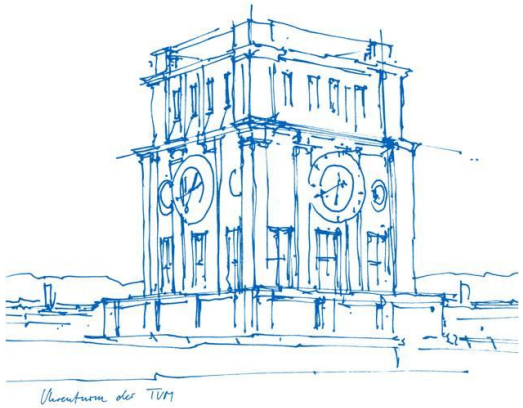
Learning for Self-Driving Cars and Intelligent Systems

Qadeer Khan, Mariia Gladkova

https://vision.in.tum.de/teaching/ws2021/intellisys_ws2021



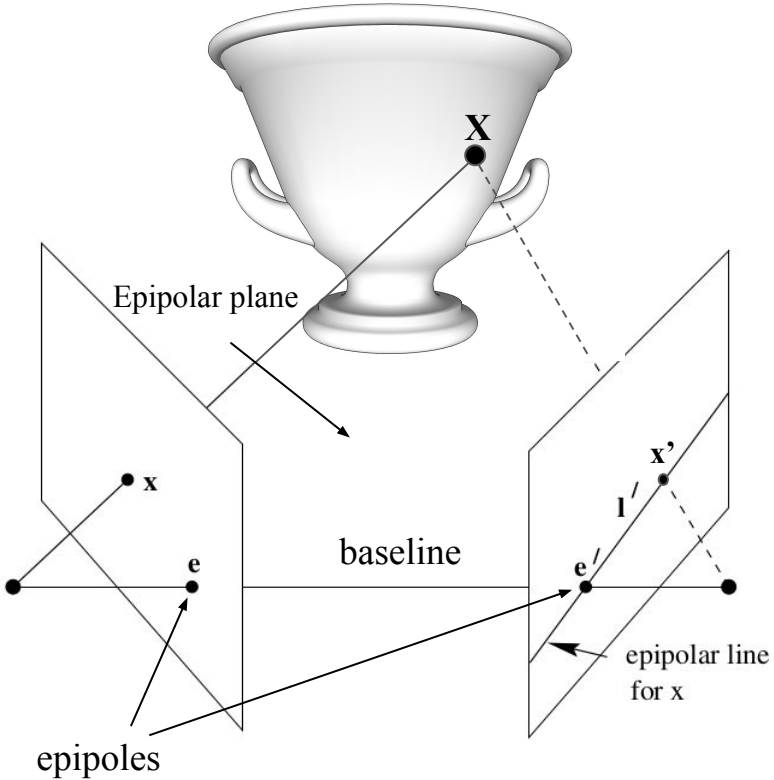
Reminder for Task 2

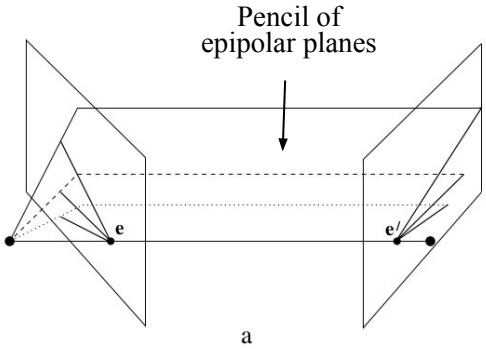
- Submission deadline for Task 2 is 08 November, 2021 (end of day i.e. 23:59)

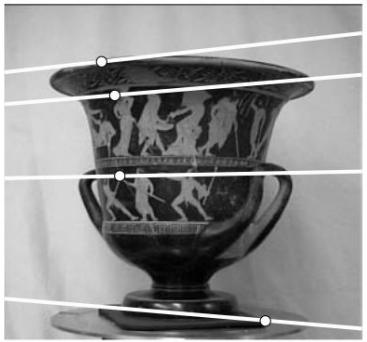
Contents of today’s lecture

- Two-View Geometry
  - **Homography** from week 1 addresses a specialized two-view case of points on a plane
- N-View Methods
  - Bundle Adjustment

