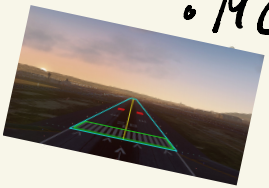
Today: My summer work & future research

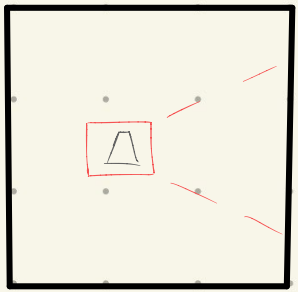
Romeo Valentin
Summer '23
romeov@stanford.edu

- Motivation
- The PNP problem
  - Least-Squares
  - Fully Bayesian
  - MCMC

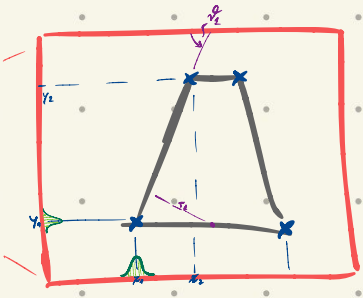


- Software Framework
  - Automatic Diff.
  - Unitful type system
- Faster NN through SoftArgMax

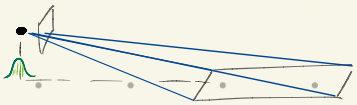
The VLS Problem



Stage 1:
Find runway



Stage 2:
Extract runway parametrization



Step 3:
Solve position

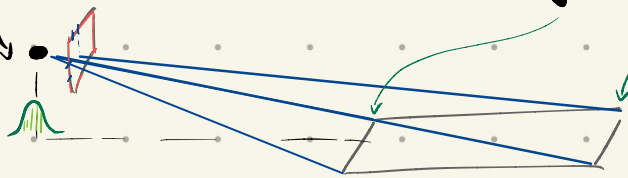
Why camera?

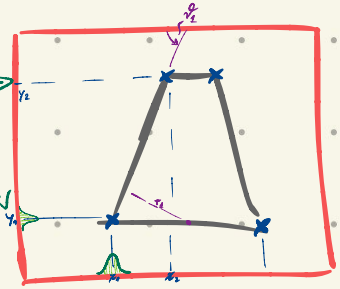
Why uncertainty?

Useful features?

Sensitivity to errors?

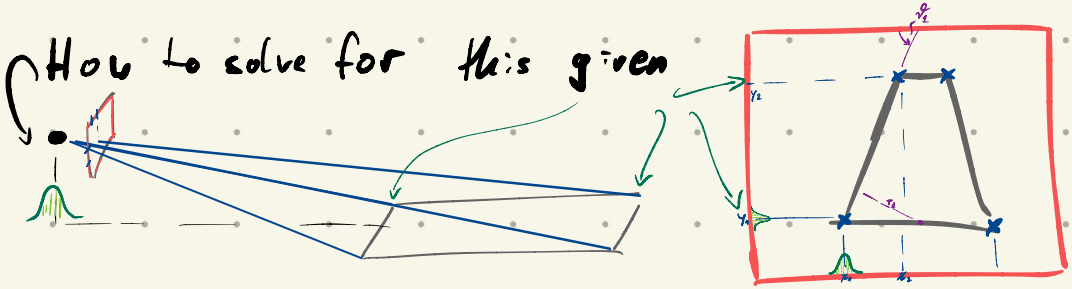
How to solve for this given





First approach:

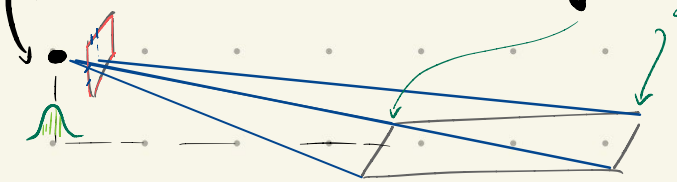
How to solve for this given

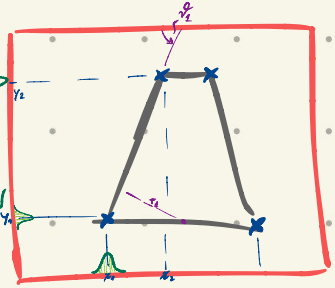


First approach:

$$\underset{x}{\arg\min} \left[project(corner_1) - measured(corner_1) \right]^2 + \left[project(corner_2) - measured(corner_2) \right]^2 + \cdots + \left[angle_{proj}(lhs) - angle_{measured}(lhs) \right]^2 + \cdots$$

