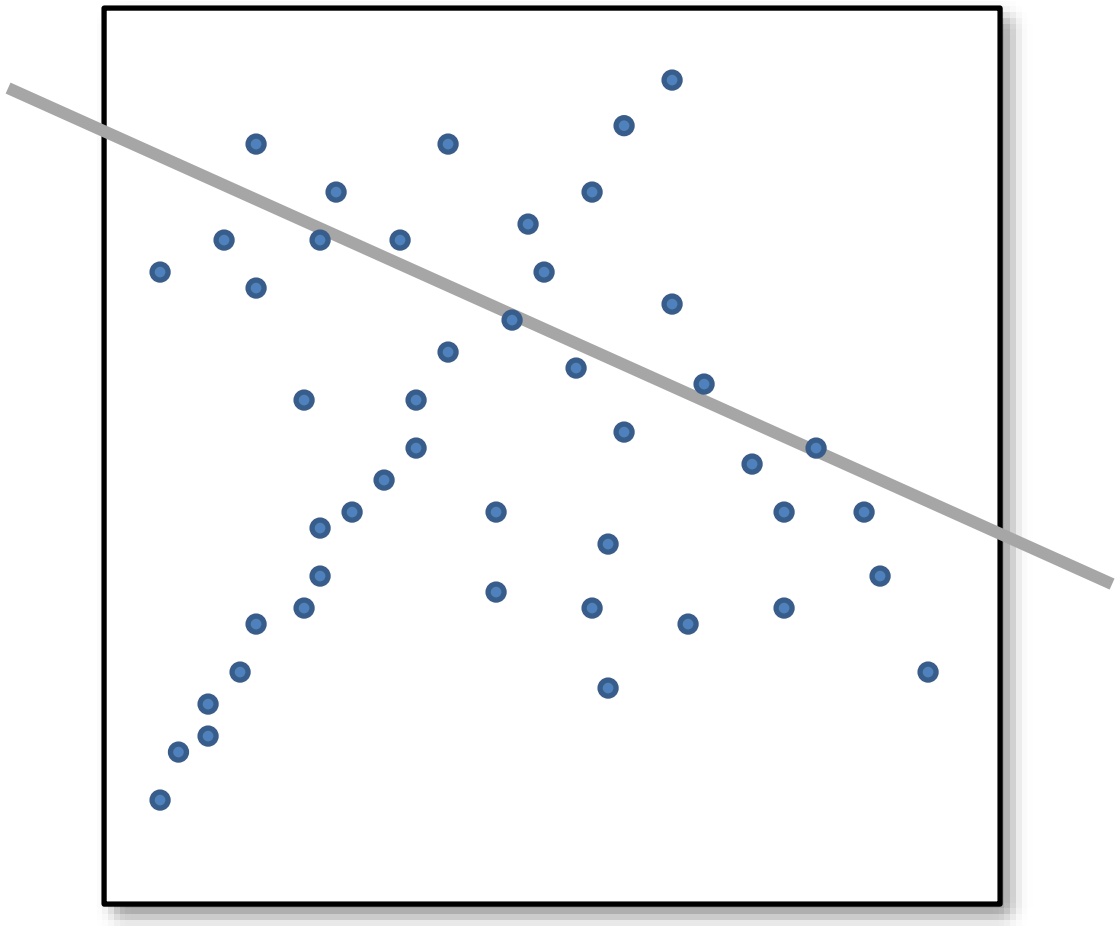
- Given a hypothesized line
- Count the number of points that “agree” with the line
  - “Agree” = within a small distance of the line
  - I.e., the **inliers** to that line

