3D Computer Vision

Local optimization and state of the art

Ivan Marković

University of Zagreb Faculty of Electrical Engineering and Computing
Department of Control and Computer Engineering
Laboratory for Autonomous Systems and Mobile Robotics (lamor.fer.hr)

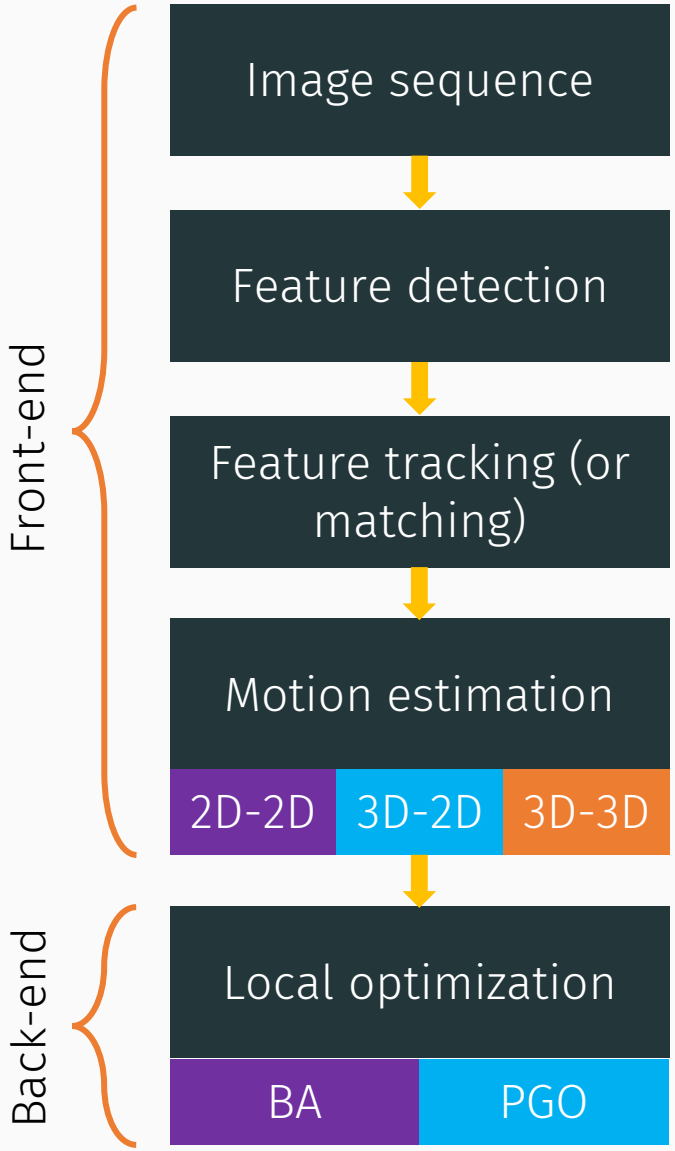
Outline

- Introduction
  - Reprojection error
- Non-linear optimization primer
- Local optimization
  - Bundle adjustment
  - Exploiting sparsity
  - Pose graph optimization
- Visual odometry state of the art
  - Feature-based and direct methods

Outline

- Introduction
  - Reprojection error
- Non-linear optimization primer
- Local optimization
  - Bundle adjustment
  - Exploiting sparsity
  - Pose graph optimization
- Visual odometry state of the art
  - Feature-based and direct methods

Local optimization for VO



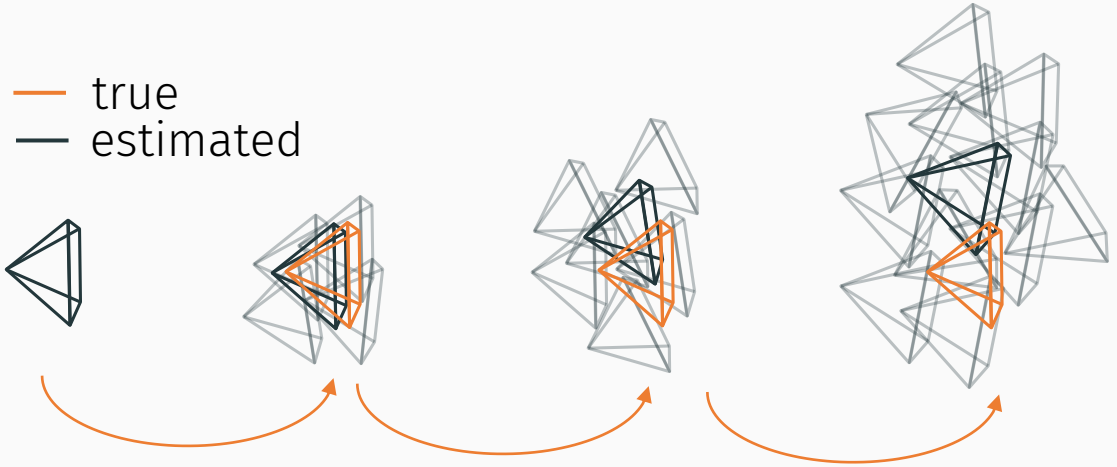
The front-end outputs a motion estimate from a pair of subsequent images (monocular or stereo) and the trajectory can be obtained by concatenating the sequence of transformations.

However, this small displacement concatenations lead to integration (summing up) of small displacement errors which over time can yield rather large final pose errors – we say that the odometry drifts.

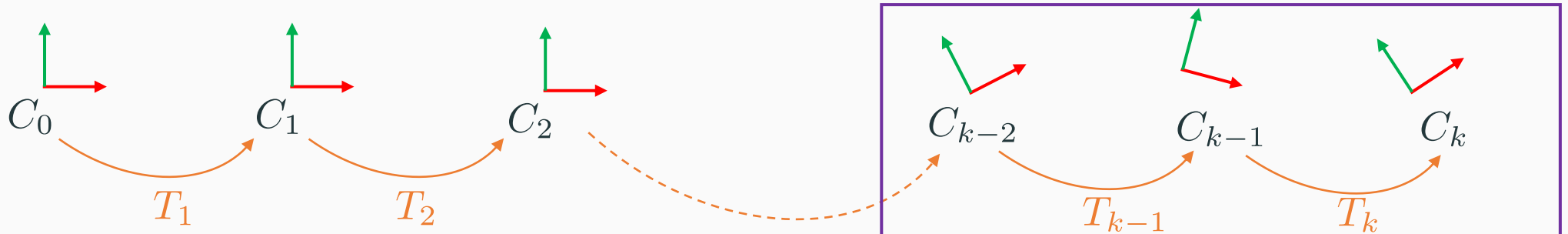
One strategy of addressing the drift issue is to perform local optimization on a window of past *m* frames that will reduce the drift and increase overall trajectory estimation accuracy.

Alleviating drift

Odometry inherently drifts and pose uncertainty increases with each subsequent frame.



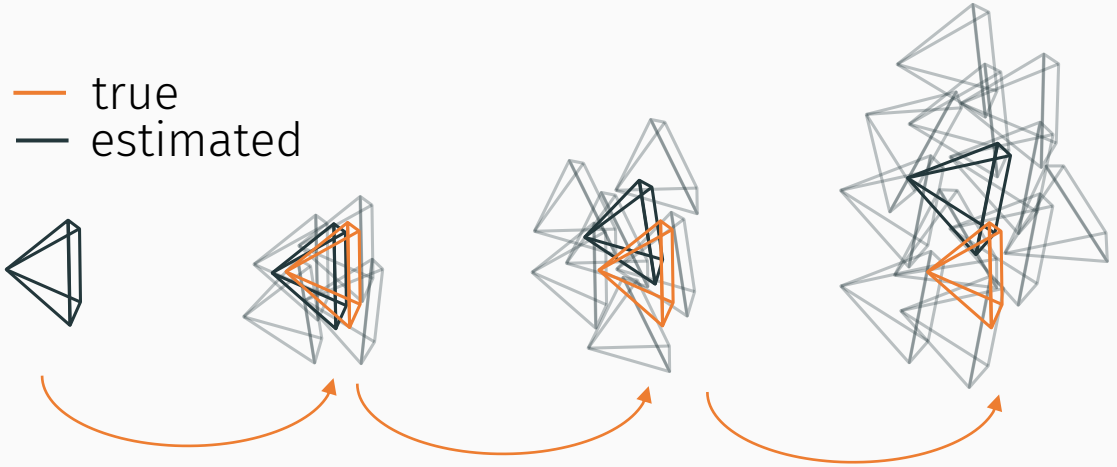
To alleviate effects of drift, we perform local optimization over a window of past *m* frames, thus enforcing local consistency.



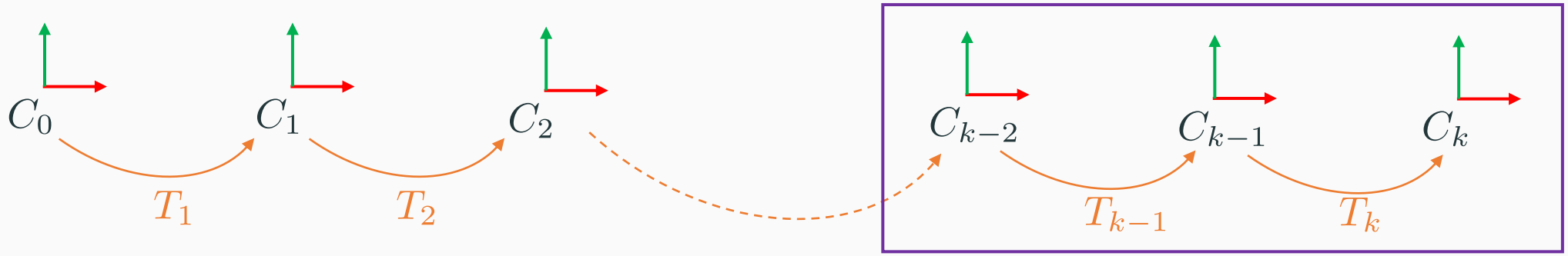
Local optimization over *m* frames

Alleviating drift

Odometry inherently drifts and pose uncertainty increases with each subsequent frame.

