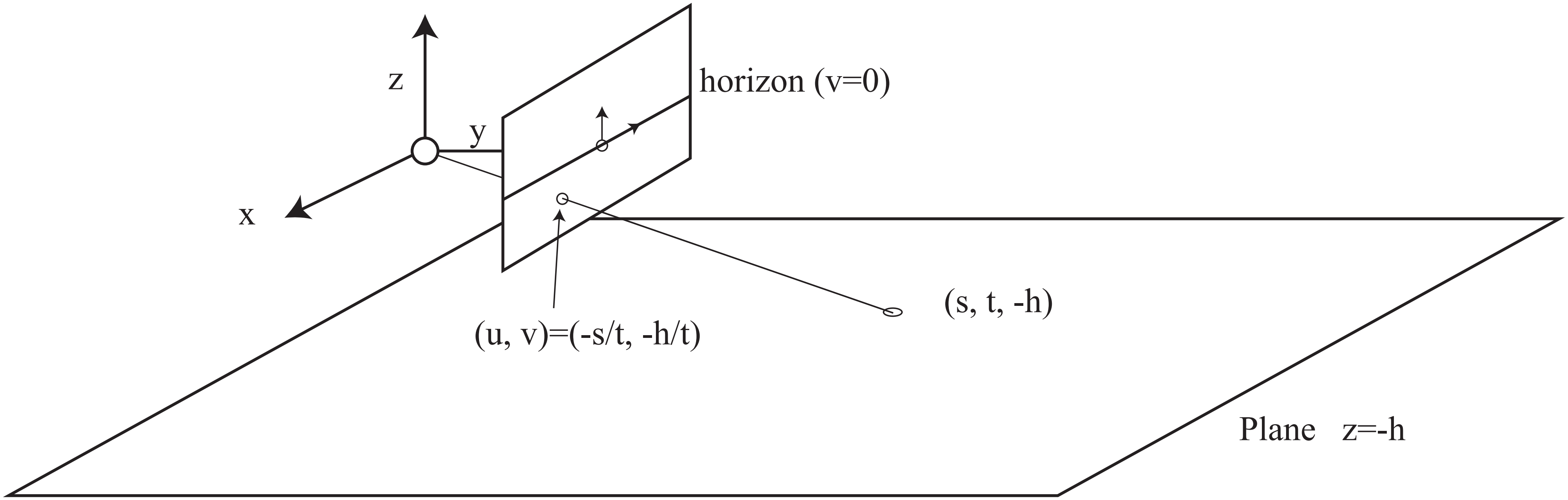
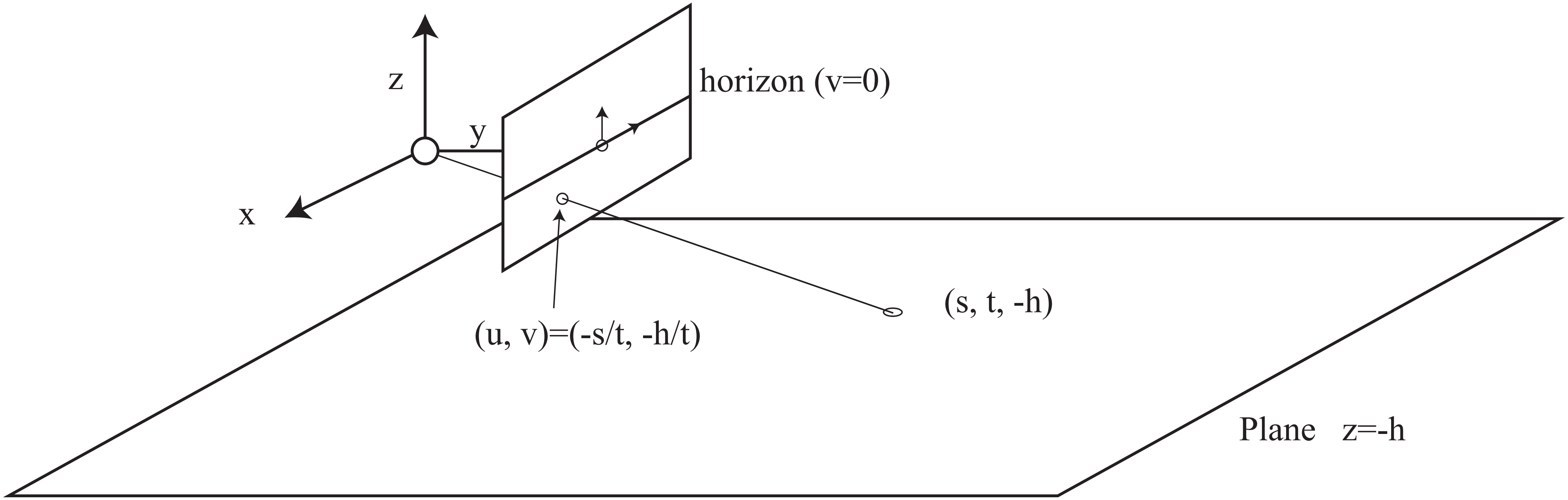
Semi-direct Methods for VO and SLAM

D.A. Forsyth, UIUC

PIPH

- Perpendicular image plane, fixed height
- Imagine a camera at a fixed height
  - moving rigidly over a textured ground plane
  - bottom half of image is distorted ground plane texture





$$\overset{\mathcal{C}}{\downarrow}$$

$$\begin{pmatrix} U/W \\ V/W \\ 1 \end{pmatrix} \equiv \begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} s \\ t \\ -h \end{pmatrix}$$

Image coordinates

Plane texture coords

State

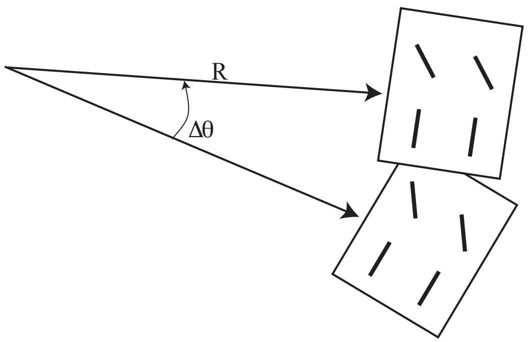
Position and orientation of the robot

$$\mathbf{x} = \begin{bmatrix} \mathcal{R} \\ \mathcal{M} \end{bmatrix} = \begin{bmatrix} \mathcal{R} \\ \mathcal{L}_1 \\ \dots \\ \mathcal{L}_n \end{bmatrix}$$

Landmark 1 position in OCF

All landmark positions in original coordinate frame

A general movement model



$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \rightarrow \begin{bmatrix} x + R(\sin(\theta + \Delta\theta) - \sin\theta) \\ y - R(\cos(\theta + \Delta\theta) - \cos\theta) \\ \theta + \Delta\theta \end{bmatrix}$$

THIS ISN'T LINEAR!

v_t = velocity
omega_t = rotational velocity

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{v_t}{\omega_t}(\sin(\theta + \omega_t \Delta t) - \sin(\theta)) \\ -\frac{v_t}{\omega_t}(\cos(\theta + \omega_t \Delta t) - \cos(\theta)) \\ \omega_t \Delta t \end{bmatrix}$$

State update

$$\mathbf{x}_i = f(\mathbf{x}_{i-1}, \mathbf{n})$$

$$\mathbf{x} = \begin{bmatrix} \mathcal{R} \\ \mathcal{M} \end{bmatrix} = \begin{bmatrix} \mathcal{R} \\ \mathcal{L}_1 \\ \dots \\ \mathcal{L}_n \end{bmatrix}$$

- The vehicle moves, as above;
  - but the landmarks don't move
  - and there isn't any noise

$$\begin{bmatrix} \mathcal{R} \\ \mathcal{L}_1 \\ \mathcal{L}_2 \\ \dots \\ \mathcal{L}_n \end{bmatrix} \rightarrow \begin{bmatrix} h(\mathcal{R}) + \xi \\ \mathcal{L}_1 \\ \mathcal{L}_2 \\ \dots \\ \mathcal{L}_n \end{bmatrix}$$

Measuring position

- Landmark is at:
  - in world coordinate system
- We record position in vehicle's frame:

$$\begin{bmatrix} u \\ v \\ -h \end{bmatrix}$$

We don't actually observe this

$$\begin{bmatrix} x_u \\ x_v \\ -h \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{-\theta} & 0 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} (u-x) \\ (v-y) \\ -h \end{bmatrix}$$

vehicle orientation in world coords

point posn in world coords

point posn in vehicle coords

vehicle posn in world coords

THIS ISN'T LINEAR!