Epipolar Geometry

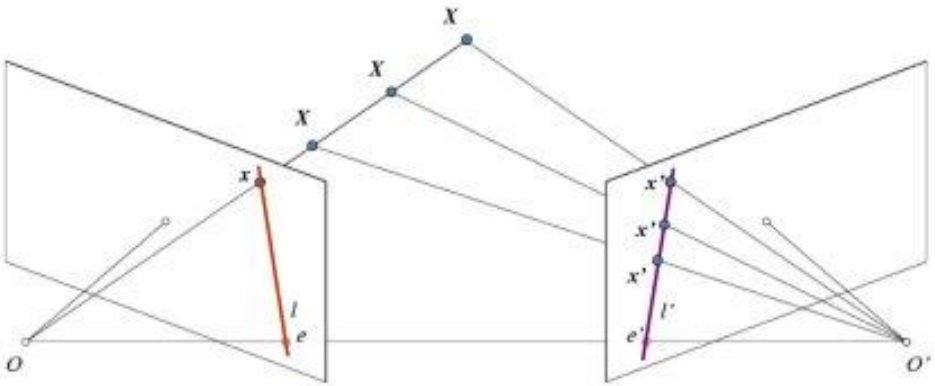






[1] Zisserman, Richard Hartley Andrew. "Multiple view geometry in computer vision." (2004).
[2] https://www.turbosquid.com/3d-models/greek-pottery-ancient-3d-model-1705279

Epipolar Geometry



- Epipolar constraint (Fundamental matrix **F**)

$$\boldsymbol{x}^T \underbrace{\boldsymbol{F}\boldsymbol{x}'}_{\text{epipolar line } \boldsymbol{l}} = 0$$

- Correspondences $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^3$
- Properties of $\boldsymbol{F} \in \mathbb{R}^{3\times 3}$
  - 7 DoF
  - rank 2

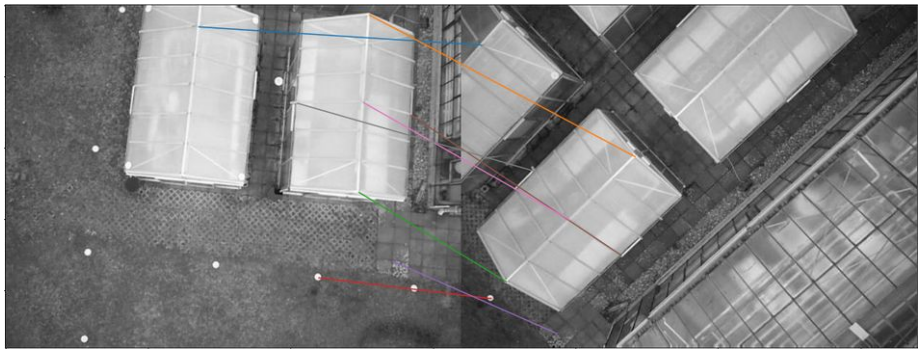
[1] Zisserman, Richard Hartley Andrew. "Multiple view geometry in computer vision." (2004).
[2] https://docs.opencv.org/4.5.0/da/de9/tutorial_py_epipolar_geometry.html

From Correspondences to Relative Pose

$$\boldsymbol{F} = \boldsymbol{K}^{-T}\underbrace{\hat{\boldsymbol{T}}\boldsymbol{R}}\boldsymbol{K}'^{-1}$$