- For all possible lines, select the one with the largest number of inliers

Counting inliers

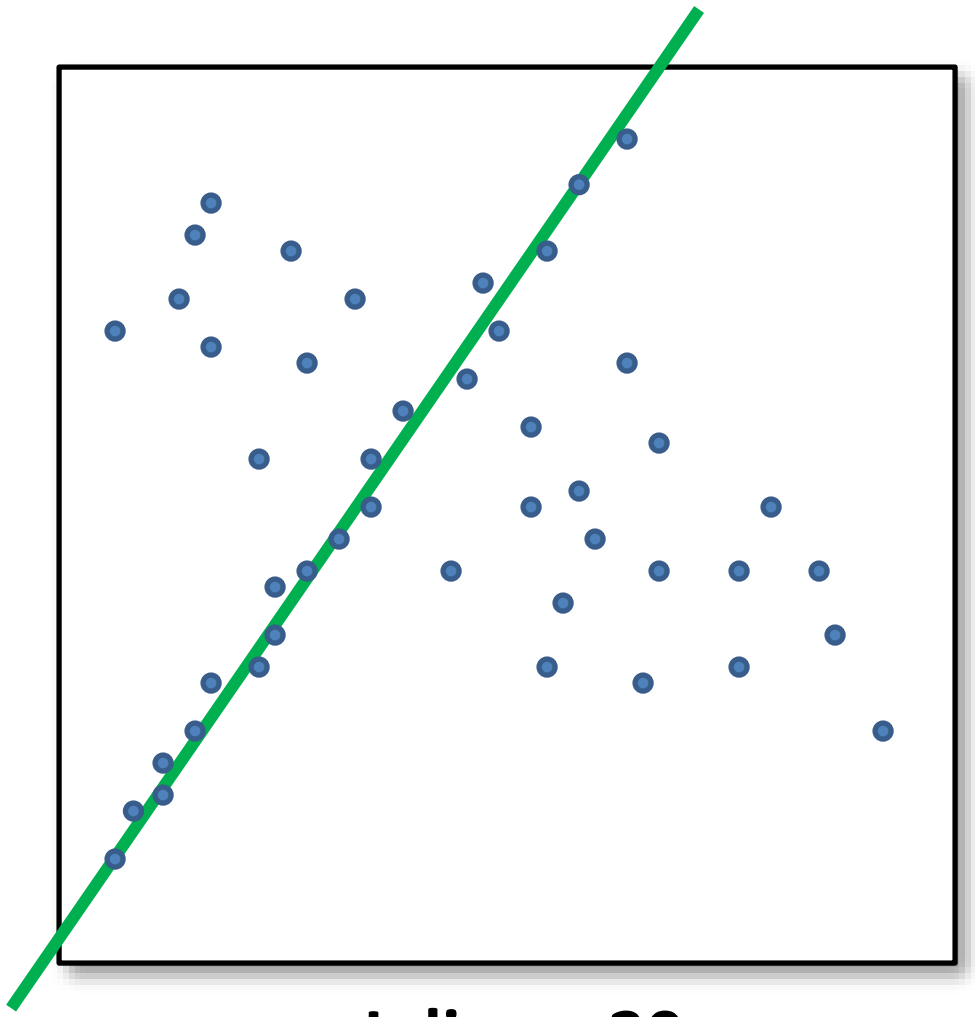


Counting inliers



Inliers: 4

Counting inliers



Inliers> 20

How do we find the best line?

- Unlike least-squares, no simple closed-form solution--- we will get back to this, e.g., using robust norms

- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

RANSAC

- General version:
  1. Randomly choose *s* samples
     - Typically *s* = minimum sample size that lets you fit a model
  2. Fit a model (e.g., line) to those samples
  3. Count the number of inliers that approximately fit the model
  4. Repeat *N* times
  5. Choose the model that has the largest set of inliers

Analysis of RANSAC

however, can be determined as a function of the desired probability of success p using a theoretical result. Let p be the desired probability that the RANSAC algorithm provides a useful result after running. RANSAC returns a successful result if in some iteration it selects only inliers from the input data set when it chooses the n points from which the model parameters are estimated. Let w be the probability of choosing an inlier each time a single point is selected, that is,

w = number of inliers in data / number of points in data

A common case is that w is not well known beforehand, but some rough value can be given. Assuming that the n points needed for estimating a model are selected independently, w^n is the probability that all n points are inliers and $1 - w^n$ is the probability that at least one of the n points is an outlier, a case which implies that a bad model will be estimated from this point set. That probability to the power of k is the probability that the algorithm never selects a set of n points which all are inliers and this must be the same as $1 - p$. Consequently,

$$1 - p = (1 - w^n)^k$$

which, after taking the logarithm of both sides, leads to

$$k = \frac{\log(1 - p)}{\log(1 - w^n)}$$

This result assumes that the n data points are selected independently, that is, a point which has been selected once is replaced and can be selected again in the same iteration. This is often not a reasonable approach and the derived value for k should be taken as an upper limit in the case that the points are selected without replacement. For example, in the case of finding a line which fits the data set illustrated in the above figure, the RANSAC algorithm typically chooses two points in each iteration and computes `maybe_model` as the line between the points and it is then critical that the two points are distinct.