What we observe

$$\begin{bmatrix} c_u \\ c_v \end{bmatrix} = \begin{bmatrix} -x_u/x_v \\ -h/x_v \end{bmatrix}$$

- Options:
  - linearize that
  - OR

$$\begin{bmatrix} x_u \\ x_v \end{bmatrix} = \begin{bmatrix} hc_u/c_v \\ -h/c_v \end{bmatrix}$$

The steps, EKF:

Have:

$$\overline{X}^{+}_{i-1} \qquad \Sigma^{+}_{i-1}$$

Construct:

$$\bar{X}^{-}_{i} = f_i(\bar{\mathbf{x}}^{+}_{i-1}, \mathbf{0}) \qquad \Sigma^{-}_{i} = \mathcal{F}_x \Sigma^{+}_{i-1} \mathcal{F}^{T}_{x} + \mathcal{F}_n \Sigma_{n,i} \mathcal{F}^{T}_{n}$$

Measurement arrives: $\mathbf{y}_i = g_i(\mathbf{x}_i, \mathbf{n})$

Now construct:

$$\bar{X}^{+}_{i} = \bar{X}^{-}_{i} + \mathcal{K}_i \left[\mathbf{y}_i - g_i(\bar{X}^{-}_{i}, \mathbf{0})\right] \qquad \Sigma^{+}_{i} = \left[Id - \mathcal{K}_i \mathcal{G}_x\right] \Sigma^{-}_{i}$$

Where:

$$\mathcal{K}_i = \Sigma^{-}_{i} \mathcal{G}^{T}_{x} \left[\mathcal{G}_x \Sigma^{-}_{i} \mathcal{G}^{T}_{x} + \mathcal{G}_n \Sigma_{m,i} \mathcal{G}^{T}_{n}\right]^{-1}$$

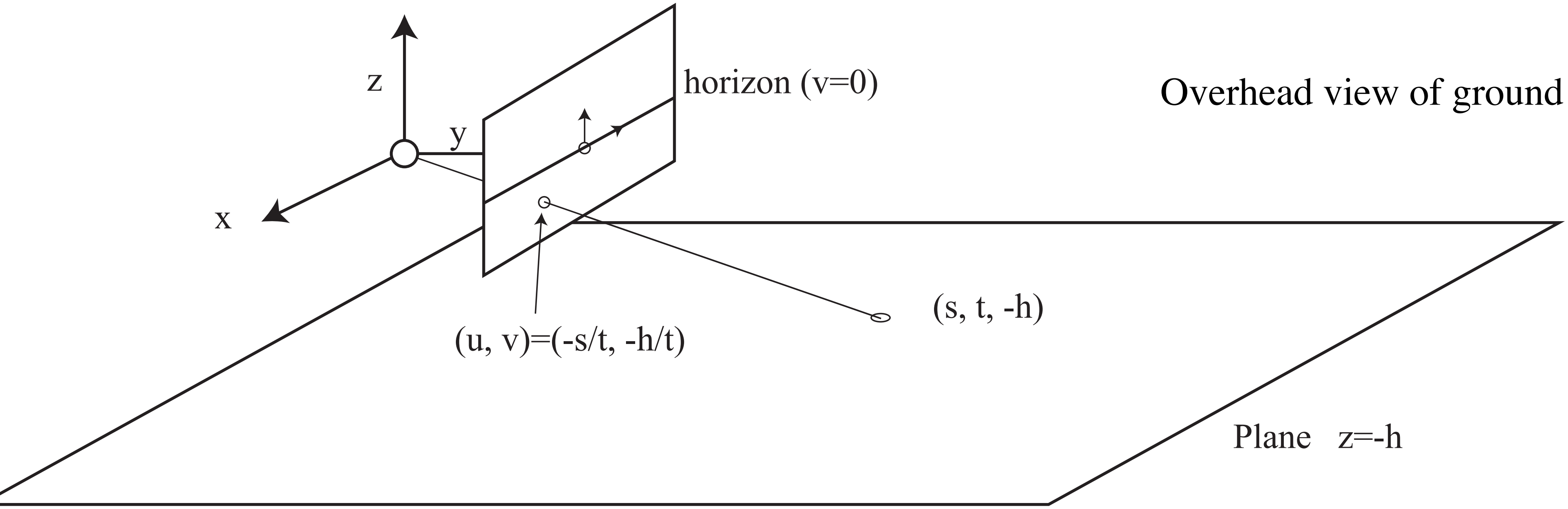
Alternative approach

- We will do odometry first

- Assume we *know* the camera height and intrinsics
  - We can *reconstruct* the ground plane pattern
    - apply the inverse of the given map
      - actually, don’t need intrinsics (later; fairly mathy)

  - Now if camera translates, ground plane translates
    - if camera rotates, ground plane rotates