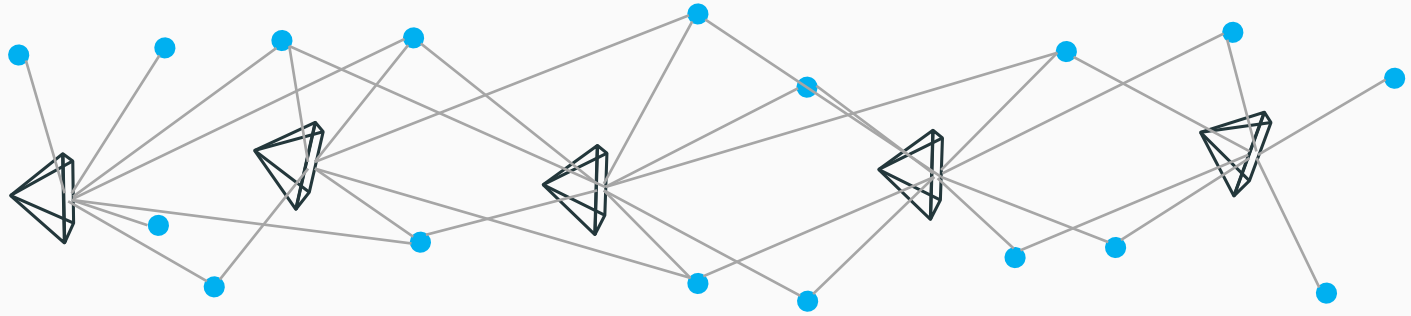


To alleviate effects of drift, we perform local optimization over a window of past *m* frames, thus enforcing local consistency.



Local optimization over *m* frames

BA and PGO

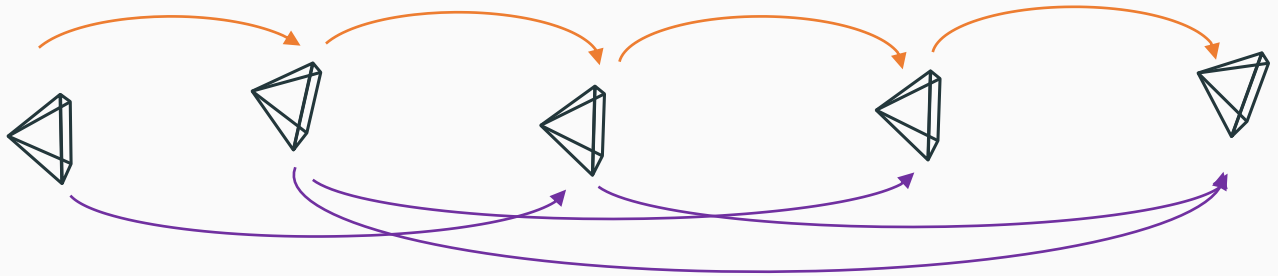


Local optimization

BA | PGO

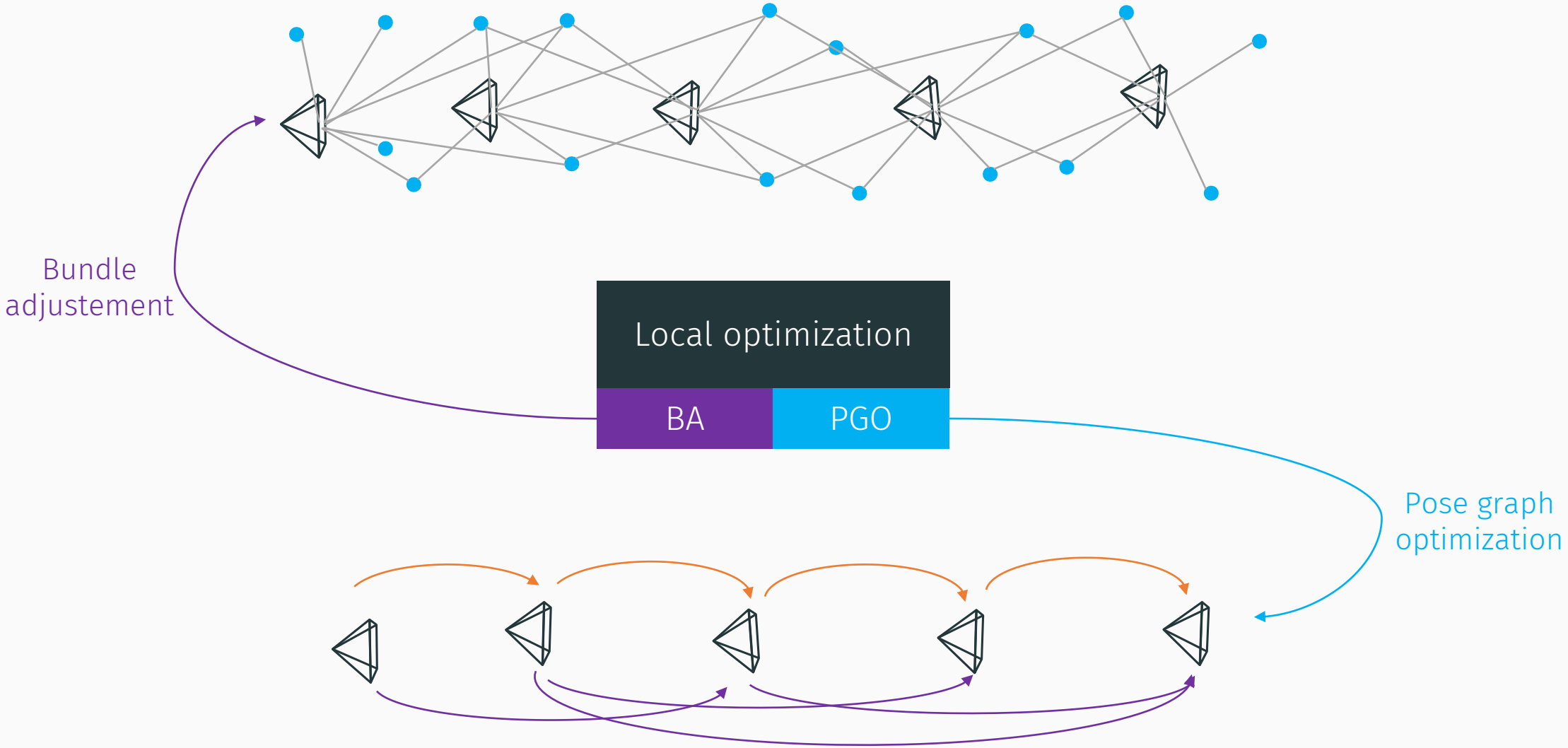
Bundle adjustement[1] - optimizes over camera poses (extrinsics) and triangulated features (structure, map) – can also include the intrinsic camera parameters

Pose graph optimization – optimizes only over camera poses



[1] B. Triggs; P. McLauchlan; R. Hartley; A. Fitzgibbon (1999). „Bundle Adjustment — A Modern Synthesis."

BA and PGO


Bundle
adjustement
Local optimization
BA
PGO
Pose graph
optimization

Reprojection function

Before formulating the BA problem, lets review first the projection equations

1. Transform the 3D point to the camera frame

$$X = RX' + t = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$$

2. Project the point to the normalized plane

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix}^T = \begin{bmatrix} X/Z & Y/Z & 1 \end{bmatrix}^T$$

3. Undistort the coordinates (here only radial distortion is modelled)

$$\delta x = x'(1 + k_1 r^2 + k_2 r^4), \quad x = x' + \delta x$$

$$\delta y = y'(1 + k_1 r^2 + k_2 r^4), \quad y = y' + \delta y$$

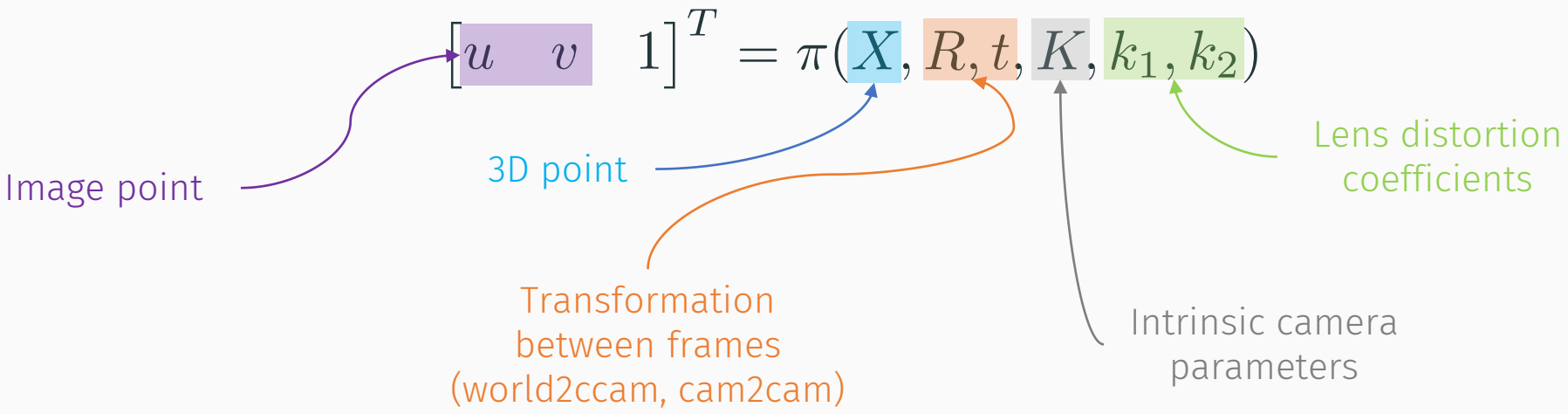
4. Compute the pixel coordinates using intrinsic parameters

$$\begin{bmatrix} u & v & 1 \end{bmatrix}^T = K \begin{bmatrix} x & y & 1 \end{bmatrix}^T \Rightarrow \begin{matrix} u = f_x x + u_0 \\ v = f_y y + v_0 \end{matrix}$$

5. We summarize the whole process in a single reprojection function

$$\begin{bmatrix} u & v & 1 \end{bmatrix}^T = \pi(X, R, t, K, k_1, k_2)$$

Reprojection function

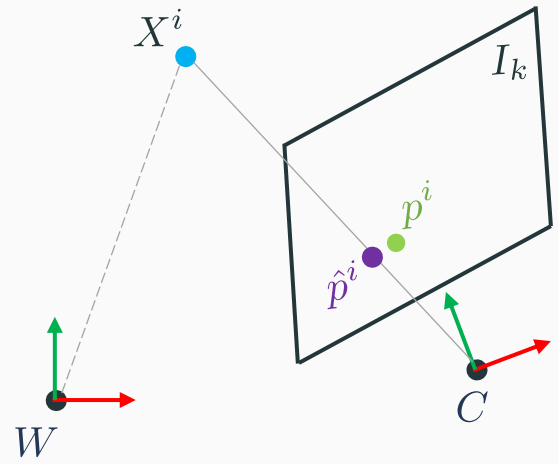


The reprojection function $\pi(\cdot)$ requires all of the above parameters but when it is used in an optimization problem, usually only the unknown optimization variables are stated, e.g., for the BA we have $\pi(X, R, t)$.