How to solve for this given





First approach:

"simulation" ! location observations from stage 2

$$\underset{\vec{x}}{\operatorname{argmin}} \sum_i \left[f_i(\vec{x}) - o_i \right]^2$$

Problems:

- sensitivity of minimum?
- $[1m]^2 + [1°]^2$?!
- using uncertainties?
- producing uncertainties?
- weighting terms by "influence"?
- correlated errors?

Task: Study usefulness of sideline angles.

Approach 1: OpenCV PnP solver.

18:50 Tue 14 Nov

OpenCV 4.8.0-dev
Open Source Computer Vision

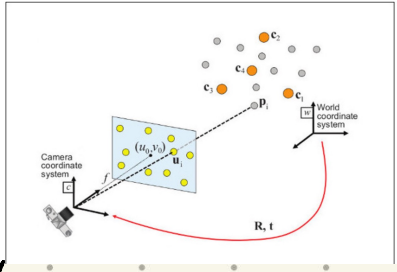
Main Page | Related Pages | Modules | Namespaces | Classes | Files | Examples | Java documentation | Search

Perspective-n-Point (PnP) pose computation

Pose computation overview

The pose computation problem [178] consists in solving for the rotation and translation that minimizes the reprojection error from 3D-2D point correspondences.

The `solvePnP` and related functions estimate the object pose given a set of object points, their corresponding image projections, as well as the camera intrinsic matrix and the distortion coefficients, see the figure below (more precisely, the X-axis of the camera frame is pointing to the right, the Y-axis downward and the Z-axis forward).



18:51 Tue 14 Nov

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Pose computation methods

Refer to the **cv::SolvePnPMethod** enum documentation for the list of possible values. Some details about each method are d

- **cv::SOLVEPNP_ITERATIVE** Iterative method is based on a Levenberg-Marquardt optimization. In this case the functio minimizes reprojection error, that is the sum of squared distances between the observed projections "imagePoints" and **cv::projectPoints**) "objectPoints". Initial solution for non-planar "objectPoints" needs at least 6 points and uses the DL planar "objectPoints" needs at least 4 points and uses pose from homography decomposition.
- **cv::SOLVEPNP_P3P** Method is based on the paper of X.S. Gao, X.-R. Hou, J. Tang, H.-F. Chang "Complete Solution C Three-Point Problem" ([96]). In this case the function requires exactly four object and image points.
- **cv::SOLVEPNP_AP3P** Method is based on the paper of T. Ke, S. Roumeliotis "An Efficient Algebraic Solution to the Pe ([141]). In this case the function requires exactly four object and image points.
- **cv::SOLVEPNP_EPNP** Method has been introduced by F. Moreno-Noguer, V. Lepetit and P. Fua in the paper "EPnP: E Camera Pose Estimation" ([153]).
- **cv::SOLVEPNP_DLS** **Broken implementation. Using this flag will fallback to EPnP.** Method is based on the paper of J. Hesch and S. Roumeliotis. "A Direct Least-Squares (DLS) Method for PnP" ([122]).
- **cv::SOLVEPNP_UPNP** **Broken implementation. Using this flag will fallback to EPnP.** Method is based on the paper of A. Penate-Sanchez, J. Andrade-Cetto, F. Moreno-Noguer. "Exhaustive Linearization fo Focal Length Estimation" ([209]). In this case the function also estimates the parameters f_x and f_y assuming that both cameraMatrix is updated with the estimated focal length.
- **cv::SOLVEPNP_IPPE** Method is based on the paper of T. Collins and A. Bartoli. "Infinitesimal Plane-Based Pose Estim requires coplanar object points.
- **cv::SOLVEPNP_IPPE_SQUARE** Method is based on the paper of Toby Collins and Adrien Bartoli. "Infinitesimal Plane- This method is suitable for marker pose estimation. It requires 4 coplanar object points defined in the following order:
  - point 0: [-squareLength / 2, squareLength / 2, 0]
  - point 1: [squareLength / 2, squareLength / 2, 0]
  - point 2: [squareLength / 2, -squareLength / 2, 0]
  - point 3: [-squareLength / 2, -squareLength / 2, 0]
- **cv::SOLVEPNP_SQPNP** Method is based on the paper "A Consistently Fast and Globally Optimal Solution to the Persp Terzakis and M.Lourakis ([263]). It requires 3 or more points.