Camera image

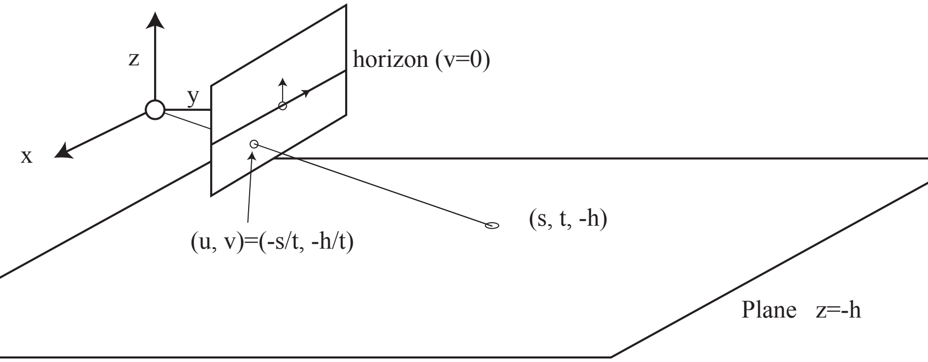


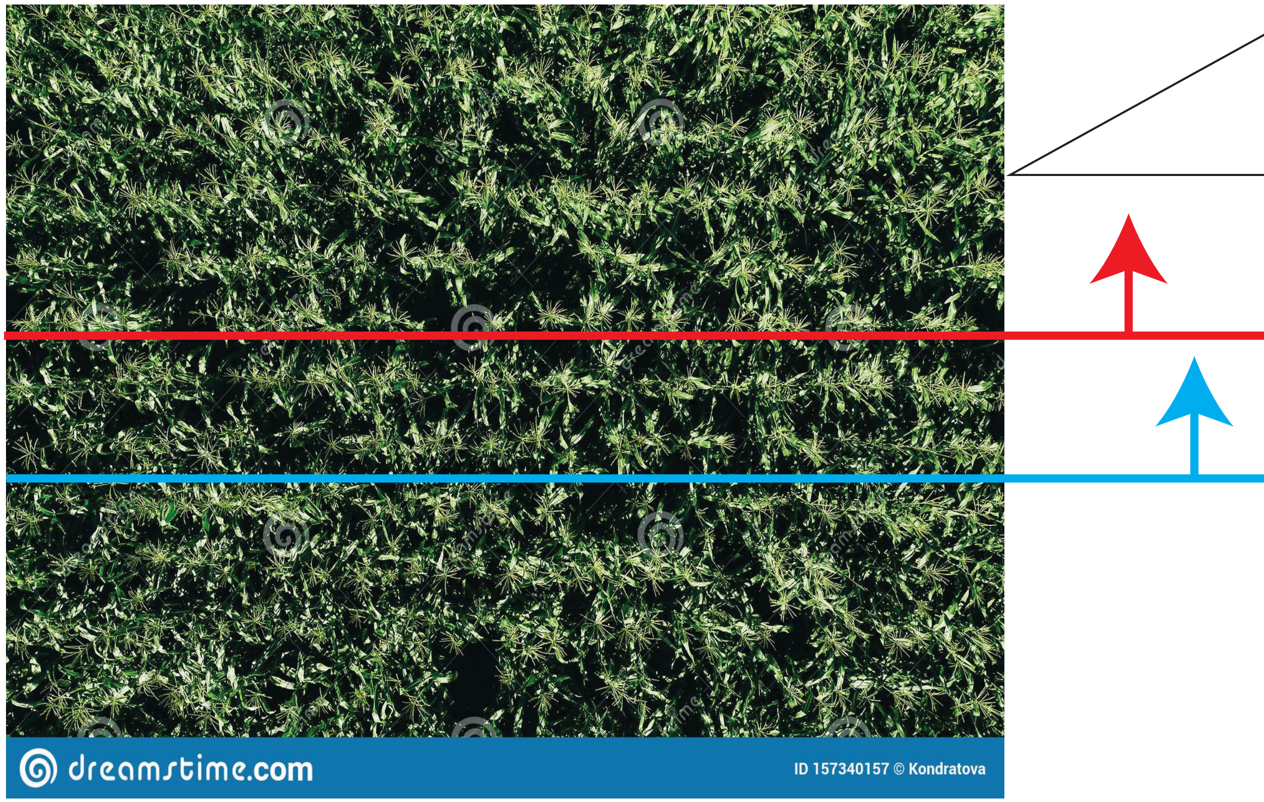
Overhead view of ground plane

Camera image

Overhead view of ground plane

Ground plane for translating camera

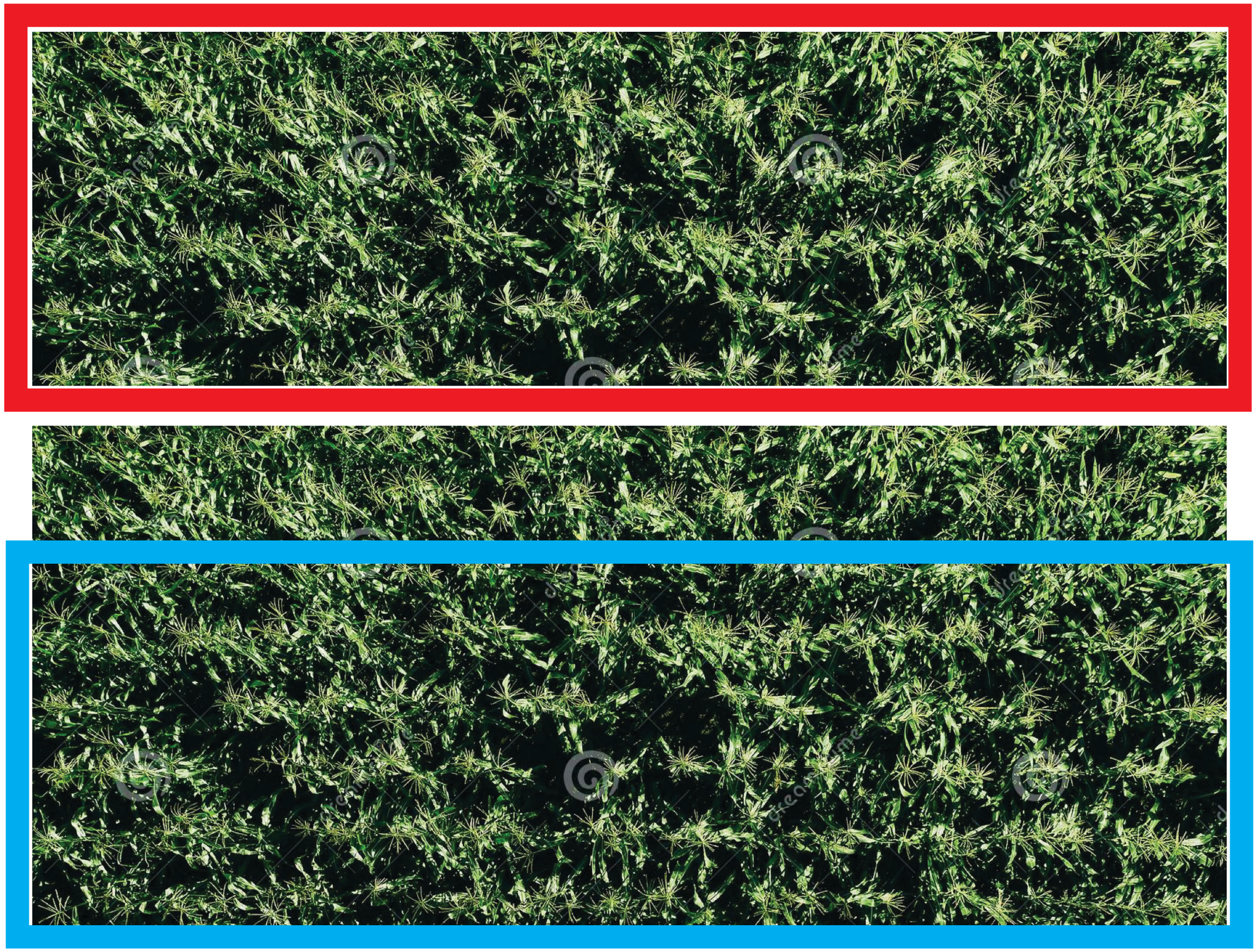




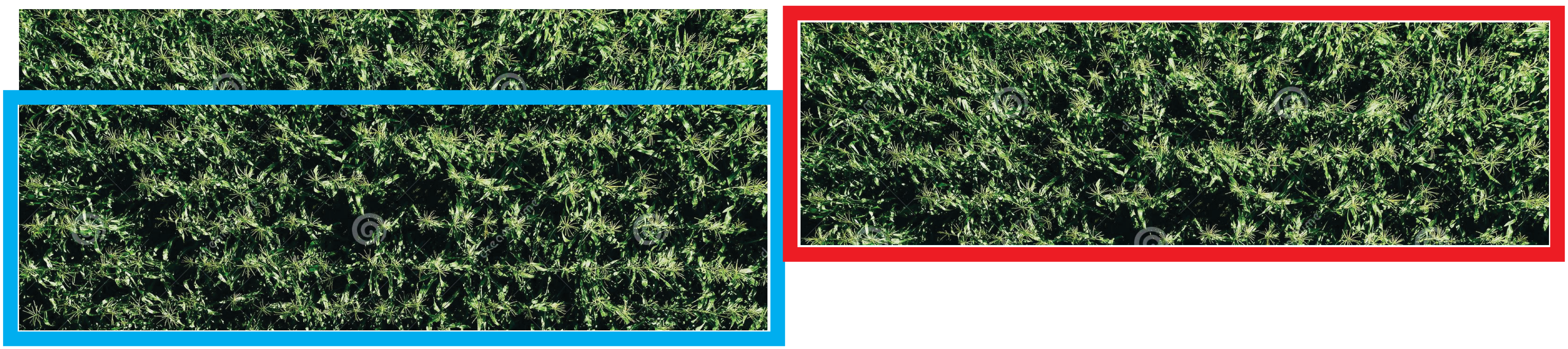
Red camera (i+1) sees forward of this

Blue camera (i) sees forward of this

The two ground planes…



can be aligned and this reveals translation



How to align?