In essence, $\pi(\cdot)$ takes a 3D point defined in some reference coordinate frame (world, previous, left/right camera frame) and projects it to the image plane to obtain its pixel coordinates in the image.

Reprojection error

Let's assume that we have detected in image I_k feature p^i whose 3D coordinates X^i in the world frame we know. Using $\pi(\cdot)$ we can comptue projection of X^i to the image plane in pixel coordinates $\hat{p}^i$.



For this example we assume that rotation, translation and camera parameters are known.

Given that, we can define an error as the difference of the detected p^i and projected $\hat{p}^i$ point coordinates

$$e^i = p^i - \hat{p}^i = p^i - \pi(X^i, R, t).$$

We call this error the reprojection error, since the 3D points are triangulated from feature correspondences and then *reprojected* to the image.

When used as the *cost function* in optimization problems it calls for non-linear optimization techniques.

Outline

- Introduction
  - Reprojection error
- Non-linear optimization primer
- Local optimization
  - Bundle adjustment
  - Exploiting sparsity
  - Pose graph optimization
- Visual odometry state of the art
  - Feature-based and direct methods

Non-linear least squares

In many instances targeted methods involve non-linear optimization. Indeed, many are initialized with closed-form solutions (e.g., 3D-2D with P3P) and then refined via non-linear optimization (e.g., by minimizing the reprojection error).

Non-linear optimization is a field of its own but here we will look into the specific case of the non-linear least squares

$$x^* = \arg\min_x \sum_i ||z_i - f_i(x)||^2.$$

optimal state value

measurements (data)

Non-linear function $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m$

Linear least squares

In case f is linear $f_i(x) = A_i x$, the problem boils down to least squares

$$x^* = \arg\min_x \sum_i ||z_i - A_i x||^2 = \arg\min_x ||z - Ax||^2$$

$$\begin{aligned}\arg\min_x ||z - Ax||^2 &= \arg\min_x (z - Ax)^T (z - Ax) \\ &= \arg\min_x \left(x^T A^T A x - z^T A x - x^T A^T z + z^T z\right) \\ &= \arg\min_x F(x).\end{aligned}$$

To solve this problem we simply minimize the error *F(x)*

$$\frac{\partial F(x)}{\partial x} = 2A^T A x - 2z^T A x = 0 \Rightarrow x^* = (A^T A)^{-1} A^T z$$

This is the standard linear least squares solution.

Non-linear iterative optimization

Unlike linear least squares, non-linear problems cannot be solved directly but require an iterative solution starting from a suitable initial estimate.

A variety of algorithm exist that differ in how they locally approximate the nonlinearity but they all share the following basic structure:

1. Form an initial estimate x_0
2. Repeat
    1. In each iteration calculate the increment Δx
    2. Update the next estimate $x^{k+1} \leftarrow x^k + \Delta x$
3. Until convergence criteria are reached

Non-linear minimization

In the non-linear case

$$x^* = \arg\min_x \sum_i ||z_i - f_i(x)||^2 = \arg\min_x ||z - f(x)||^2$$

$$\begin{aligned}\arg\min_x ||z - f(x)||^2 &= \arg\min_x (z - f(x))^T (z - f(x)) \\ &= \arg\min_x \left(f(x)^T f(x) - z^T f(x) - f(x)^T z + z^T z \right) \\ &= \arg\min_x F(x).\end{aligned}$$

To solve this problem we again aim to minimize the error *F(x)*

$$\frac{\partial F(x)}{\partial x} = 2\frac{\partial f(x)^T}{\partial x} f(x) - 2\frac{\partial f(x)^T}{\partial x} z = 2J(x)^T f(x) - 2J(x)^T z = 0.$$

Non-linear function Jacobian

$$\frac{\partial F(x)}{\partial x} = 2J(x)^T f(x) - 2J(x)^T z = 0,$$

Where *J(x)* is the Jacobian matrix containing partial derivatives with respect to the vector *x*

$$J(x) = \begin{bmatrix} J_1(x) \\ \vdots \\ J_m(x) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Evidently, we cannot simply solve the equation for *x* as in the linear case.

Gauss-Newton method

The Gauss-Newton method approaches the optimization problem as follows.

Lets define a linearization point x_0 and the Jacobian matrix of $f(x)$ is

$$J(x_0) = \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_0}.$$

Now we can expand $f(x)$ using first-order Taylor expansion

$$f(x_0 + \Delta x) \approx f(x_0) + J(x_0)\Delta x.$$

The cost function *F(x)* now takes the form

$$\begin{aligned}\left.\frac{\partial F(x)}{\partial x}\right|_{x \leftarrow x_0} &= 2J(x_0)^T(f(x_0) + J(x_0)\Delta x) - 2J(x_0)^T z \\ &= 2J^T f(x_0) + 2J^T J\Delta x - 2J^T z \\ &= 2J^T J\Delta x - 2J^T(f(x_0) - z) = 2J^T J\Delta x - 2J^T b = 0\end{aligned}$$

Gauss-Newton method

Now the problem is one of linear least squares and the optimal solution is obtained by solving the following normal equation

$$J^T J \Delta x = J^T b.$$

In essence, this way we are solving the following optimization problem

$$\Delta x^* = \arg\min_{\Delta x} ||b - J\Delta x||^2$$

Computationally most efficient is Cholesky factorization (since $J^T J$ is symmetric), or QR and SVD factorization. If $J^T J$ is very large, then methods like pre-conditioned conjugate gradient can be used.

Gauss-Newton method

To sum up, the Gauss-Newton method iterates as follows

1. Form the initial estimate x_0
2. Repeat
   1. Calculate the Jacobian $J(x)|_{x=x_0}$ and the residual $z - f(x_0)$
   2. Solve the normal equation $J^T J \Delta x = J^T b$
   3. Update the state $x_{k+1} \leftarrow x_k + \Delta x$
   4. The updated state is the new initial estimate $x_0 \leftarrow x_{k+1}$
3. Until convergence criteria are reached ($\Delta x << $ or *max_iter*)

Gauss-Newton is widely used due to its effectivness but it can encounter problems if the local approximation with a quadratic is not good enough.

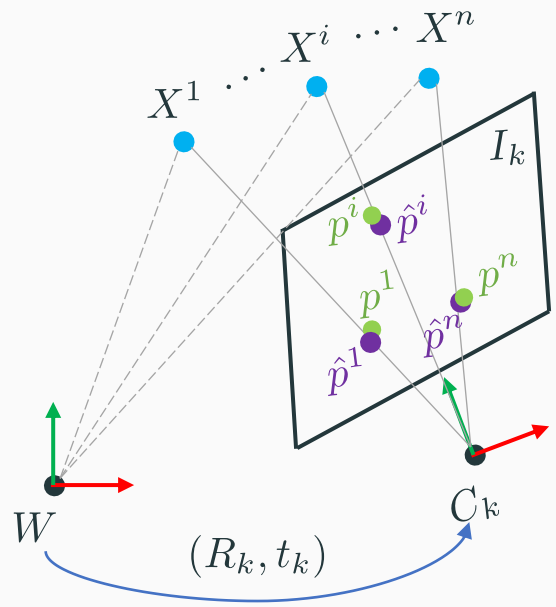
More robust approaches, at the expense of possibly slower convergence, can be used like trust region methods, e.g., Levenberg-Marquardt.

Outline

- Introduction
  - Reprojection error
- Non-linear optimization primer
- Local optimization
  - Bundle adjustment
  - Exploiting sparsity
  - Pose graph optimization
- Visual odometry state of the art
  - Feature-based and direct methods

Minimizing the reprojection error

For *n* points the reprojection error for frame C_k amounts to the following (as it is usually used in optimization, we take the sum of squared norms).



$$e_k = \sum_{i=1}^{n} ||p_k^i - \hat{p}_k^i||^2 = \sum_{i=1}^{n} ||p_k^i - \pi(X^i, R_k, t_k)||^2.$$

A non-linear error function

$$\sum_i ||z_i - f_i(x)||^2$$

Optimal parameters can then be found by minimizing the reprojection error using non-linear least squares.

Minimizing the reprojection error

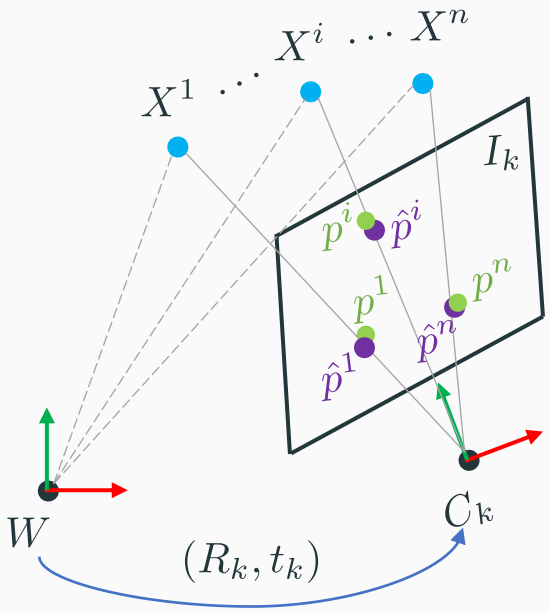
If we were solving for (R_k^*, t_k^*) by minimizing the reprojection error

$$(R_k^*, t_k^*) = \arg\min_{R_k, t_k} \sum_i ||p_k^i - \pi(X^i, R_k, t_k)||^2.$$

We the above equation would be solving the PnP problem via non-linear optimization.