Essential matrix $\boldsymbol{E} \in \mathbb{R}^{3\times3}$

$$\hat{\boldsymbol{T}} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

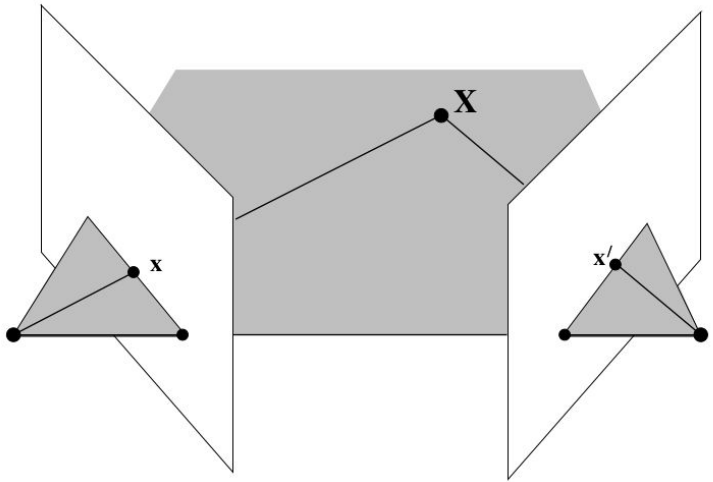


R, t (up-to a scale)

https://www.ipb.uni-bonn.de/online-training-pcv/

From Relative Pose to 3D Structure Estimation

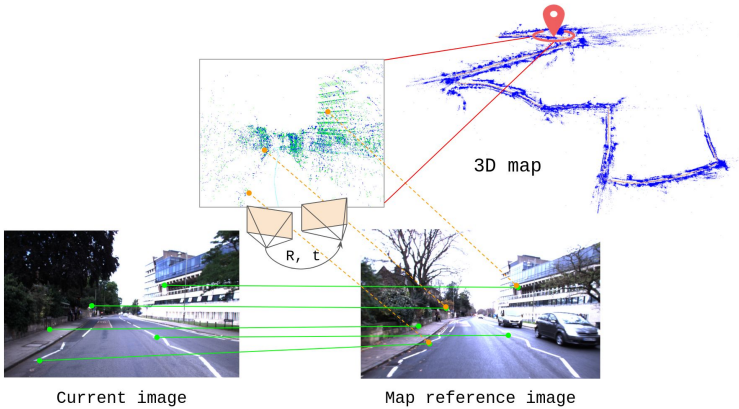
- Having calibration matrix **K** and relative motion transformation **[R | t]** one can compute 3D coordinates of a point **X** corresponding to a pair x <-> x’ (aka triangulation)
  - Slide 13: Additional material: Linear Triangulation Method

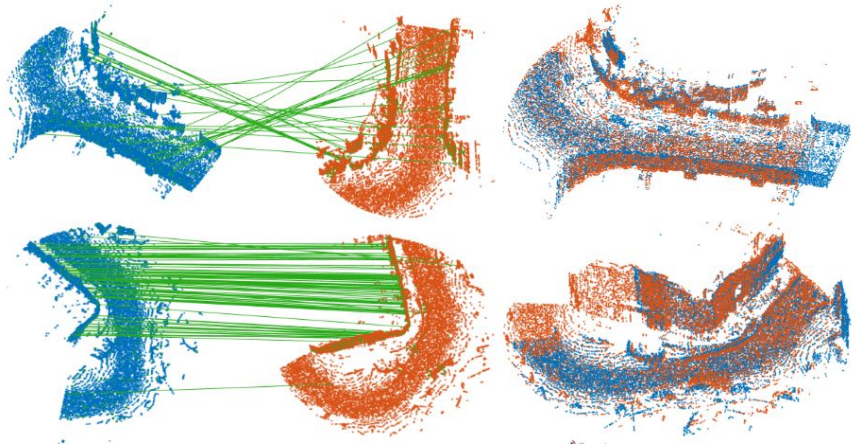


Zisserman, Richard Hartley Andrew. "Multiple view geometry in computer vision." (2004).