- Alternatives
  - find interest points; use registration methods, above
  - direct method (now)
  - semi-direct method (shortly)

A crude direct method

Image obtained at current posn, orientation
projected down to ground plane

$$\min \sum_{s,t} \| G(s,t) - I^{\downarrow}((s-x)\cos\theta + (t-y)\sin\theta, -(s-x)\sin\theta + (t-y)\cos\theta) \|^2$$

position of camera in ground plane

Ground plane model in
world coordinates

rotation in ground plane

This class of cost function is known as a photometric consistency constraint
KEY IDEA - if you what you see back using your new pose,
you should see the ground plane
KEY IDEA - all pixels contribute to estimate of pose,
so could be very accurate

It’s crude…

- Minimize how?
  - not easy in that form
- As translation gets big, you’ll lose precision
  - guaranteed by perspective effects

Steps to fix

- Patches rather than all pixels
  - centered on interest points
  - semi-direct method
- Be willing to drop patches
  - to control perspective problems
- Write in terms of warps
  - to manage notation
- Equivalent to Lucas-Kanade
  - see L-K, Baker

In terms of warps

Image in current frame

Parameter update

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x}) \right]^2$$

Pixels on patch

Parametric warp

Patch in original ground plane

Parameters to i'th frame

$$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}.$$

ish

- Minimize how?
- Pose by accumulating updates
  - advantage
    - efficient
  - BUT
    - accumulated error (drift - which we'll fix)

Minimize How? Gauss Newton

$$\min_{\mathbf{p}} \sum_k f_k^2(\mathbf{p})$$

At minimum, we have:

$$\sum_k f_k(\mathbf{p}) \left(\nabla_{\mathbf{p}} f_k\right) = 0$$

Imagine we are at p_i close to minimum; extract step from:

$$\sum_k f_k(\mathbf{p}_i + \delta\mathbf{p}) \left(\nabla_{\mathbf{p}} f_k\right)\big|_{\mathbf{p}+\delta\mathbf{p}} = 0$$

Minimize How? Gauss Newton

$$\sum_k f_k(\mathbf{p}_i + \delta\mathbf{p}) \left(\nabla_{\mathbf{p}} f_k\right)|_{\mathbf{p}+\delta\mathbf{p}} = 0$$

By truncating a Taylor series, this is approximately:

$$\sum_k \left(f_k(\mathbf{p}_i) + \nabla_{\mathbf{p}} f_k|_{\mathbf{p}}^T \delta\mathbf{p})\right) \left(\nabla_{\mathbf{p}} f_k|_{\mathbf{p}} + \mathcal{H}\delta\mathbf{p}\right) = 0$$

Hessian - matrix of second partials
Ignore this, in the hope it is small
and because it is inconvenient,
expand to get

$$\sum_k \left(f_k(\mathbf{p}_i)\nabla_{\mathbf{p}} f_k\right) = -\sum_k \left(\nabla_{\mathbf{p}} f_k\right)\left(\nabla_{\mathbf{p}} f_k\right)^T \delta\mathbf{p}$$

Notice

$$\sum_k \left(f_k(\mathbf{p}_i)\nabla_{\mathbf{p}} f_k\right) = -\sum_k \left(\nabla_{\mathbf{p}} f_k\right)\left(\nabla_{\mathbf{p}} f_k\right)^T \delta\mathbf{p}$$

Approximate Hessian:
Issue: need to recompute
at every step, which is slow

Inverse compositional method

Image in current frame

Patch in original ground plane

Parameter update

$$\sum_{\mathbf{x}} \left[T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p})) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2$$

Pixels on patch

Parametric warp

Parameters to i’th frame

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$$

Why? cheap updates

Taylor series

$$\sum_{\mathbf{x}} \left[T(\mathbf{W}(\mathbf{x};\mathbf{0})) + \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta \mathbf{p} - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]^2 .$$

Yields:

$$\Delta \mathbf{p} \;=\; H^{-1} \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x}) \right]$$

Where:

$$H \;=\; \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$$

But this is evaluated at 0 warp, and doesn't change from step to step

We now have odometry

- IF height is known, camera is calibrated
- Issues
  - managing perspective issues
    - drop patches
    - create patches
  - managing compute
    - interleave two kinds of registration
      - interest points (v. quick, quite accurate)
      - patches (slower, more accurate)
  - map
    - put all patches in original coordinate system
  - loop closure
    - camera updates may drift

Working in 3D

- Map
  - Point measurements
    - do we have depth?
      - If so, EKFSlam, above, will work easily
      - if we don't, it will still work (initialize depths carefully)
  - Patches and photometric consistency
    - must account for depth in matching
    - must control computation

SLAM with depth measurements

- RGB-D or stereo (sketch)
  - align depth map in view 2 with that of view 1
    - using rotation, translation, m-estimator+IRLS
      - wrinkle - use intensity as well as depth
  - keep
    - aligned depths (prune redundancies) for mapping
    - transformation (for localization)
- Key question
  - How do we manage computational cost?

SLAM with depth measurements - II

- Strategies:
  - Find image/depth interest points and register those
    - advantage: we know how to do this, straightforward, fast
    - possible disadvantage: ignoring a lot of information
  - Direct method: minimize photometric/depth alignment cost at every point
    - advantage: we know how to do this, all points contribute, accurate
    - disadvantage: much more expensive

Semi-direct methods

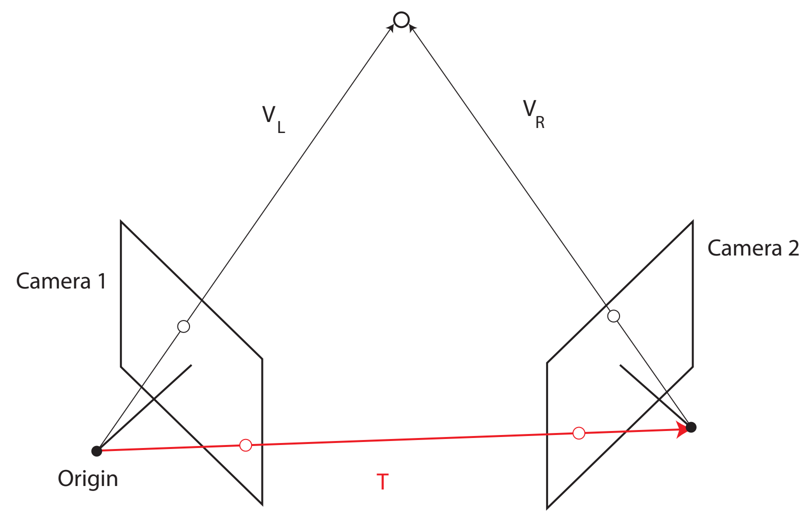
- Problem:
  - a direct method means you must touch many image/depth measurements
  - accurate, but likely slow
- Idea:
  - find interest points
  - use interest point AND its neighborhood
    - computing photometric consistency for neighborhood

Semi-direct visual odometry and mapping

- Monocular cameras
  - semi-direct
    - for the moment, consider interest points in the image
      - worry about neighborhoods in a moment
  - to predict image positions, we need depths
    - associate a depth with each interest point, and a depth estimate
      - for that point in each frame
      - stick a filter on this

SVO (semi-direct visual odometry)

- Must estimate camera motion from interest points
  - assume we have a depth associated with i'th interest point
    - update when camera moves
  - start
    - depth from prior OR
    - stereo in first two frames OR
    - depth from single image estimate



SVO (semi-direct visual odometry)

- Steps:
  - estimate camera motion from correspondences using photometric error
    - assume constant depth at each patch
  - now move patches in 2D to improve photometric error
  - now adjust 3D configuration of points and camera motion