In BA, we include also the 3D points as we are estimating the map (structure) of the scene

$$(R_k^*, t_k^*, X^*) = \arg\min_{R_k, t_k, X^{(1:N)}} \sum_i ||p_k^i - \pi(X^i, R_k, t_k)||^2.$$

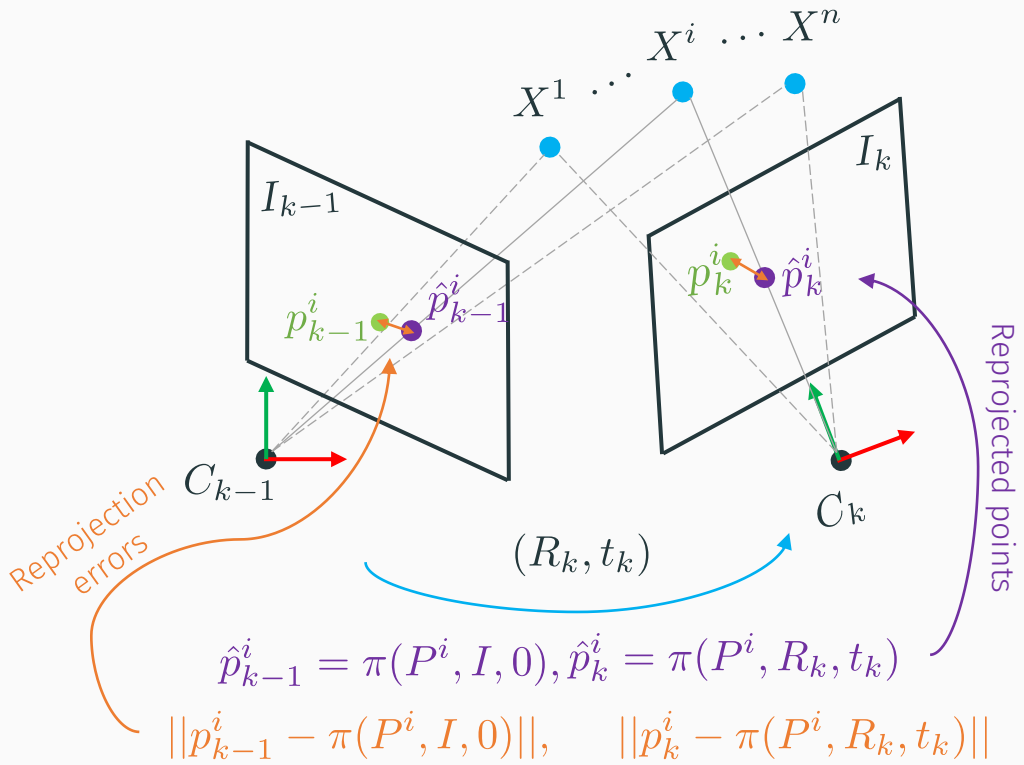


Two-view BA

As a minimal example let's have a look at a two view BA, which, again, is a non-linear, joint optimization of the map (structure) $X^i, \forall i = 1, \ldots, N$ and motion (R_k, t_k).

Commonly initialized with the result from the 5- or 8-point algorithm.

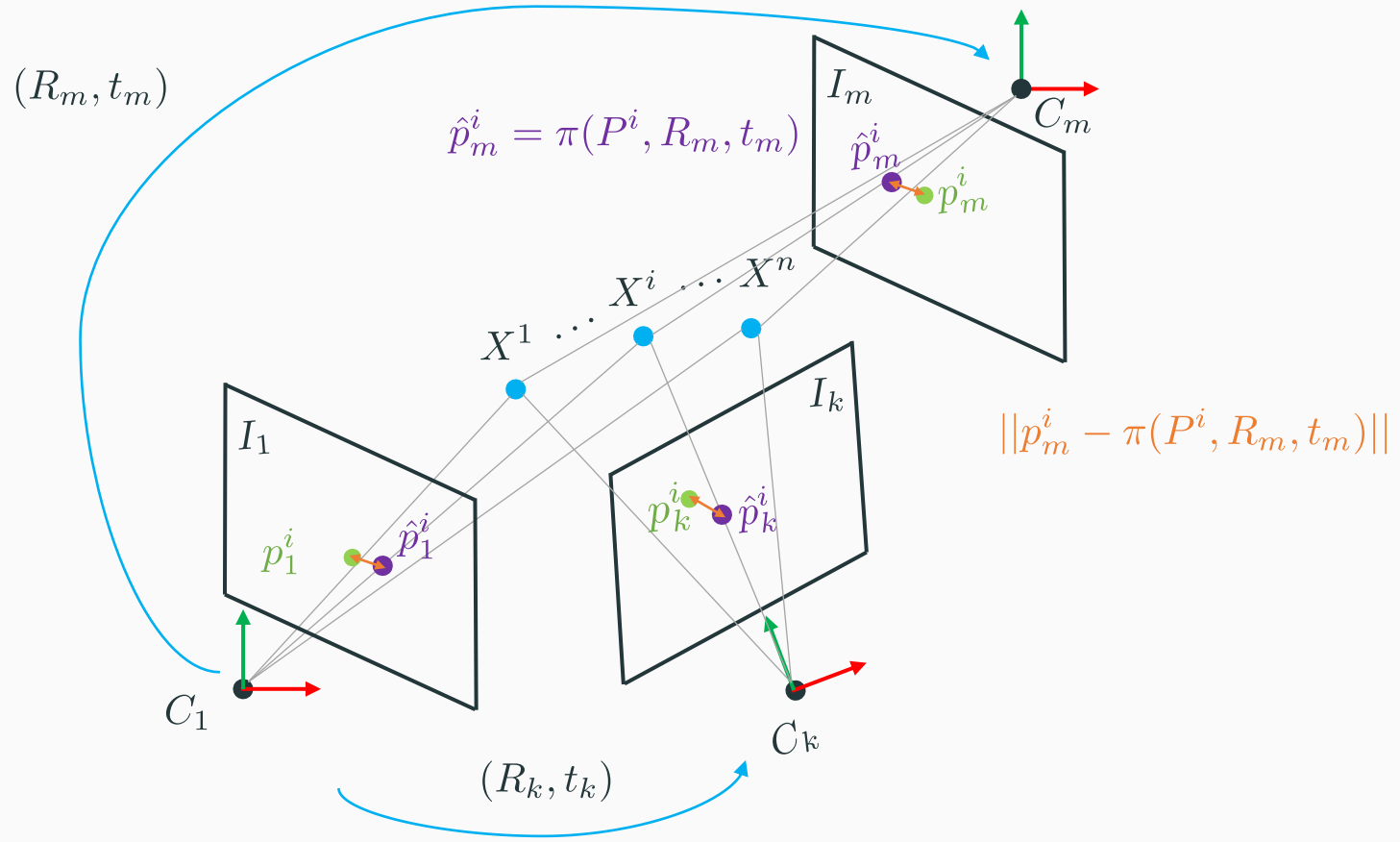
Solution is found by minimizing the sum of squared reprojection errors



$$(R_k^*, t_k^*, X^*) = \underset{R_k, t_k, X^{(1:n)}}{\arg\min} \sum_{i=1}^{N} \left(||p^i_{k-1} - \pi(X^i, I, 0)||^2 + ||p^i_k - \pi(X^i, R_k, t_k)||^2 \right).$$

Left camera as origin, but this is arbitrary. Can also be modified for instrinsics.

Multi-view BA



Multi-view BA

BA over m frames is again a non-linear, joint optimization of the map (structure) $X^i, \forall i = 1, \ldots, N$ and camera poses $C_1 = (I, 0), \ldots$ $\ldots, C_k = (R_k, t_k), \ldots, C_m = (R_m, t_m)$ and we minimize the sum of squared reprojection errors across all views.

$$(C^*_{1:m}, X^*) = \underset{C_{1:m}, X^{(1:n)}}{\arg\min} \sum_{k=1}^{m} \sum_{i=1}^{N} ||p^i_k - \pi(X^i, C_k)||^2.$$

We assumed that the first camera is the origin, but this is a matter of choice. The non-linear least squares problem can be solved using, e.g., the Gauss-Newton or Levenberg-Marquardt algorithm.

Popular optimization frameworks that are often used in robotics to solve such problems include GTSAM, iSAM2, Ceres, and g2o.