To gain additional confidence, the standard deviation or multiples thereof can be added to k. The standard deviation of k is defined as

$$\mathrm{SD}(k) = \frac{\sqrt{1 - w^n}}{w^n}$$

https://en.wikipedia.org/wiki/Random_sample_consensus

Reweighted Least Squares

When the fraction of inliers > 50%

L^p norm linear regression [edit]

To find the parameters $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)^{\mathrm{T}}$ which minimize the L^p norm for the linear regression problem,

$$\arg\min_{\boldsymbol{\beta}} \left\|\mathbf{y} - X\boldsymbol{\beta}\right\|_p = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} \left|y_i - X_i\boldsymbol{\beta}\right|^p,$$

the IRLS algorithm at step $t + 1$ involves solving the weighted linear least squares problem:[4]

$$\boldsymbol{\beta}^{(t+1)} = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} w_i^{(t)} \left|y_i - X_i\boldsymbol{\beta}\right|^2 = \left(X^{\mathrm{T}} W^{(t)} X\right)^{-1} X^{\mathrm{T}} W^{(t)} \mathbf{y},$$

where $W^{(t)}$ is the diagonal matrix of weights, usually with all elements set initially to:

$$w_i^{(0)} = 1$$

and updated after each iteration to:

$$w_i^{(t)} = \left|y_i - X_i\boldsymbol{\beta}^{(t)}\right|^{p-2}.$$

https://en.wikipedia.org/wiki/Iteratively_reweighted_least_squares

When the fraction of inliers > 50%

where $W^{(t)}$ is the diagonal matrix of weights, usually with all elements set initially to:

$$w_i^{(0)} = 1$$

and updated after each iteration to:

$$w_i^{(t)} = \left|y_i - X_i \boldsymbol{\beta}^{(t)}\right|^{p-2}.$$

In the case $p = 1$, this corresponds to least absolute deviation regression (in this case, the problem would be better approached by use of linear programming methods,[5] so the result would be exact) and the formula is:

$$w_i^{(t)} = \frac{1}{\left|y_i - X_i \boldsymbol{\beta}^{(t)}\right|}.$$

To avoid dividing by zero, regularization must be done, so in practice the formula is:

$$w_i^{(t)} = \frac{1}{\max\left\{\delta, \left|y_i - X_i \boldsymbol{\beta}^{(t)}\right|\right\}}.$$

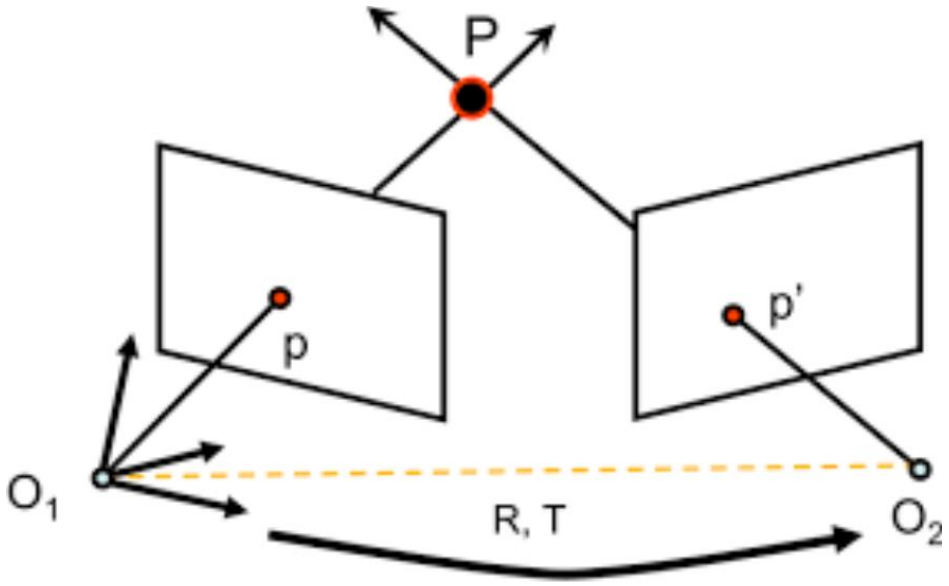
where δ is some small value, like 0.0001.[5] Note the use of δ in the weighting function is equivalent to the Huber loss function in robust estimation.