Estimate camera motion from correspondences using photometric error

Recall: we know how to make these patches

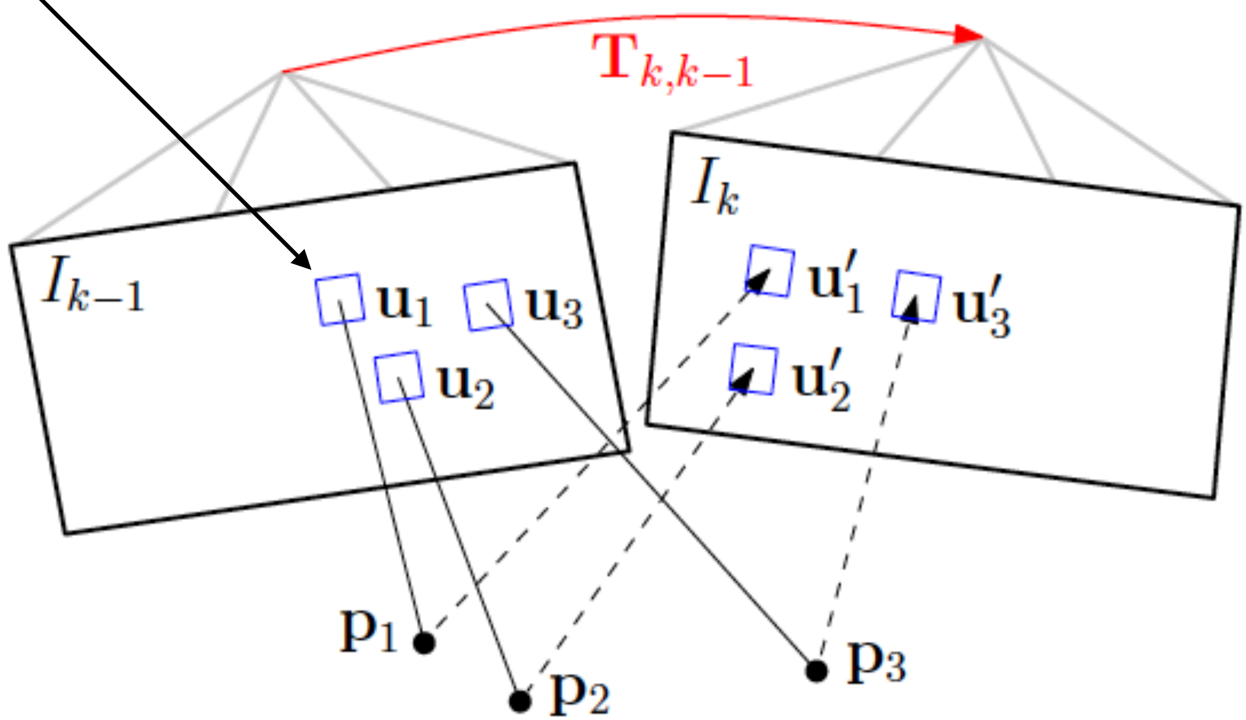


Fig. 2: Changing the relative pose $\mathbf{T}_{k,k-1}$ between the current and the previous frame implicitly moves the position of the reprojected points in the new image $\mathbf{u}'_i$. Sparse image alignment seeks to find $\mathbf{T}_{k,k-1}$ that minimizes the photometric difference between image patches corresponding to the same 3D point (blue squares). Note, in all figures, the parameters to optimize are drawn in red and the optimization cost is highlighted in blue.

now move patches in 2D to improve photometric error

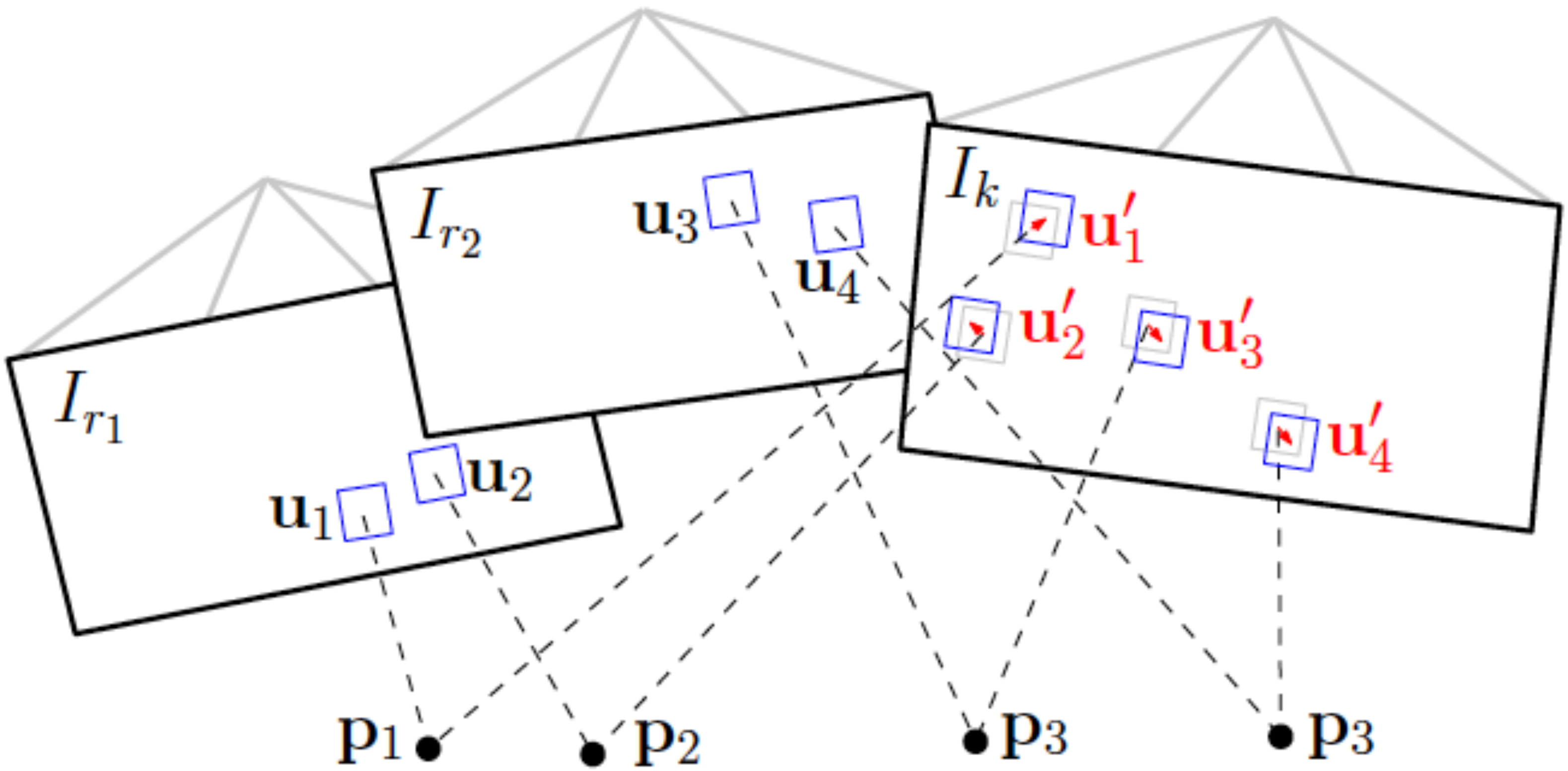


Fig. 3: Due to inaccuracies in the 3D point and camera pose estimation, the photometric error between corresponding patches (blue squares) in the current frame and previous keyframes r_i can further be minimised by optimising the 2D position of each patch individually.