Jacobian sparsity in BA

Our state vector *x* in the general BA case is

$$x = [C_1, \ldots, C_m, \quad X^1, \ldots, X^N]^T.$$

While the non-linear function (now for a specific point *i* and camera pose *k)*

$$f_k^i(x) = \pi(X^i, C_k)$$

The Jacobian for point *i* and camera pose *k* evaluates to

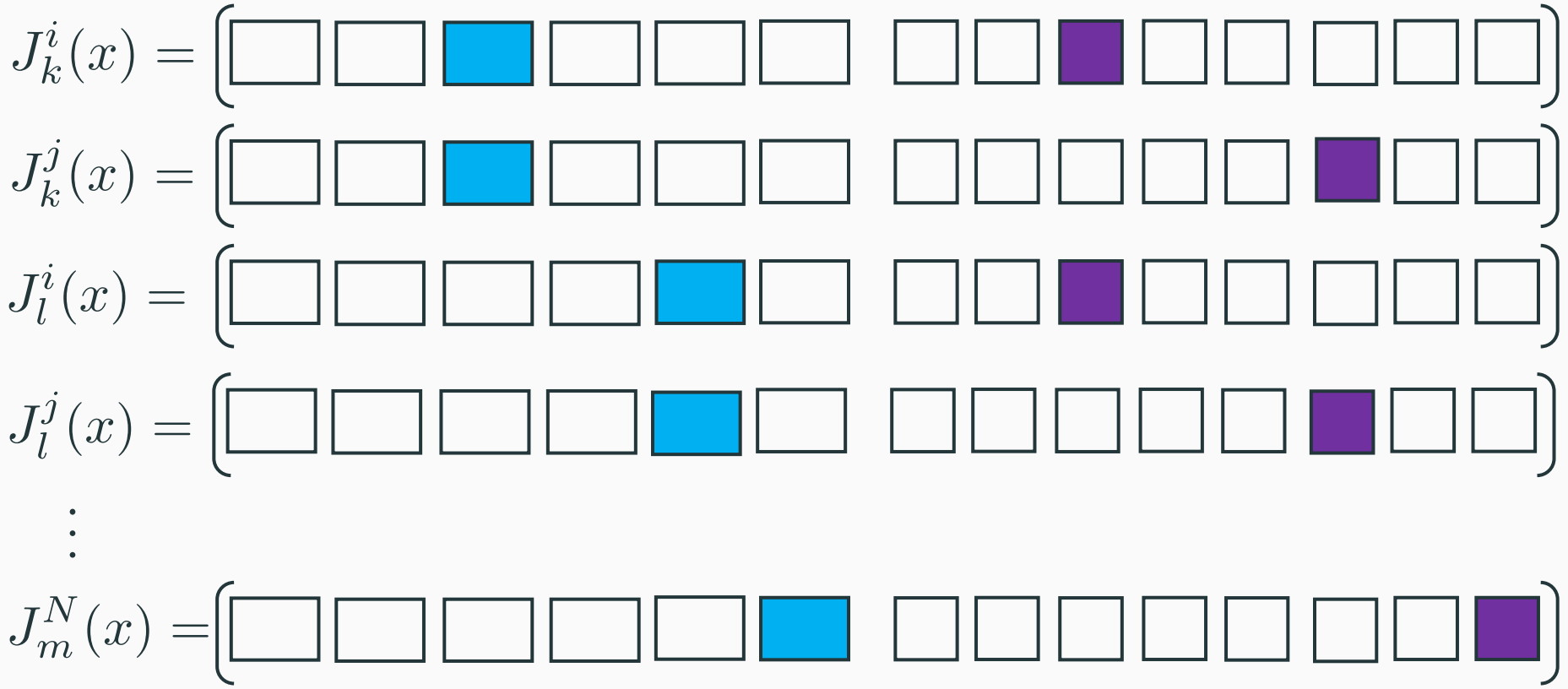
$$J_k^i(x) = \frac{\partial f_k^i(x)}{\partial x} = \frac{\partial \pi(X^i, C_k)}{\partial x} = \frac{\partial \pi(X^i, C_k)}{\partial [C_1, \ldots, C_m, \quad X^1, \ldots, X^N]^T}$$

$$= [0, \ldots, \frac{\partial \pi(X^i, C_k)}{\partial C_k}, \ldots, 0, \quad 0, \ldots, \frac{\partial \pi(X^i, C_k)}{\partial X^i}, \ldots, 0]$$

$J_k^i(x) =$ 

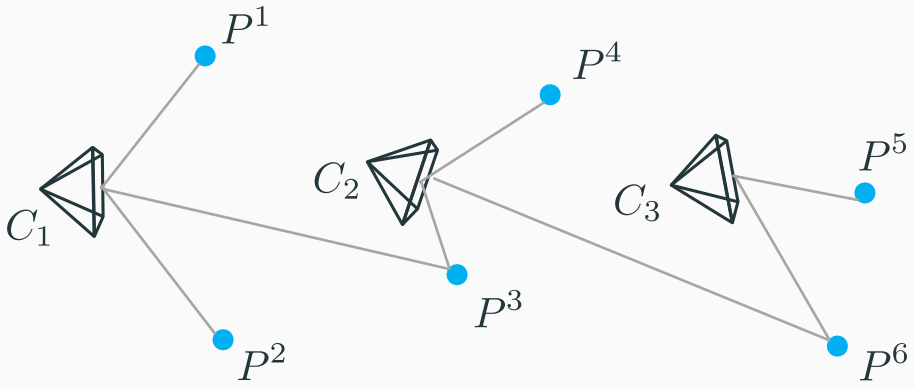
Jacobian sparsity in BA

We can note that the Jacobian exhibits a sparse structure

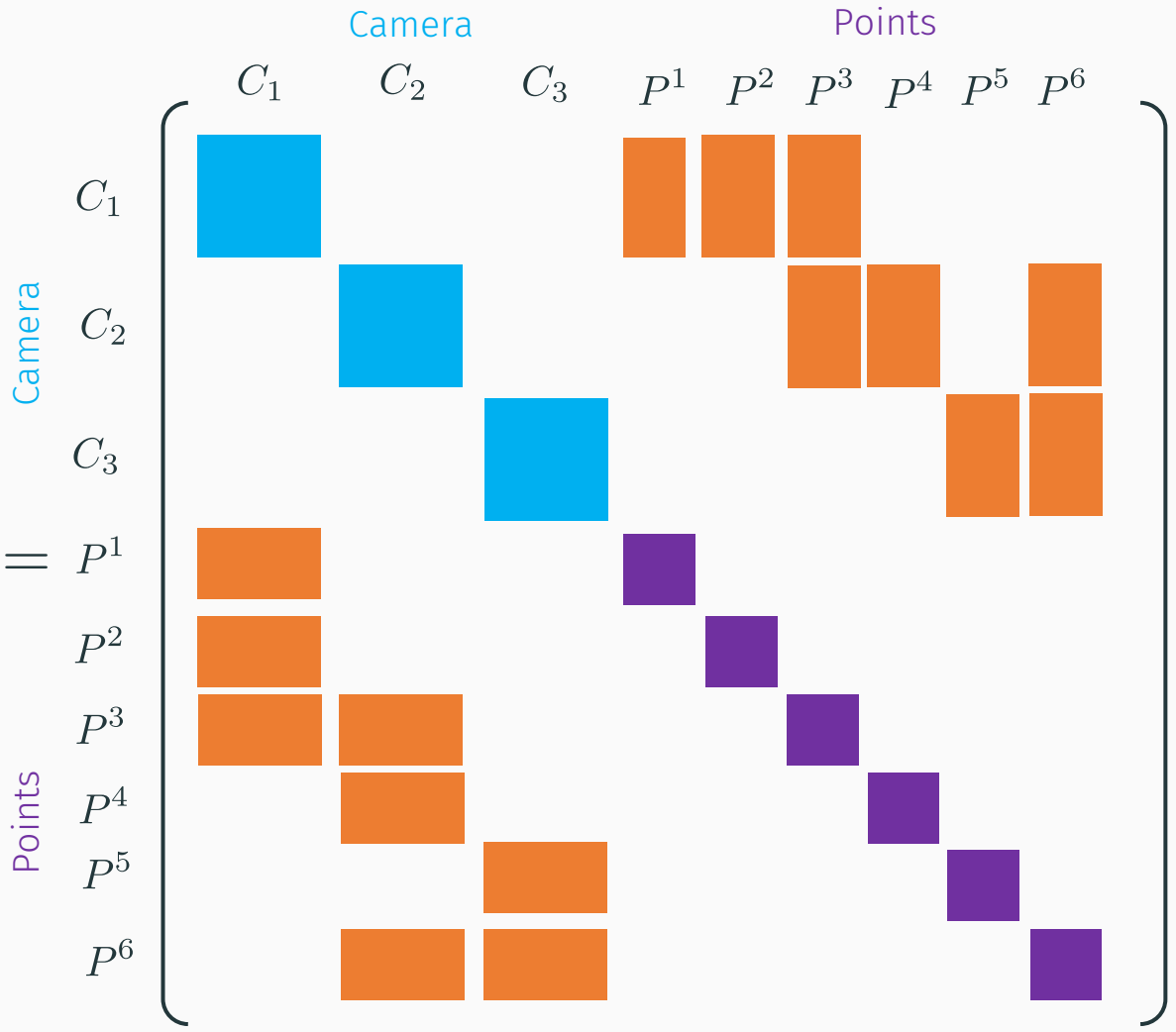


Sparsity in BA

The effect of sparsity is best noticed on the structure of matrix $J^T J$ which corresponds to the bundle depicted below.