https://en.wikipedia.org/wiki/Iteratively_reweighted_least_squares

Bundle Adjustment

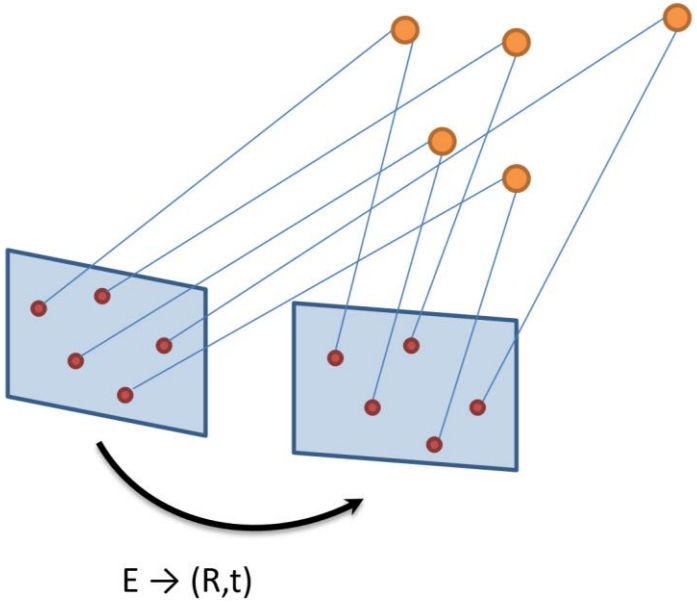
Recap: Structure-From-Motion

- Two views initialization:
  - 8-point linear algorithm



Recap: Structure-From-Motion

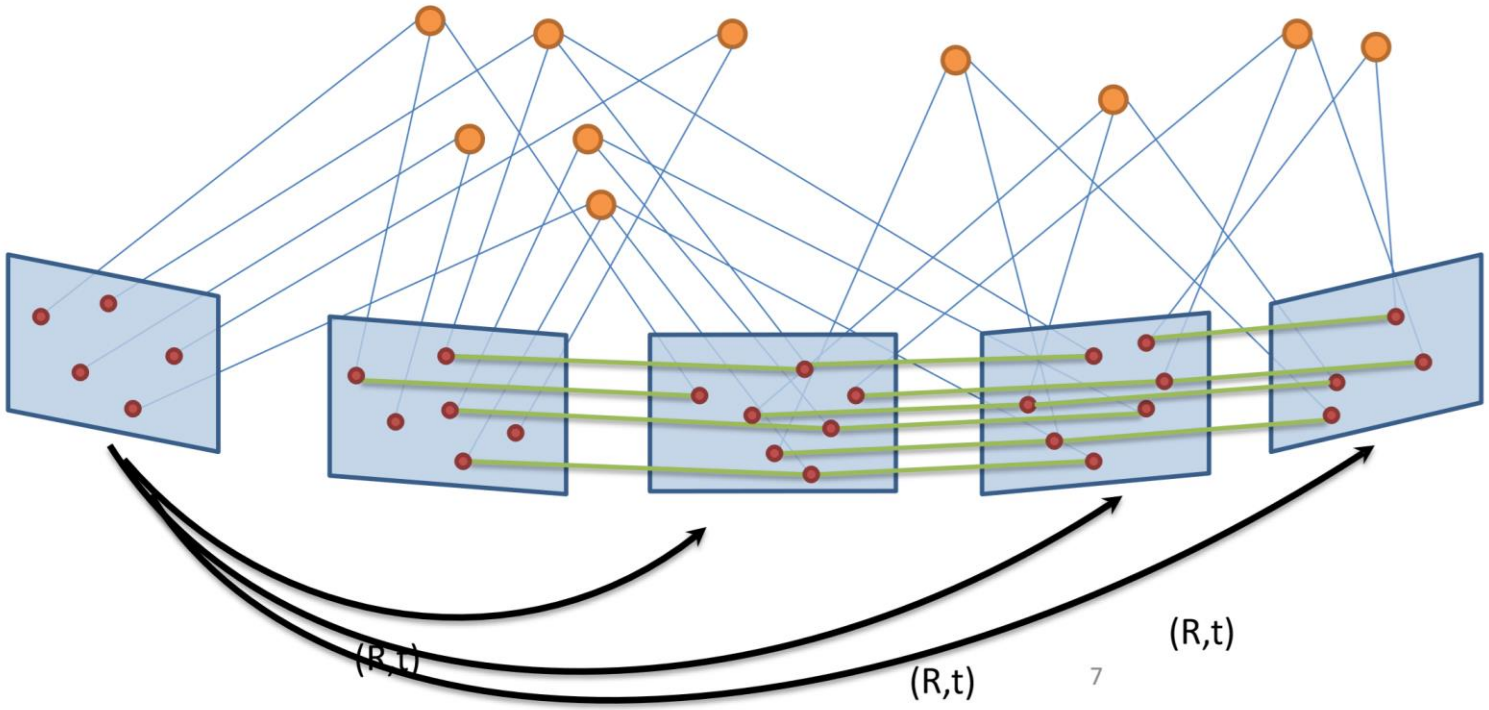
- Triangulation: 3D Points



Slide Credit: https://cvg.ethz.ch/teaching/3dvision/slides/class06eth18_annotated.pdf