now adjust 3D configuration of points and camera motion

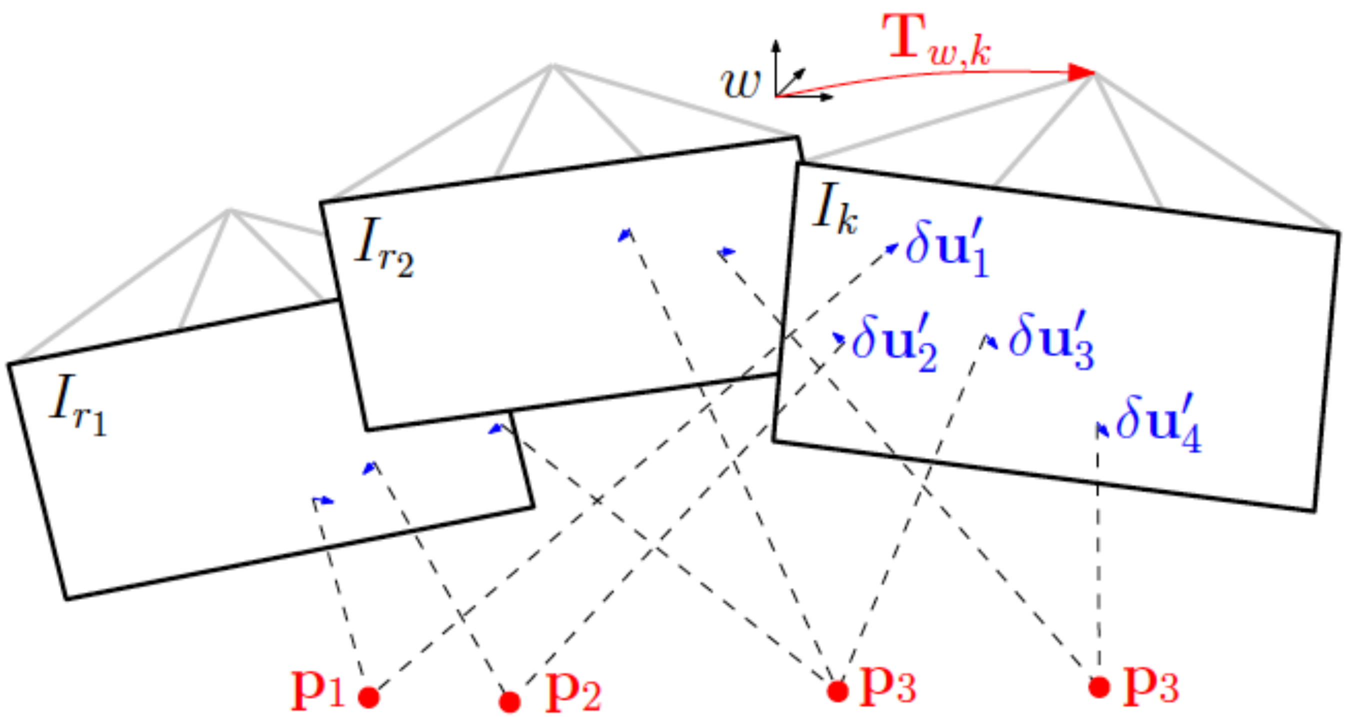


Fig. 4: In the last motion estimation step, the camera pose and the structure (3D points) are optimized to minimize the reprojection error that has been established during the previous feature-alignment step.