From Correspondences to Relative Pose

- So far: 2D - 2D correspondences are used to recover camera motion
- Alternatively, one can utilize
  - 3D - 2D correspondences (e.g. Perspective-n-Point (PnP) algorithm)
  - 3D - 3D correspondences (e.g. Iterative Closest Point (ICP) algorithm)

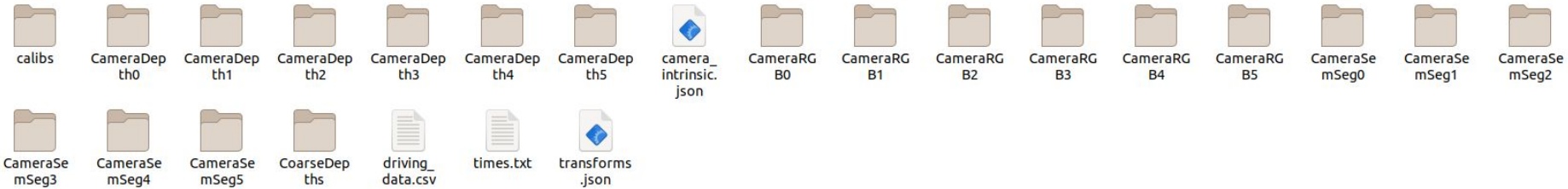




[1] M. Gladkova, R. Wang, N. Zeller and D. Cremers, "Tight Integration of Feature-based Relocalization in Monocular Direct Visual Odometry," https://arxiv.org/pdf/2102.01191.pdf
[2] Du, Juan, Rui Wang, and Daniel Cremers. "Dh3d: Deep hierarchical 3d descriptors for robust large-scale 6dof relocalization." https://arxiv.org/pdf/2007.09217.pdf

GN-Net Benchmark

- Available on remote machines
  - /storage/group/intellisys/datasets/carla/GNNET_BENCHMARK_PUBLIC/
- Navigator view
  - /storage/group/intellisys/datasets/carla/GNNET_BENCHMARK_PUBLIC/carla_training_validation/all_weathers/episode_000/



[1] Von Stumberg, Lukas, et al. "GN-Net: The gauss-newton loss for multi-weather relocalization." *https://arxiv.org/pdf/1904.11932.pdf*
[2] Supplementary material + GN-Net Benchmark: https://vision.in.tum.de/research/vslam/gn-net

GN-Net Benchmark

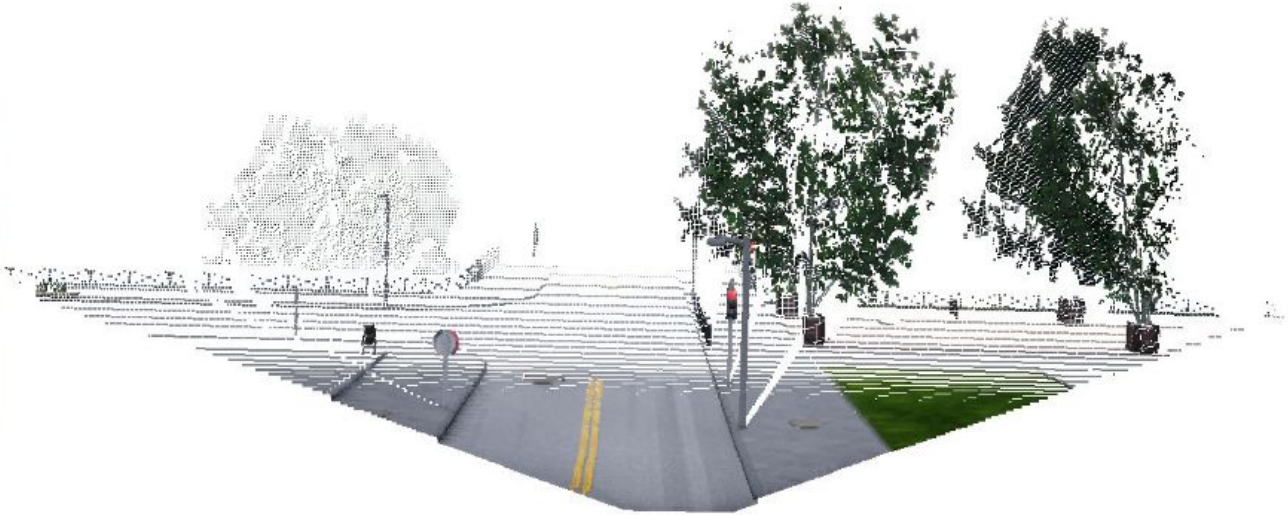
- Relocalization tracking benchmark accompanied the GN-Net paper
- CARLA-generated dataset
- Data is divided into training, validation (**carla_training_validation** subfolder) and testing (**carla_testset** subfolder) splits
- Multi-weather and one-weather conditions
- Each episode contains information from 6 cameras (labeled 0 to 5) and it is 500-frames long
- Training and validation splits:
  - Each camera has RGB, depth and semantic segmentation maps
  - Camera intrinsics is in **camera_intrinsic.json**
  - Camera extrinsics is in **transforms.json**