$$J^T J =$$



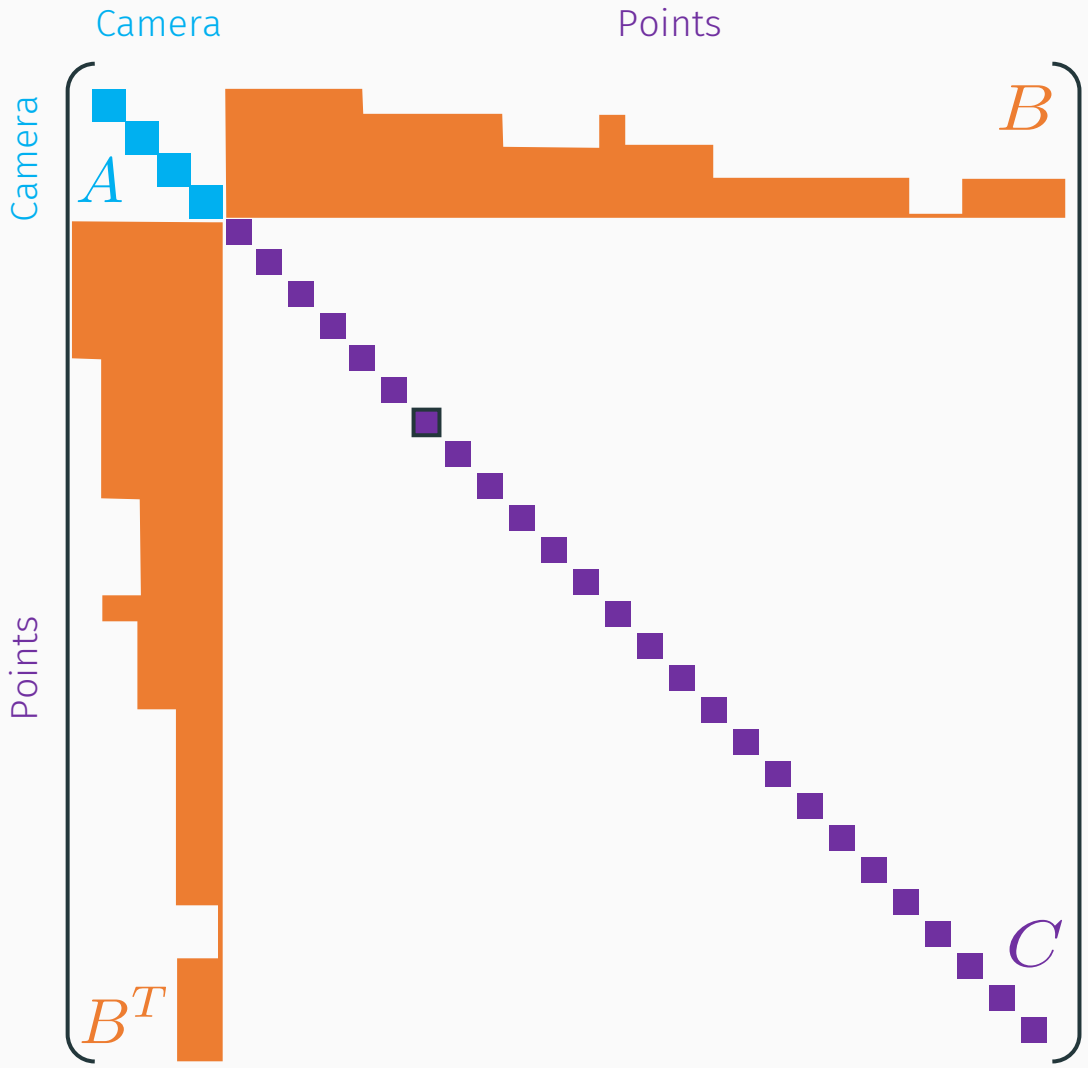
Sparsity in BA

Why do we care so much about the structure of $J^T J$? Because BA is solved using non-linear least squares that solve the following normal equation in each iteration

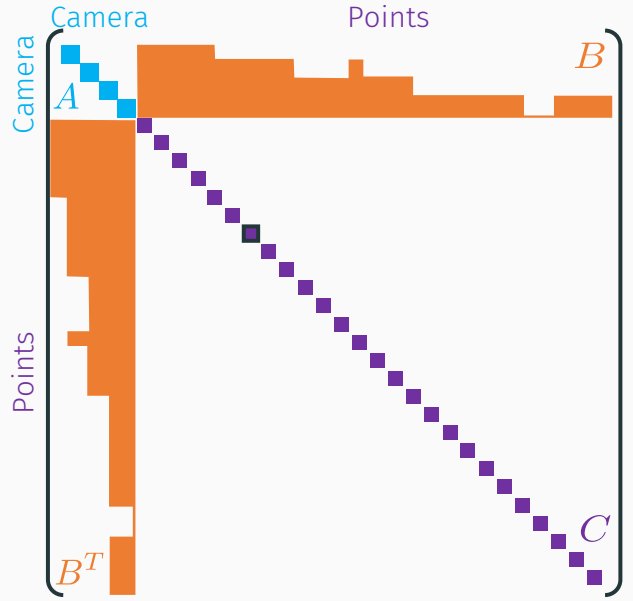
$$J^T J \Delta x = J^T b.$$

This involves inverting $J^T J$ and sparsity can be exploited to obtain computationally more efficient solutions.

Specifically, in real-world BA the number of features is much larger than the number of camera poses, and the matrix $J^T J$ has an arrow-like structure.



Sparsity in BA



The submatrices A and C have a block diagonal structure, where each block in C is a 3 x 3 matrix, while in A it is 6 x 6 (3 for translation and 3 for rotation).

The structure of the off-diagonal elements depends on the point observations of the camera poses.

We can restructure out normal equation as follows

$$J^T J \Delta x = J^T b.$$

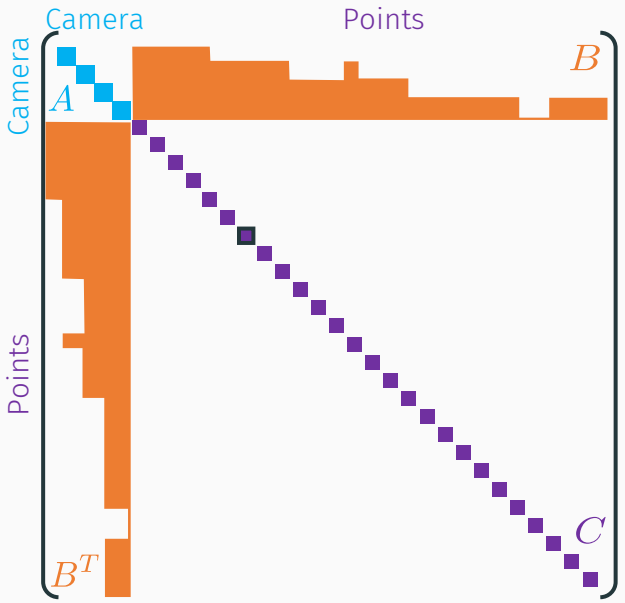
$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} b_c \\ b_p \end{bmatrix}$$

It will be computationally less complex to invert a block-diagonal matrix than a general dense matrix.

Schur elimination

The idea is to eliminate the non-diagonal part B of the restructured normal equation and this is called the Schur elimination

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} b_c \\ b_p \end{bmatrix}$$



$$\begin{bmatrix} I & -BC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} I & -BC^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} b_c \\ b_p \end{bmatrix}$$

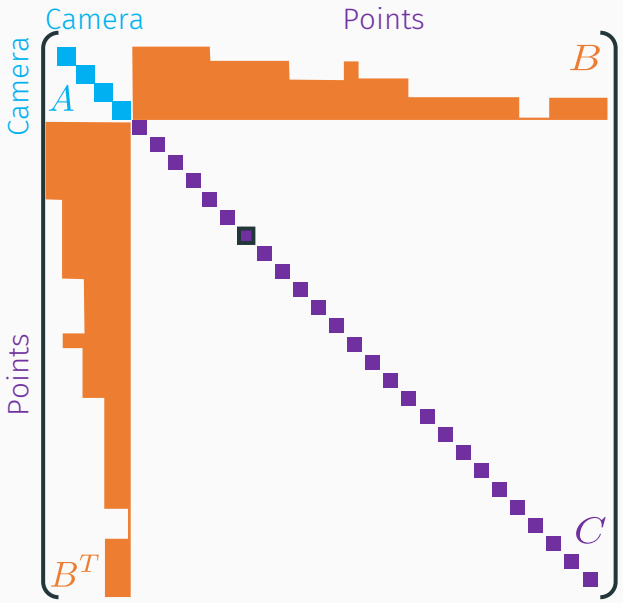
$$\begin{bmatrix} A - BC^{-1}B^T & 0 \\ B^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} b_c - BC^{-1}b_p \\ b_p \end{bmatrix}.$$

The upper row is independent of Δx_p and produces a reduced system just for the camera update that is of much lower dimension than the whole problem

$$[A - BC^{-1}B^T]\Delta x_c = b_c - BC^{-1}b_p.$$

Schur elimination

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta x_p \end{bmatrix} = \begin{bmatrix} b_c \\ b_p \end{bmatrix}$$



The, the map (structure) is solved by back substitution where we insert the solved Δx_c in the equation to obtain the points update

$$\Delta x_p = C^{-1}(b_p - B^T \Delta x_c).$$

Note that this is still computationally efficient since we only need to compute the block diagonal inverse C^{-1}.

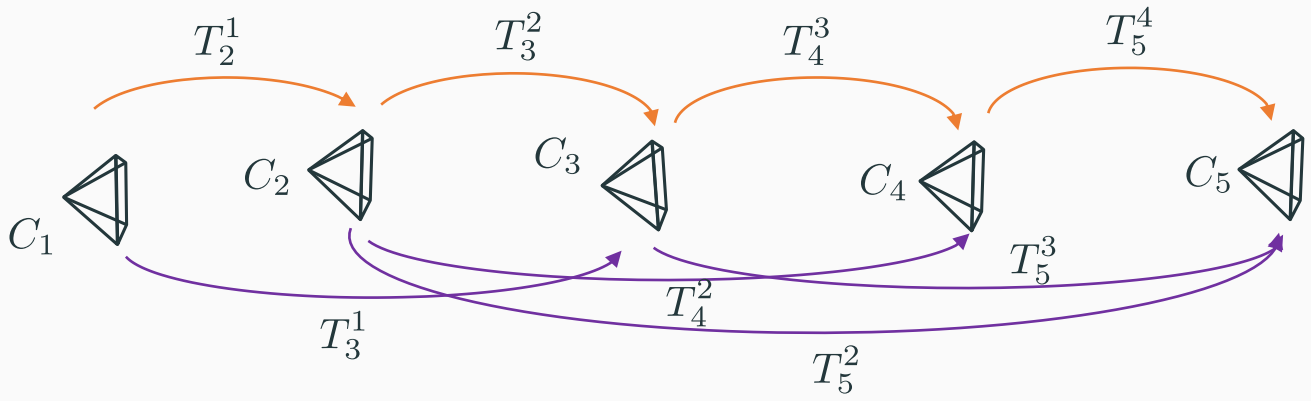
It is also interesting to mention that the matrix $[A - BC^{-1}B^T]$ is called the co-visibility matrix whose non-zero elements indicate that there is at least one common observation between the corresponding two camera poses.

Pose graph optimization

An alternative to BA is pose-graph optimization (PGO) that does not estimate the map (structure) but only optimizes over the camera poses $C_1 = (I, 0), \ldots, C_k = (R_k, t_k), \ldots, C_m = (R_m, t_m)$.

In odometry, this means that besides consecutive poses we also have additional constraints in the form of non-adjacent camera poses in the window of m frames

$$C^*_{1:m} = \arg\min_{C_{1:m}} \sum_i \sum_j ||C_i - C_j T^i_j||^2.$$



Pose graph optimization

Note that we are slightly *abusing* the notation in $||C_i - C_j T_j^i||^2$ since we need to measure distance between two matrices, while their difference is not what we are looking for.