Recap: Structure-From-Motion

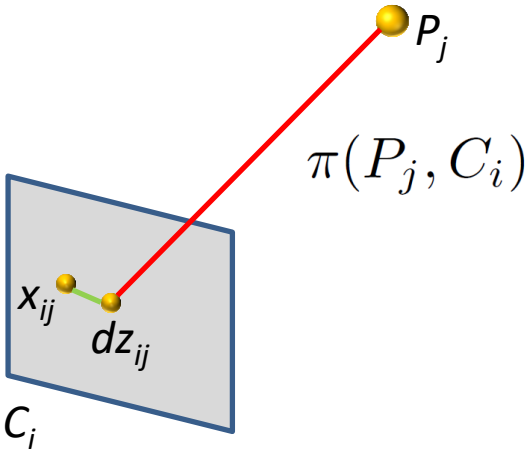
- Triangulation: 3D Points



Slide Credit: https://cvg.ethz.ch/teaching/3dvision/slides/class06eth18_annotated.pdf

Bundle Adjustment

- Refinement step in Structure-from-Motion
- Refine a visual reconstruction to produce jointly optimal 3D structures *P* and camera poses *C*
- Minimize total re-projection errors dz



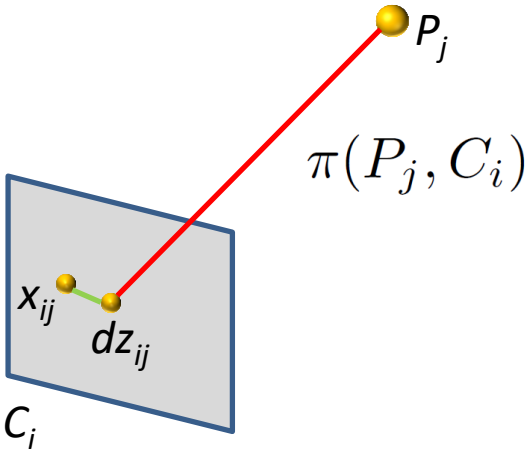
Cost Function:

$$\arg\min_X \sum_i \sum_j \|\boldsymbol{x}_{ij} - \pi(P_j, C_i)\|^2$$

$X = [P,C]$

Bundle Adjustment

- Refinement step in Structure-from-Motion
- Refine a visual reconstruction to produce jointly optimal 3D structures *P* and camera poses *C*
- Minimize total re-projection errors dz



Cost Function:

Measurement error covariance matrix

$$\arg\min_X \sum_i \sum_j dz_{ij}^T W_{ij} dz_{ij}$$

$$dz_{ij} = x_{ij} - \pi(P_j, C_i)$$

$X = [P,C]$

Bundle Adjustment

- Minimize the cost function: $\arg\min_X f(X)$
  - Gradient Descent
  - Newton Method
  - Gauss-Newton
  - Levenberg-Marquardt
  - All line search based techniques

Bundle Adjustment

- Gradient Descent

Initialization: $X_k = X_0$

Iterate until convergence

Compute gradient: $\boldsymbol{g} = \frac{\partial f(X)}{\partial X}|_{X=X_k} = d\boldsymbol{Z}^T W J$

Update: $X_k \leftarrow X_k - \eta \boldsymbol{g}$

Jacobi

Very slow convergence

Bundle Adjustment

- Newton Method

  2ndorder approximation (Quadratic Taylor Expansion):

$$f(\boldsymbol{x}+\delta) = f(\boldsymbol{x}) + \boldsymbol{g}^T\delta + \frac{1}{2}\delta^T H\delta$$

Hessian matrix: $H = \frac{\partial^2 f(\boldsymbol{x}+\delta)}{\partial^2 \delta}|_{X=X_k}$

$$X_k \leftarrow X_k - H^{-1}\boldsymbol{g}$$

H is expensive to compute, and H may not be positive definite

Bundle Adjustment

- Levenberg-Marquardt

  Regularized Gauss-Newton with damping factor λ

$$(J^T W J + \lambda I)\delta = -J^T W d\boldsymbol{z}$$

  $\lambda \to 0$:Gauss-Newton (When convergence is rapid)