Data visualization

https://github.com/carla-simulator/driving-benchmarks/blob/master/version084/carla/image_converter.py

2D image

3D Cordinates
point cloud



Quantitative Evaluation of Pose Accuracy

- Rotation matrices are elements of a Lie group SO(3), transformation matrices are elements of a Lie group SE(3) -> cannot simply use addition and subtraction as operations

$$\Delta R = R_{\mathrm{gt}}^T R_{\mathrm{e}} \qquad \Delta t = R_{\mathrm{gt}}^T (t_{\mathrm{e}} - t_{\mathrm{gt}})$$

- Or simply with transformation matrices:

$$\Delta T = T_{\mathrm{gt}}^{-1} T_{\mathrm{e}} = \begin{bmatrix} R_{\mathrm{gt}}^T & -R_{\mathrm{gt}}^T t_{\mathrm{gt}} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} R_{\mathrm{e}} & t_{\mathrm{e}} \\ \mathbf{0} & 1 \end{bmatrix}$$

Quantitative Evaluation of Pose Accuracy

- Rotation matrices are elements of a Lie group SO(3), transformation matrices are elements of a Lie group SE(3) -> cannot simply use addition and subtraction as operations

$$\Delta R = R_{\mathrm{gt}}^{T} R_{\mathrm{e}} \qquad \Delta t = R_{\mathrm{gt}}^{T}(t_{\mathrm{e}} - t_{\mathrm{gt}})$$

- Rotation error: transform the rotation error to angle-axis representation and use absolute angle value

$$|\theta| = \arccos\left(\frac{\Delta R.\mathrm{trace} - 1}{2}\right)$$

- Translational error: consider the Euclidean norm, i.e. $||\Delta t||_2$

Quantitative Evaluation of Pose Accuracy

- Trajectory error
  - Aggregated metric for **n** frames
  - Commonly only translational part of an error is considered, since rotational error is correlated [1]
  - Trajectories need to be aligned before comparison since their origin can differ

[1] Sturm, Jürgen, et al. "A benchmark for the evaluation of RGB-D SLAM systems." https://vision.in.tum.de/_media/spezial/bib/sturm12iros.pdf
[2] Choice of the metrics: ATE and number of frames used for initialization are based on the evaluation done by Xingwei Qu in his master thesis *Initialization Methods for Visual and Visual-inertial SLAM*

Quantitative Evaluation of Pose Accuracy

- Absolute Trajectory Error (ATE)
- Relative Pose Error (RPE)

$$ATE_{1...n} = \left(\frac{1}{n} \sum_{i=1}^{n} ||transl(F_i)||_2^2 \right)^{\frac{1}{2}} \quad \text{where} \quad F_i = Q_i^{-1} S P_i$$

Ground truth pose

Estimated pose

Alignment matrix

$$RPE_{1...n} = \left(\frac{1}{n-1} \sum_{i=1}^{n-1} ||transl(E_{i,i+1})||_2^2 \right)^{\frac{1}{2}} \quad \text{where} \quad E_{i,j} = (Q_i^{-1} Q_j)^{-1} (P_i^{-1} P_j)$$

Δ Time difference between two frames: 1, 3, 7...