Metrics on the space of rigid body transformations and optimization on the space thereof is an involved subject and part of our postgraduate studies.

Nevertheless, popular optimization frameworks often used in robotics can also solve PGO problems (GTSAM, iSAM2, Ceres, and g2o) with addition of SE-Sync that is specialized for PGO.

Outline

- Introduction
  - Reprojection error
- Non-linear optimization primer
- Local optimization
  - Bundle adjustment
  - Exploiting sparsity
  - Pose graph optimization
- Visual odometry state of the art
  - Feature-based and direct methods

Feature-based vs. direct odometry

The approaches that we have discussed thus far were based of feature detection and matching, in literature such methods are nowadays called feature-based or indirect methods.

Why indirect? Simply to put them in contrast to direct methods that instead of minimizing the reprojection error, aim to minimize the photometric error

$$(R_k^*, t_k^*, X^*) = \underset{R_k, t_k, X^{(1:N)}}{\arg\min} \sum_i ||I_{k-1}(p_{k-1}^i) - I_k(\pi(X^i, R_k, t_k))||^2.,$$

$$(R_k^*, t_k^*, X^*) = \underset{R_k, t_k, X^{(1:N)}}{\arg\min} \sum_i ||p_k^i - \pi(X^i, R_k, t_k)||^2.$$

Given that, direct methods work directly on pixel intensity values.

Feature-based vs. direct odometry

Compared to feature-based (indirect) methods that

1. detect and match features
2. run RANSAC to determine the inlier set
3. minimize the reprojection error,

Direct methods, on the other hand, have no feature detection and matching, thus no RANSAC is required, but only

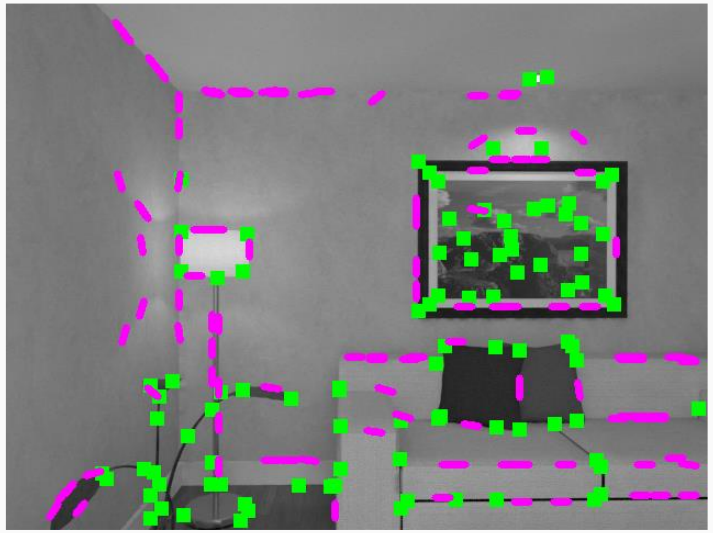
1. minimize the photometric error.

Thus, direct methods have the advantage of (1) lower computational complexity, (2) higher robustness to motion blur and weak texture; however, drawbacks are (1) limitation to smaller baselines and are (2) sensitivity to initialization (due to the high non-linearity of the optimization problem).

They are evidently also sensitive to illumination changes, but due to high framerate, this might not pose such a problem.

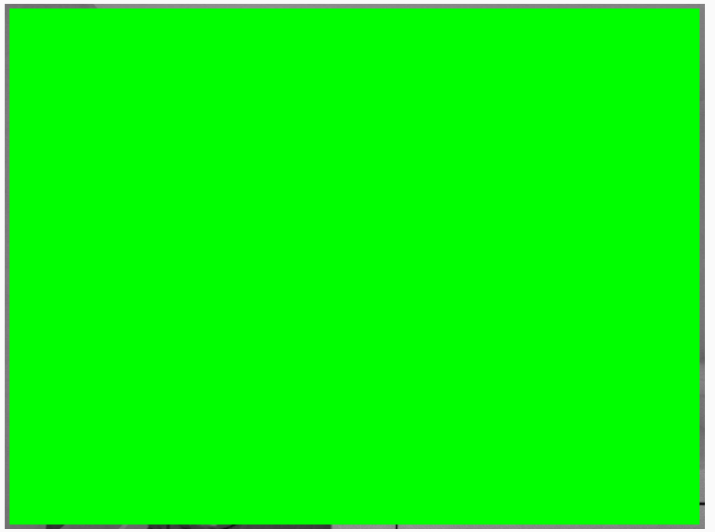
Direct methods: sparse, semi-dense, and dense

Direct methods are also divided based on the volume of used pixels; thus we have sparse, semi-dense, and dense methods.



Sparse

Semi-dense



Dense

Dense methods use all the pixels, semi-dense focus on edges (parts with a strong gradient), while sparse use corners and „edgelets”.

[1] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, D. Scaramuzza (2017.) „SVO: Semi-Direct Visual Odometry for Monocular and Multi-Camera Systems.”

ORB-SLAM

ORB-SLAM[2] is a feature-based odometry and SLAM system:

1. Based on detecting FAST corners and matching them using ORB descriptors
2. Minimizes the reprojection error (3D-2D)
3. Uses local bundle adjustement over a sliding window of camera frames
4. Can relocalize (once lost find the location within the built map
5. Includes asynchronous global optimization

Available in open source and extended to ORB-SLAM3 leverage IMU sensors and wide-angle lenses.

[2] R. Mur-Artal, J. D. Tardos, J. M. M. Montiel (2015.) „ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras."

SOFT2

SOFT2[3] is a croatian-made visual odometry system:

1. Based on detecting and matching blob patterns via normalized cross-correlation
2. Runs multihypothesis patch perspective correction for matching
3. Minimizes point-to-epipolar line distances (2D-2D)
4. Online extrinsic camera calibration
5. Runs local epipolar-line bundle adjustement over a window of frames
6. Pure odometry (no loop closing and global optimization)
7. For road vehicles only

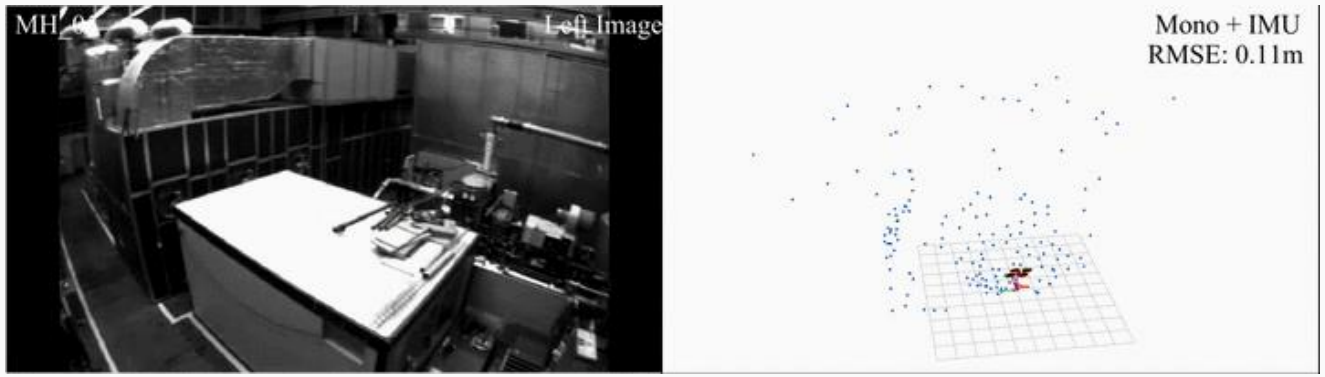
At the moment of writing the highest ranking visual odometry on the KITTI and KITTI-360 datasets.

[3] I. Cvišić, I. Marković, I. Petrović (2022.) „SOFT2: Stereo Visual Odometry for Road Vehicles based on a Point-to-Epipolar-Line Metric."

VINS-Fusion

VINS-Fusion[4] is a feature-based SLAM system:

1. Optical-flow based feature detection and tracking
2. Fuses inertial measurements in a 3D-2D motion estimation pipeline
3. Includes IMU bias correction and online extrinsic calibration
4. Runs local bundle adjustement over a sliding window of camera frames
5. Contains loop closing, relocalization and global pose graph optimization over keyframes
6. Contains rolling shutter support

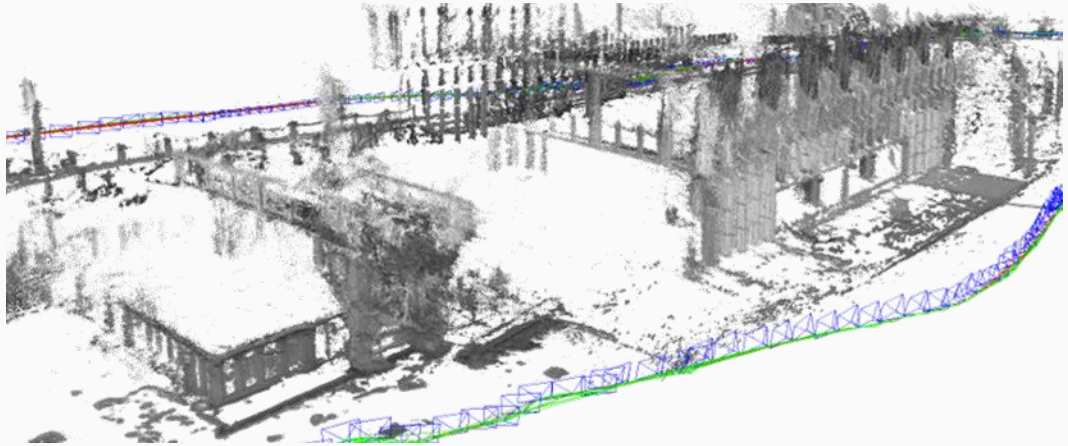


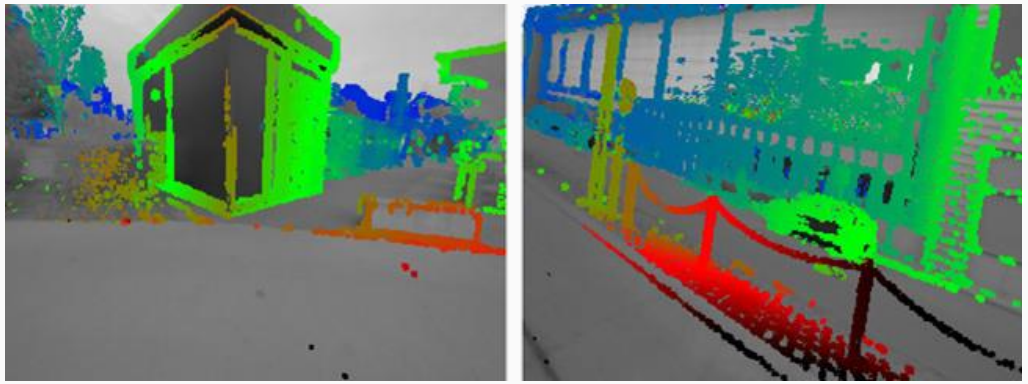
[4] T. Qin, S. Cao, J. Pan, P. Li, S. Shen (2019.) „VINS-Fusion: An optimization-based multi-sensor state estimator."