P3P

The cv::solveP3P() computes an object pose from exactly 3 3D-2D point correspondences. A P3P problem has up to 4 soluti

Essentially:

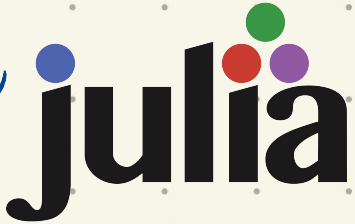
$$\underset{g}{\operatorname{argmin}} \left[\text{project}(\text{corner}_i) - \text{measured}(\text{corner}_i) \right]^2$$

→ hand-computed gradients
→ no weights → points only

→ no angles !?

Approach 2: Write own (flexible) PnP solver.

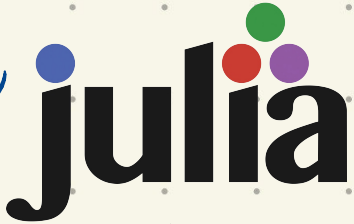
- projection code
- automatic differentiation, 2nd or third! order (!)
- trust-region Newton method or Levenberg-Marquardt method



→ recent funding by Boeing, Chan-Zuckerberg foundation, Williams F1 team, & others

Approach 2: Write own (flexible) PnP solver.

- projection code
- automatic differentiation, 2nd or third! order (!)
- trust-region Newton method or Levenberg-Marquardt method



→ recent funding by Boeing, Chan-Zuckerberg foundation, Williams F1 team, & others

---

Two challenges left

1) How to add [px], [m], [°], etc... ?

2) How much error do we typically make ?

"simulation"

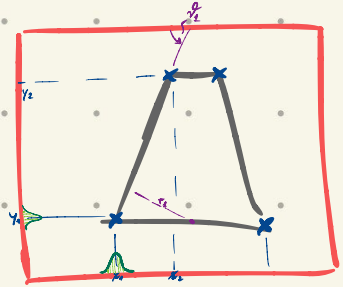
location

observations from stage 2

$$\underset{x}{\operatorname{argmin}} \sum_i \left[f_i(x) - o_i \right]^2$$

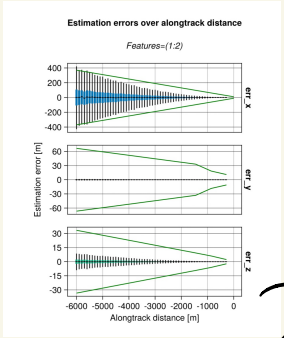
$$\underset{x}{\operatorname{argmin}} \sum_i \left[\frac{f_i(x) - o_i}{\sigma} \right]^2$$

typical error
standard deviation

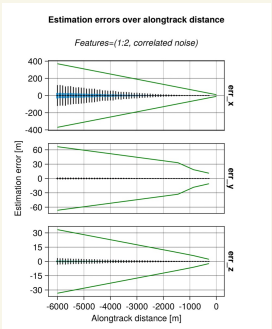


★Demo time★

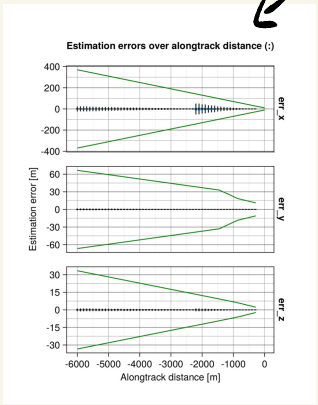
Some results:



correlated noise



multiple runways



adding angles

σ_{angle}	0.0°	0.01°	0.1°	0.3°	0.5°	1.0°	no angles
σ_x	101.22	104.61	105.89	110.58	122.37	159.51	156.54
σ_y	0.3863	0.3967	0.3986	0.4267	0.4869	0.5968	0.5678
σ_z	2.1189	2.1929	2.2021	2.3073	2.5945	3.4132	3.3251

no improvement for $\sigma_{angle} > 1°$

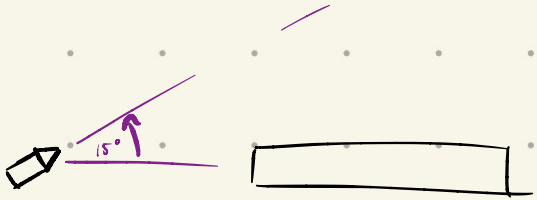
~33% for $\sigma_{angle} < 0.1°$

More results:

Influence of misalignment

σ_{angle}	0.0°	5.0°	10.0°	15.0°	20.0°	25.0°
σ_x	3.622	4.034	15.41	43.16	62.25	71.97
σ_y	0.319	0.350	4.641	32.24	46.76	55.61
σ_z	0.327	0.333	1.242	14.29	21.11	27.26

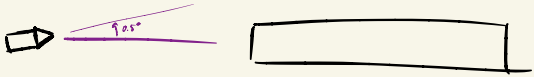
~10× at 15°



Influence of attitude error

σ_{angle}	0.0°	0.01°	0.1°	0.3°	0.5°	1.0°
σ_x	146.39	148.52	144.93	149.27	150.55	150.22
σ_y	0.5724	0.8271	6.1943	17.536	30.030	58.797
σ_z	3.1291	3.1956	6.8751	16.831	29.688	57.163

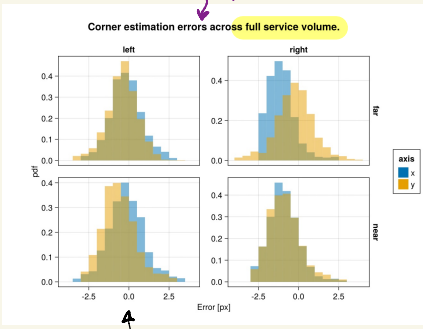
10–60× error increase if attitude is wrong 0.5°



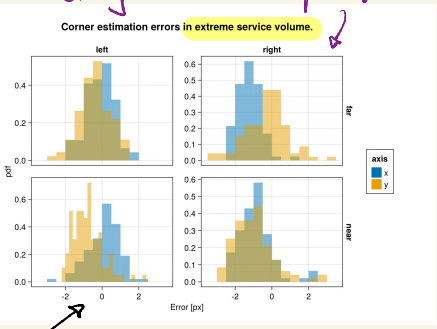
POV you're launching in Australia.

Interlude: Empirical error distribution

~1'300 samples



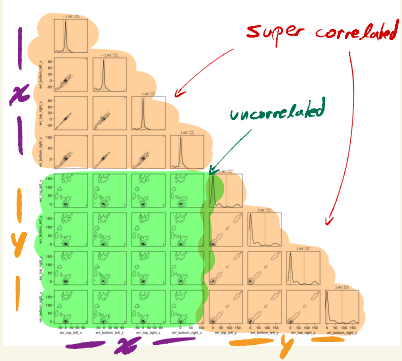
only ~130 samples ?



Approx. 1pxl of std

Takeaways:

- more hand-labeled data in extreme service vol.
- huge discrepancy between auto-labels & hand-labels
- 1 pxl of std error is pretty good
- huge error correlation is noteworthy



Other topics & continuing research

- analytic approach to sensitivity
- more principled hsq
- fully Bayesian "one-shot" pose uncertainty
- pose uncertainty "ground truth" through MCMC

- Stage 2 NN design w/ big speedup & easy uncertainties.

SISL
Stanford Intelligent Systems Laboratory