[1] Sturm, Jürgen, et al. "A benchmark for the evaluation of RGB-D SLAM systems." https://vision.in.tum.de/_media/spezial/bib/sturm12iros.pdf
[2] Choice of the metrics: ATE and number of frames used for initialization are based on the evaluation done by Xingwei Qu in his master thesis *Initialization Methods for Visual and Visual-inertial SLAM*

Quantitative Evaluation of 3D Reconstruction Accuracy

- Root Mean Squared Error (RMSE)

$$\sqrt{\frac{1}{|\boldsymbol{X}|}\sum_{\boldsymbol{X}}(z - z_{\mathrm{gt}})^2}$$

- RMSE with re-scaling (when reconstruction is up-to a projective scale)

$$\sqrt{\frac{1}{|\boldsymbol{X}|}\sum_{\boldsymbol{X}}(z * \mathrm{scale} - z_{\mathrm{gt}})^2} \qquad \mathrm{scale} \approx \frac{median(\boldsymbol{z}_{gt})}{median(\boldsymbol{z})}$$

Exercise 3.1 Essential Matrix and Triangulation

1. Utilize algorithms from exercise 1 to extract features and find matches between two consecutive frames of a sequence of your choice
   a. handcrafted or SuperPoint features
   b. CARLA (e.g. GN-Net benchmark)
2. Estimate essential matrix and decompose it into a rotation matrix and translation vector
   a. Make use of functions in the OpenCV library
3. Estimate 3D coordinates of a point per feature match
4. Visualize estimated pointcloud. You can also utilize ground truth depth maps and visualize ground truth 3D reconstruction. What do you observe? Is such difference expected? Why or why not?
5. **Bonus:** compute fundamental matrix **F** using calibration matrix **K** and the equation on slide 6, verify that epipolar constraint (equation on slide 5) is fulfilled using the computed feature correspondences x and x'

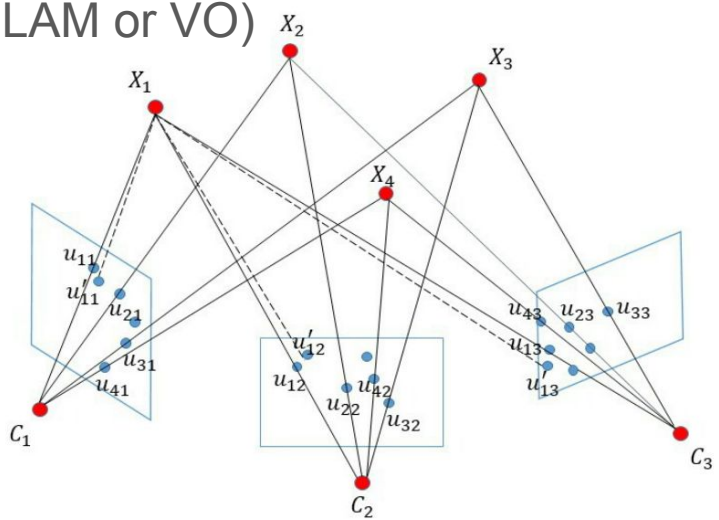
Exercise 3.1 Submission guidelines

1. Your program **ex3_1.py** should accept two images as input and output relative rotational and translational errors
   a. We will test it with images from GN-Net benchmark and supplied ground truth extrinsics
2. Make sure to provide qualitative results (images, pointcloud visualizations) to support your solution!

Bundle Adjustment

- Generalization from 2 to N views and accountancy to measurement noise
- Nonlinear least-squares problem conventionally solved using Gauss-Newton or Levenberg-Marquardt algorithms
- Final step in reconstruction pipeline (e.g. SLAM or VO)

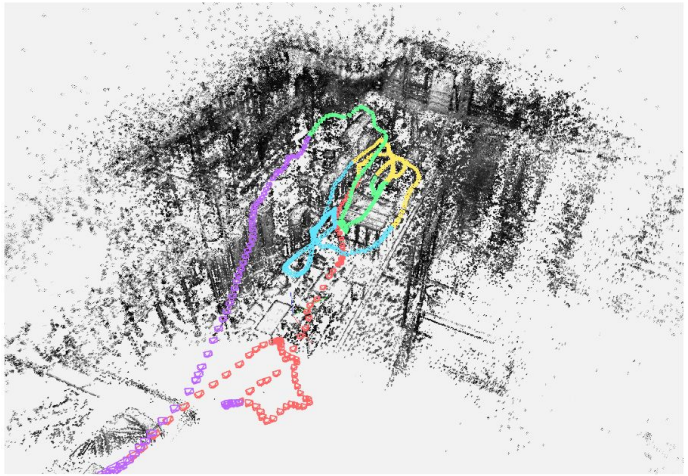
$$min \sum_{i=1}^{n} \sum_{j=1}^{m} (u_{ij} - \pi(C_j, X_i))^2$$