LSD-SLAM

LSD-SLAM[5] is a direct semi-dense odometry and SLAM system:

1. Minimizes photometric error
2. Builds a semi-dense map of edges
3. Uses local bundle adjustement over a sliding window of camera frames
4. Contains loop closing, relocalization and global optimization (asynchrounous)



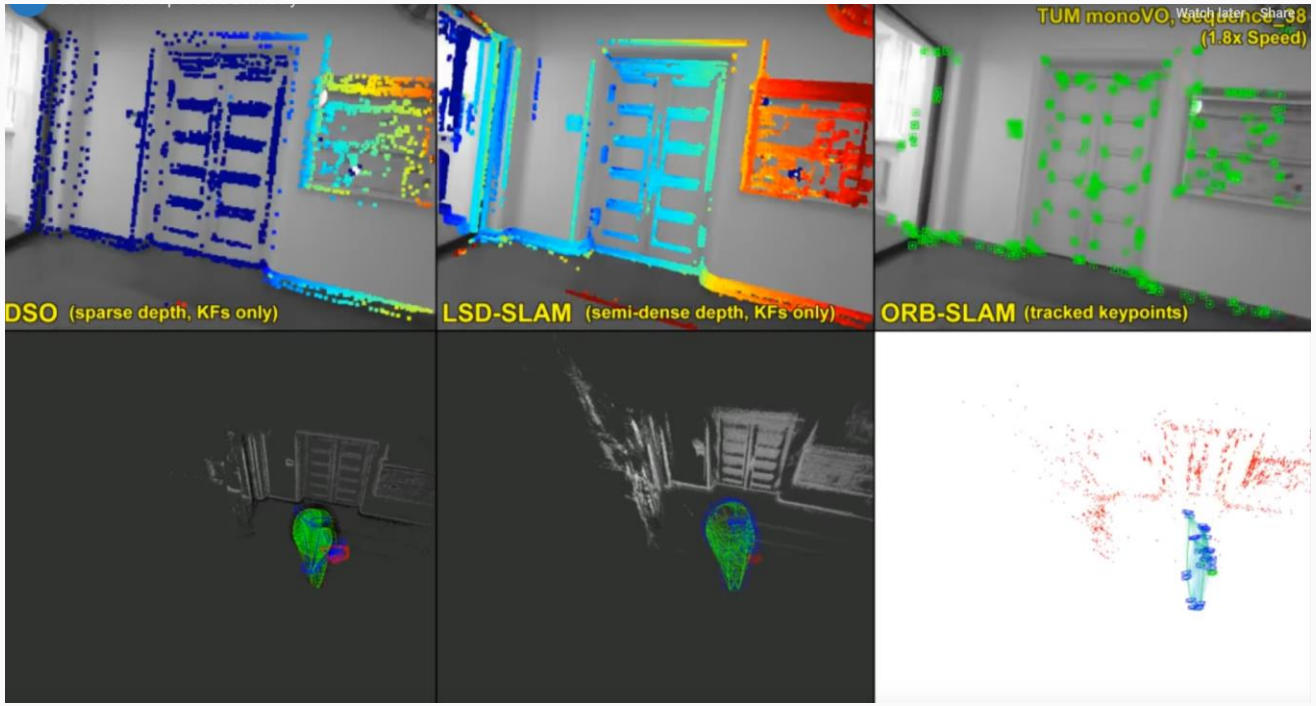


[5] J. Engel, T. Schöps, D. Cremers (2014.) „LSD-SLAM: Large-Scale Direct Monocular SLAM."

DSO

DSO[6] is a direct sparse odometry system:

1. Minimizes photometric error
2. Builds a sparse map of points with strong gradients
3. Uses local bundle adjustement over a sliding window of camera frames
4. Runs global optimization (asynchrounous)

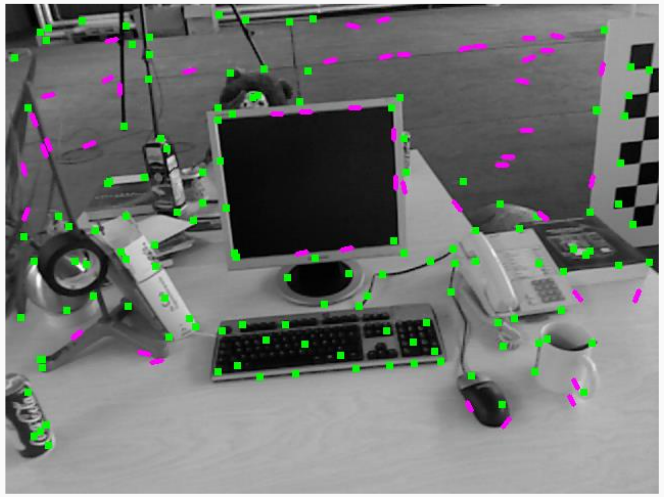


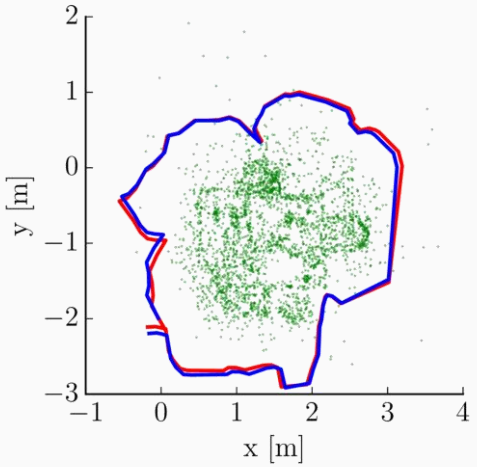
[6] J. Engel, V. Koltun and D. Cremers (2018.) „DSO: Direct Sparse Odometry."

SVO

SVO[7] is a blend of direct and feature-based approaches:

1. Minimizes photometric error for frame-to-frame motion estimation
2. Minimizes reprojection error for pose refinement to keyframes
3. Maps corners and edgelets
4. Contains loop closing, relocalization and global optimization (asynchrounous)
5. Can run up to 400 fps on laptop computers



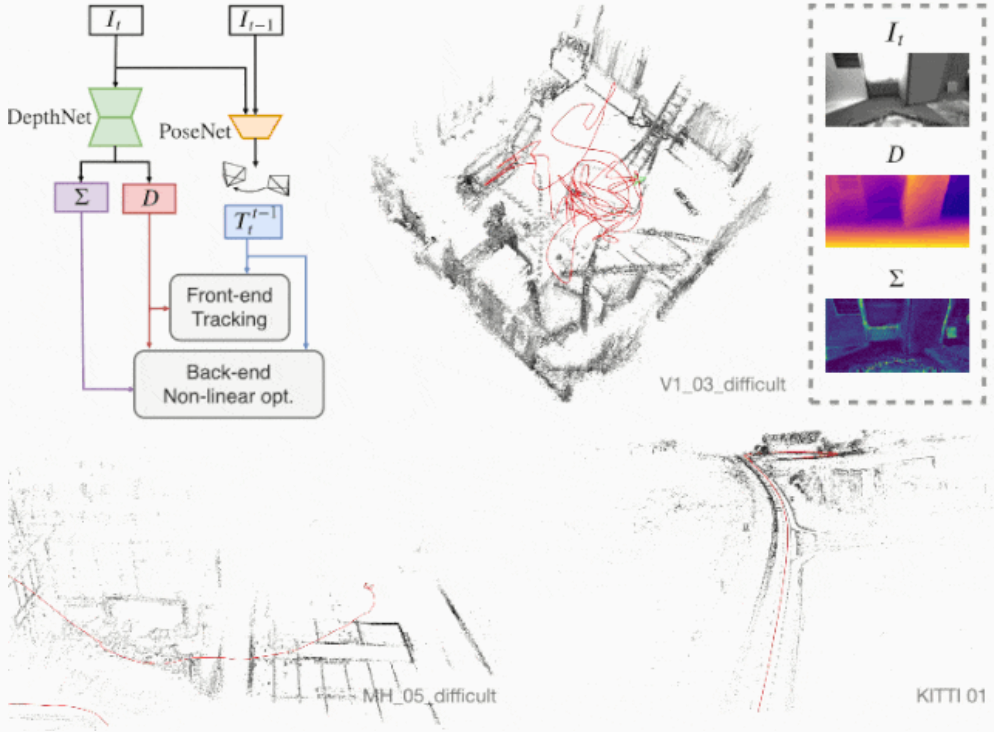


[7] C. Forster, M. Pizzoli and D. Scaramuzza (2014.) „SVO: Fast Semi-Direct Monocular Visual Odometry."

D3VO

D3VO[8] is a direct sparse odometry with deep depth estimation:

1. Minimizes photometric error
2. Builds a sparse map of points with strong gradients
3. Includes self-supervised monocular deep depth estimation (couples DepthNet and PoseNet, minimizes photometric error)
4. Models photometric uncertainties and illumination changes
5. Uses local bundle adjustment over a sliding window of camera frames



[8] C. Forster, M. Pizzoli and D. Scaramuzza (2014.) „SVO: Fast Semi-Direct Monocular Visual Odometry."

Questions?