  $\lambda \to \infty$:Gradient descent (When convergence is slow)

  Adaptin λ during optimization

  Decrease λ when function value decreases

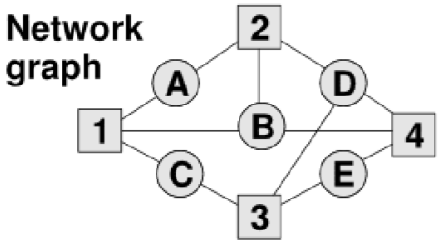
  Increase λ otherwise

  Global convergence!

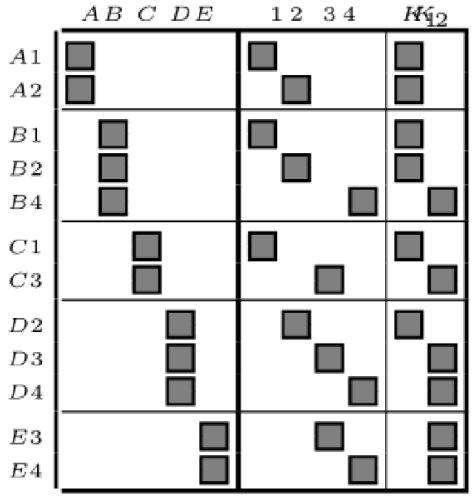
https://pdfs.semanticscholar.org/6048/ff93f92a865172dc621be1d318ba2c2ed244.pdf

Structure of the Jacobian and Hessian Matrices

- Sparse matrices since 3D structures are locally observed



$$J =$$

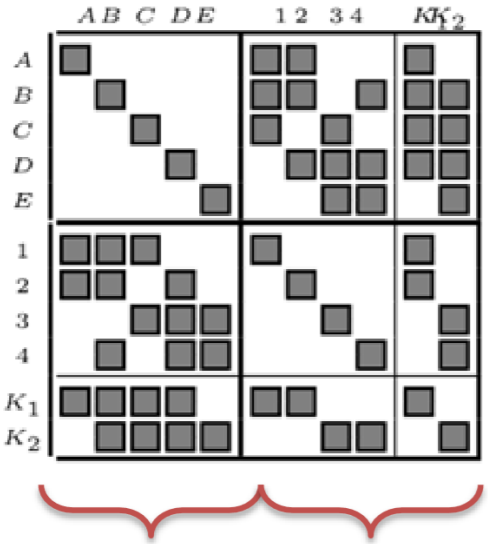


Efficiently Solving the Normal Equation

- Schur Complement: Exploit structure of H

$$H_{LM}\delta = -J^T W \Delta Z$$

$$H_{LM} =$$

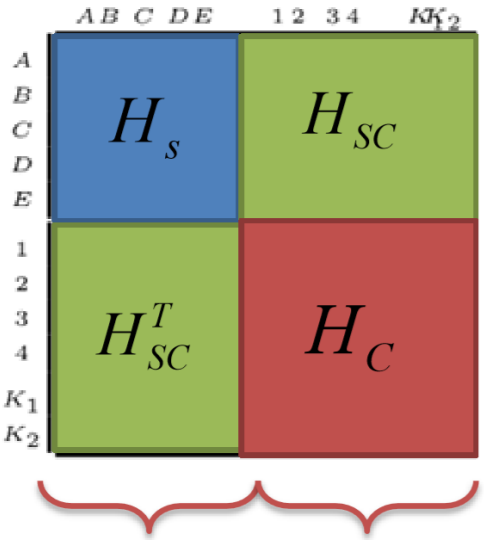


Efficiently Solving the Normal Equation

- SchurComplement: Exploit structure of H

$$H_{LM}\delta = -J^T W \Delta Z$$

$$H_{LM} =$$



Other Aspects

- Efficient solver of the linear system
  - Use the sparse structure
  - Prefactorization
- Robust cost function
- Iteratively re-weighted least-squares

State-of-the-art Solvers

Google Ceres:

- https://code.google.com/p/ceres-solver/

g2o:

- https://openslam.org/g2o.html

GTSAM:

- https://collab.cc.gatech.edu/borg/gtsam/

Multicore Bundle Adjustment

- http://grail.cs.washington.edu/projects/mcba/