Chen, Yu, Yisong Chen, and Guoping Wang. "Bundle adjustment revisited." https://arxiv.org/pdf/1912.03858.pdf

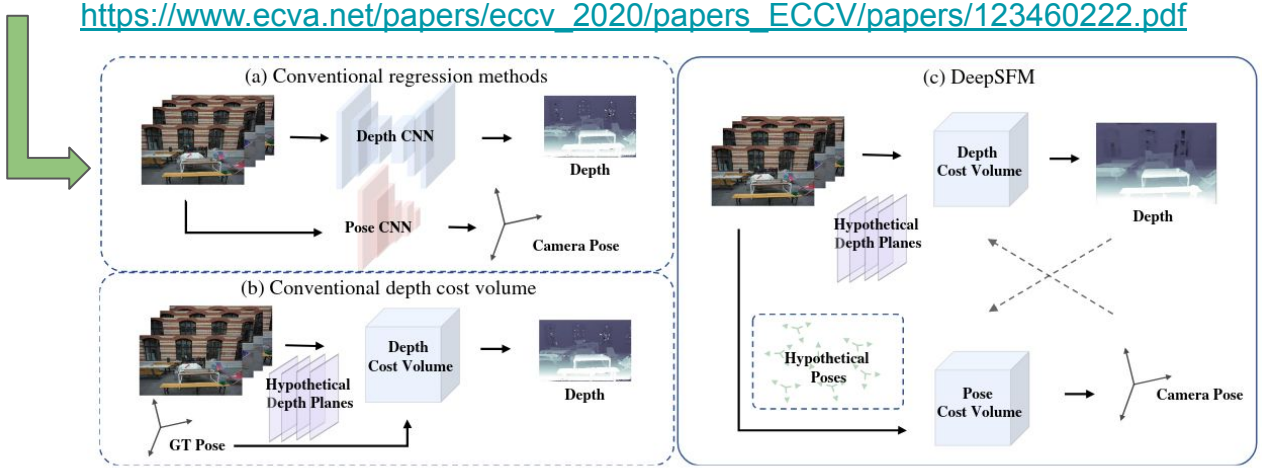
Bundle Adjustment

- Requires good initialization
- Large-scale BA is challenging and memory demanding
  - Sliding-window BA for SLAM real-time applications
  - Distributed BA
  - Various formulations (QR decomposition, preconditioned conjugate-gradients and etc.)
- Related works
  - Demmel, Nikolaus, et al. "Square Root Bundle Adjustment for Large-Scale Reconstruction." https://vision.in.tum.de/_media/spezial/bib/demmel2021rootba.pdf
  - Weber, Simon, et al. "Multidirectional Conjugate Gradients for Scalable Bundle Adjustment." https://vision.in.tum.de/_media/spezial/bib/weber2021mcg.pdf
  - Demmel, Nikolaus, et al. "Distributed Photometric Bundle Adjustment." https://vision.in.tum.de/_media/spezial/bib/demmel2020distributed.pdf



Deep Bundle Adjustment and SLAM

- Steps towards a differentiable pipeline
- Few related works:
  - Tang, Chengzhou et al. "BA-Net: Dense Bundle Adjustment Network." https://arxiv.org/pdf/1806.04807.pdf
  - Jatavallabhula, Krishna Murthy, et al. "gradSLAM: Automagically differentiable SLAM." https://arxiv.org/pdf/1910.10672.pdf
  - Wei, Xingkui, et al. "DeepSFM: Structure From Motion via Deep Bundle Adjustment." https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123460222.pdf



Resources and Libraries for DL + 3D

- PyTorch3D https://github.com/facebookresearch/pytorch3d
- LieTorch https://github.com/princeton-vl/lietorch
- Kornia https://github.com/kornia/kornia
- Open-source code of related research projects
  - https://github.com/gradslam/gradslam
  - https://github.com/weixk2015/DeepSFM