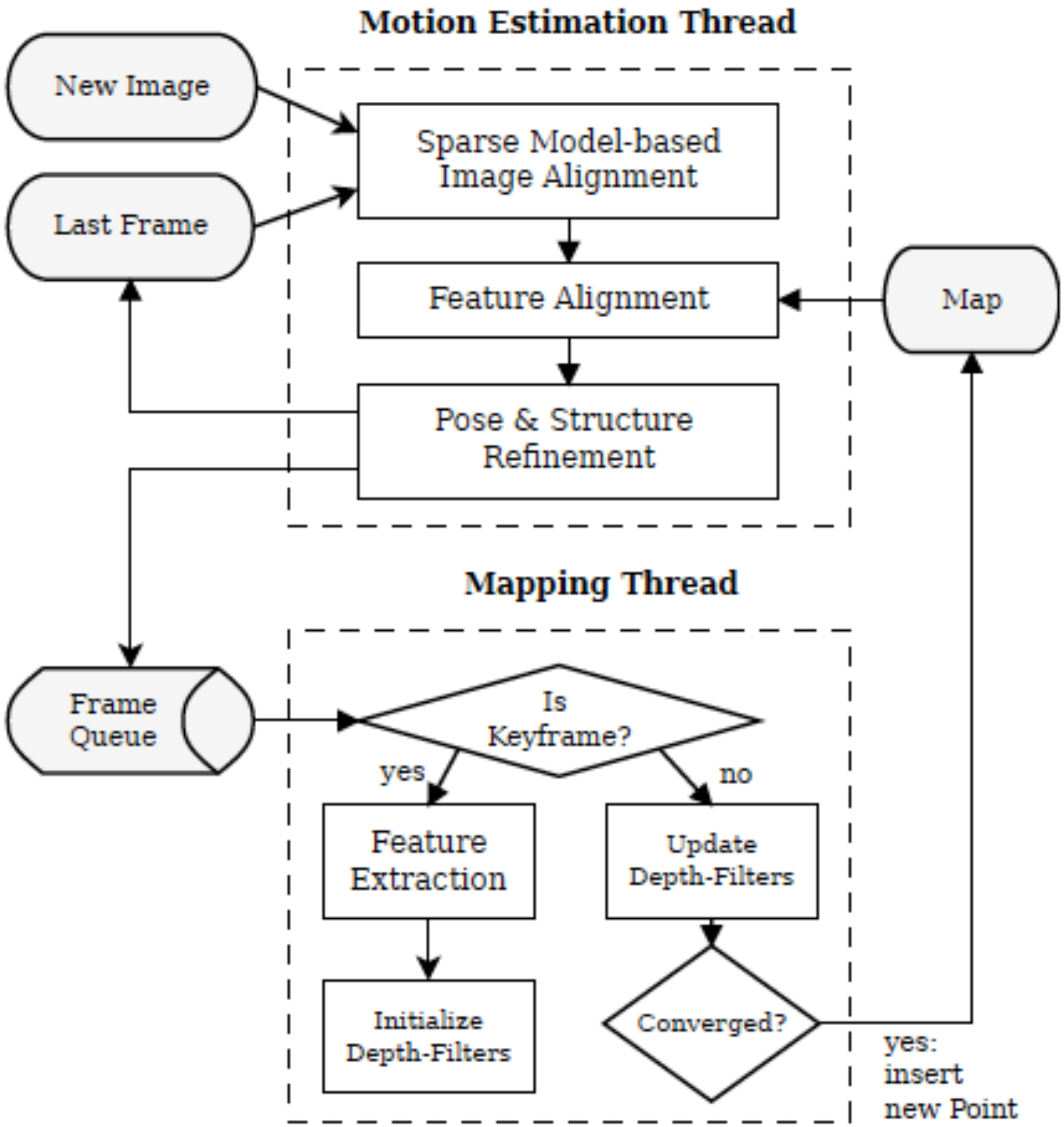


Fig. 1: Tracking and mapping pipeline

Quick and efficient

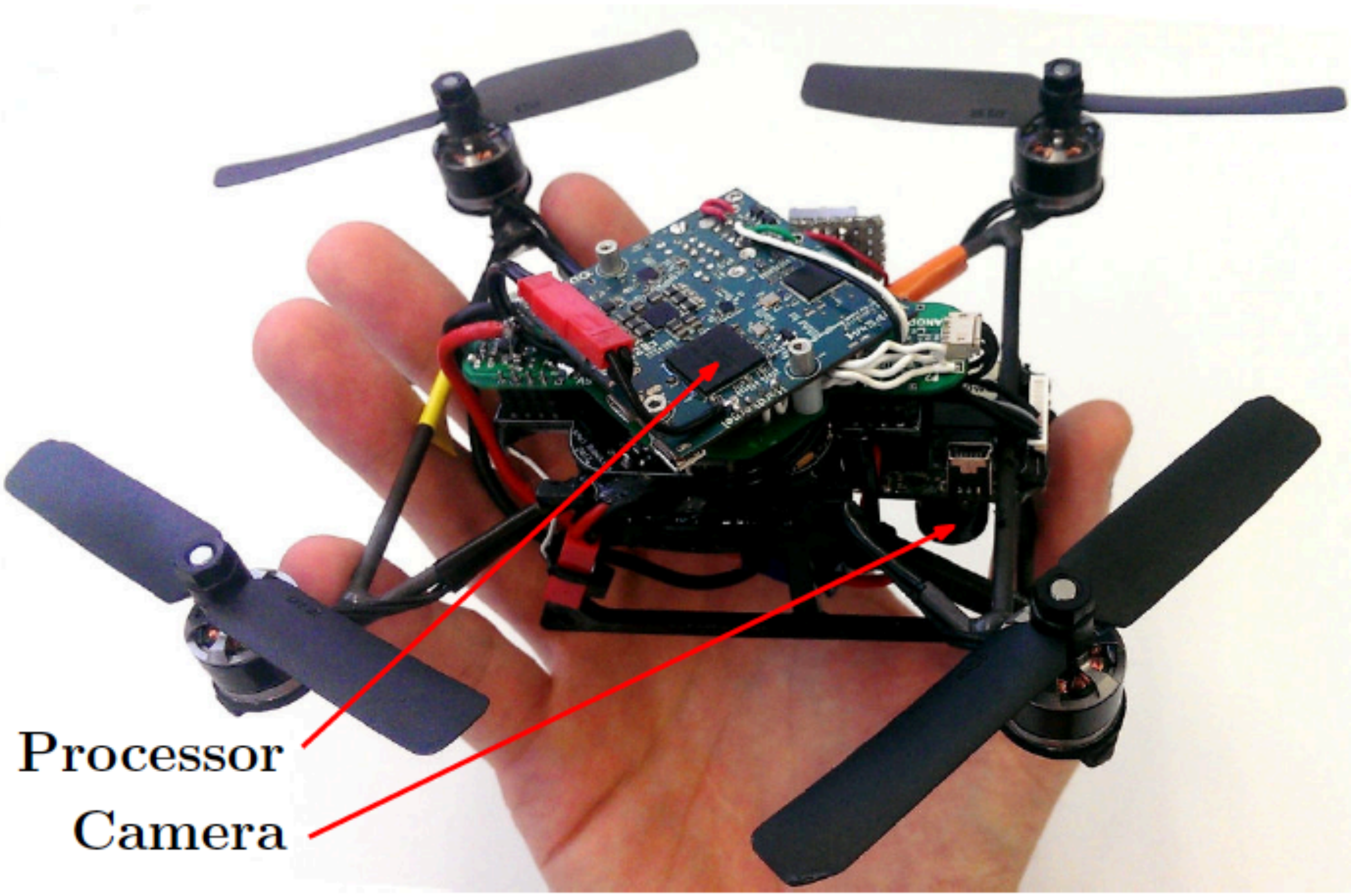


Fig. 17: "Nano+" by KMel Robotics, customized with embedded processor and downward-looking camera. SVO runs at 55 frames per second on the platform and is used for stabilization and control.

LSD-SLAM

- Essential point:
  - careful use of keyframes speeds things up a lot
  - Monocular cameras
    - direct
      - to predict image positions, we need depths
    - Recall:
      - keyframes are fine
    - Discovery:
      - keyframe depths are enough
      - pose graph:
        - key frames (nodes) linked by transforms (edges)

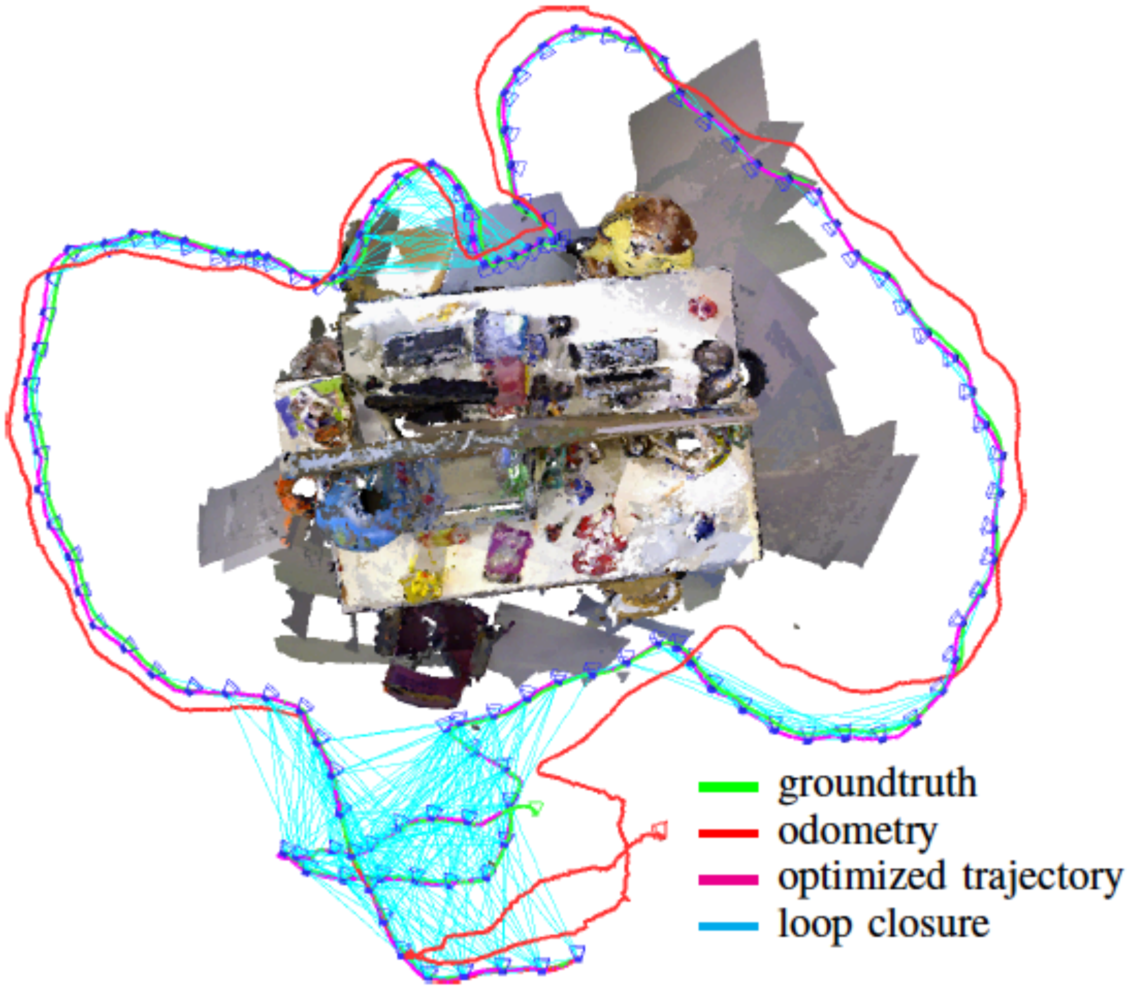


Fig. 1: We propose a dense SLAM method for RGB-D cameras that uses keyframes and an entropy-based loop closure detection to eliminate drift. The figure shows the groundtruth, frame-to-keyframe odometry, and the optimized trajectory for the fr3/office dataset.