Exercise 3.2 Differential Small-Scale BA

1. Perform Bundle Adjustment using PyTorch3D or LieTorch, i.e. iteratively estimate optimal camera poses and 3D points given a set of feature correspondences.
2. Use data from GN-Net benchmark dataset
   a. Suggestion: select 5 - 7 consecutive frames
3. Run exercise 3.1. on the chosen input image sequence for BA initialization.
   a. Suggestion: choose the first frame as a reference (i.e. fix its pose as identity) and estimate relative poses and pointcloud wrt to the first frame (i.e. compute **E** for camera pairs (1,2), (1,3), (1,4) and etc.).
4. Visualize results in your report.

Hints:

1. If your choose PyTorch3D, provided tutorials might be useful as a starting point
   a. E.g. https://pytorch3d.org/tutorials/bundle_adjustment
2. You can utilize losses, which are robust to outliers, e.g. Huber loss.
3. Keep in mind the coordinate systems and choose a reference frame for cameras and points.

Exercise 3.2 Submission guidelines

1. Submission deadline for Task 3 (both parts 3.1 and 3.2) is 15 November, 2021 (end of the day, i.e. 23:59)
2. Your program **ex3_2.py** would be tested as in the example

> python ex3_2.py --seq PATH_TO_GNNET_BENCHMARK/episode_0000/ --start 254 --end 261 --cam 0

The program should output to stdout **relative pose error** and **RMSE** for 3D reconstruction of sparse points.

3. Make sure to provide qualitative results (images, pointcloud visualizations) in your report to support your solution!

Thank you for your attentions? Any questions?

Additional material: 8-Point Algorithm for Essential Matrix

```
1  function E = E_from_point_pairs(xs, xss)
2  % xs, xss: Nx3 homologous point coordinates, N > 7
3  % E:       3x3 essential matrix
4
5  % coefficient matrix
6  for n = 1 : size(xs, 1)
7      A(n, :) = kron(xss(n, :), xs(n, :));
8  end
9
10 % singular value decomposition
11 [U, D, V] = svd(A);
12
13 % approximate E, possibly regular
14 Ea = reshape(V(:, 9), 3, 3);
15
16 % svd decomposition of E
17 [Ua, Da, Va] = svd(Ea);
18
19 % algebraically best E, singular, sa
20 E = Ua * diag([1, 1, 0]) * Va';
```

build matrix A

solve Ae=0

build matrix Ea

compute SVD of Ea

build matrix E from Ea by imposing constraints

- Rank 2
- Two identical eigenvalues
- Defined up to a scale

https://www.ipb.uni-bonn.de/html/teaching/msr2-2020/sse2-18-fe-direct.pdf

Additional material: Linear Triangulation Method

- Minimization of the algebraic error $\boldsymbol{x}_i \times \boldsymbol{P}\boldsymbol{X}_i = \mathbf{0}$
  $\boldsymbol{x'}_i \times \boldsymbol{P'}\boldsymbol{X}_i = \mathbf{0}$
- Reformulating the objective to a familiar problem $\boldsymbol{A}\boldsymbol{X}_i = 0$ for unknown 3D point coordinates Xi, where (u_i, v_i) are 2D coordinates in the first image, (u'_i, v'_i) - 2D coordinates in the second view, **P** and **P'** are projection matrices of first and second image respectively

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{p}^{1T} \\ \boldsymbol{p}^{2T} \\ \boldsymbol{p}^{3T} \end{bmatrix} \qquad \boldsymbol{A} = \begin{bmatrix} v_i\boldsymbol{p}^{3T} - \boldsymbol{p}^{2T} \\ u_i\boldsymbol{p}^{3T} - \boldsymbol{p}^{1T} \\ v'_i\boldsymbol{p'}^{3T} - \boldsymbol{p'}^{2T} \\ u'_i\boldsymbol{p'}^{3T} - \boldsymbol{p'}^{1T} \end{bmatrix}$$