Essential steps

- Compare new frame to keyframe
  - which has known depths
  - compute R, T, from
    - photometric error and depth error
      - recall:
        - depth -> image match -> photometric error
  - (could adjust depths in keyframe using R,T, photometric error)
- Keyframe selection
  - even sampling OR
  - entropy of R,T from last keyframe to current frame j = H_j
    - look at H_j/H_1
    - ie am I getting bad at computing motion?
    -

Essential steps

- Loop Closure
  - match keyframes
- Map
  - pose graph
- Start
  - how do I get depth for first keyframe?
    - doesn't seem to matter - random initialization
      - as long as you refine the depths
    - (roughly) stereo matching yields depth

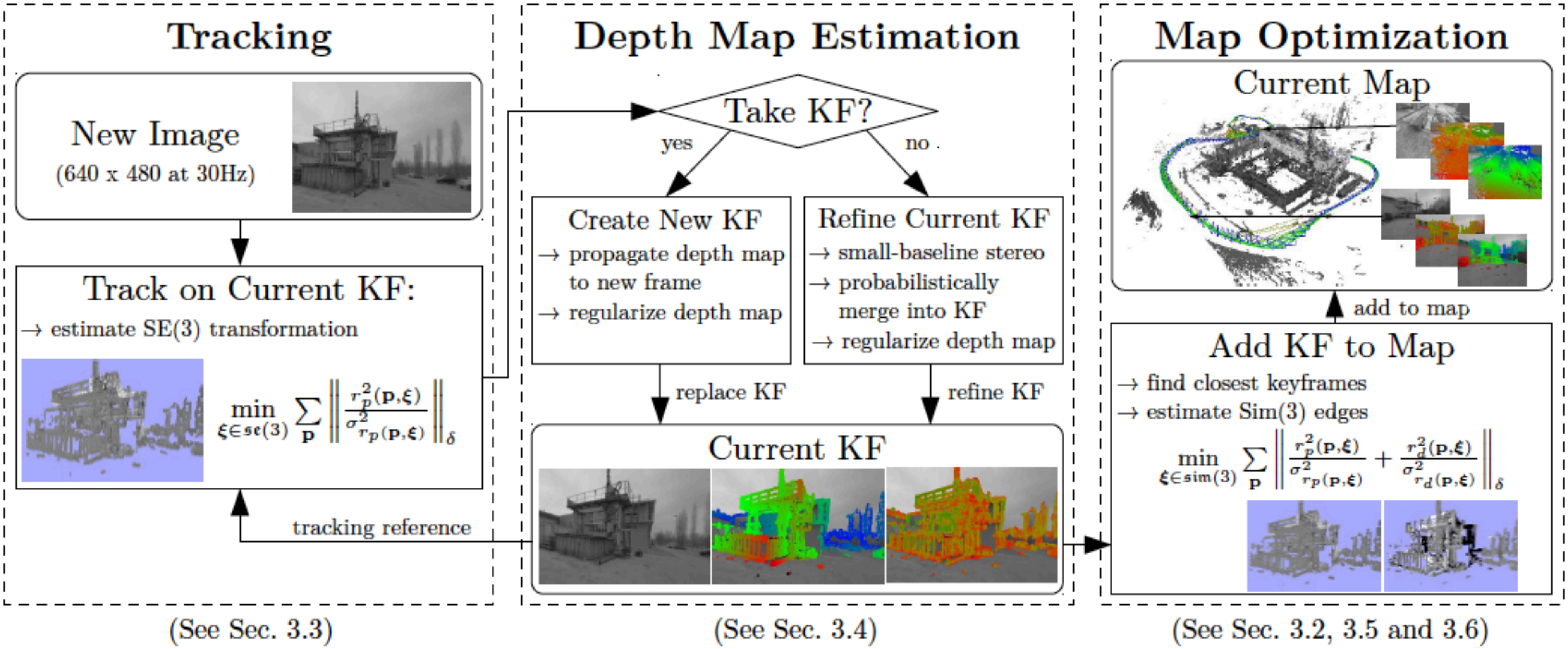


Fig. 3: Overview over the complete LSD-SLAM algorithm.

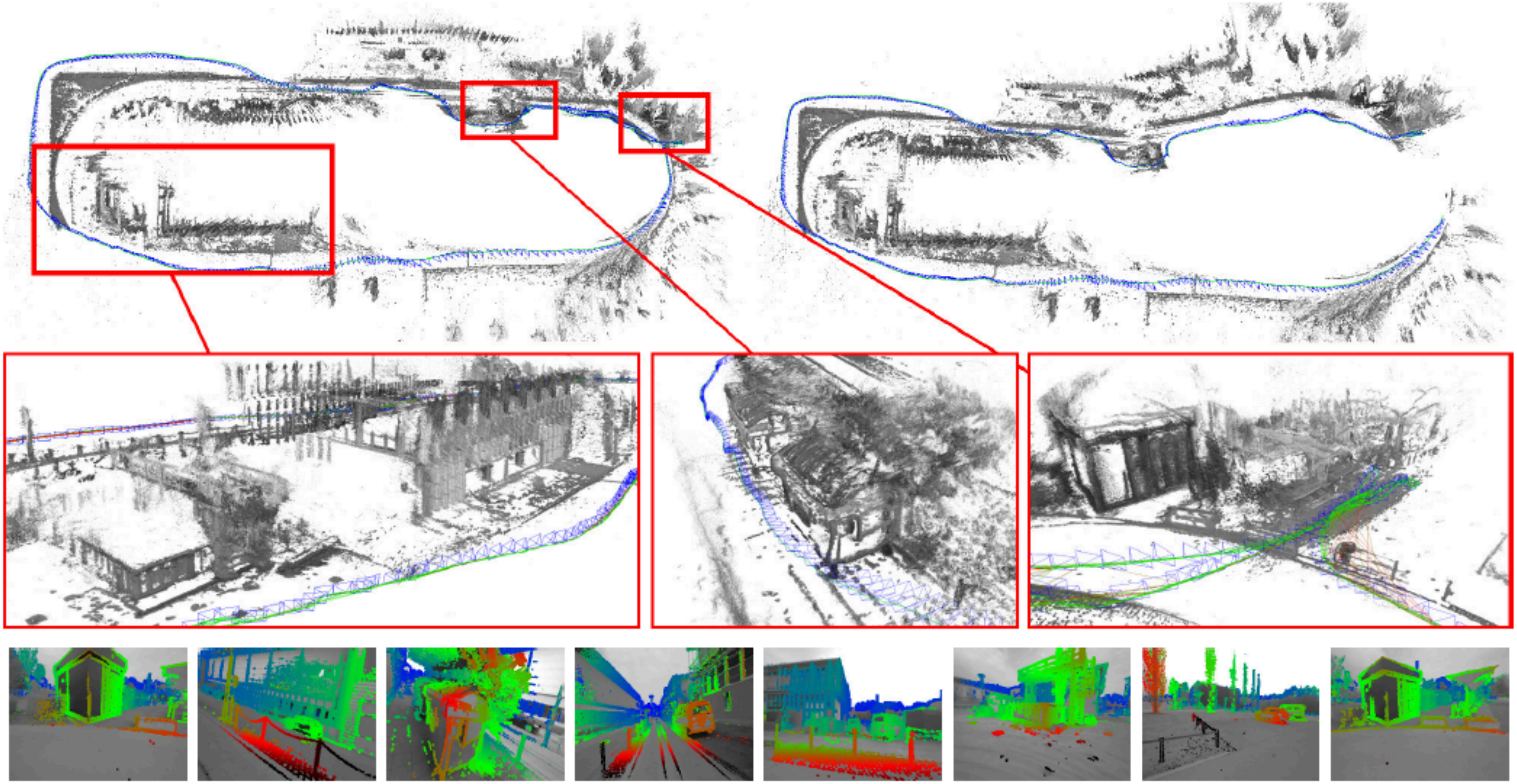


Fig. 7: Loop closure for a long and challenging outdoor trajectory (after the loop closure on the left, before on the right). Also shown are three selected close-ups of the generated pointcloud, and semi-dense depth maps for selected keyframes.

More…

https://vision.in.tum.de/research/vslam/lsdslam

References on web page