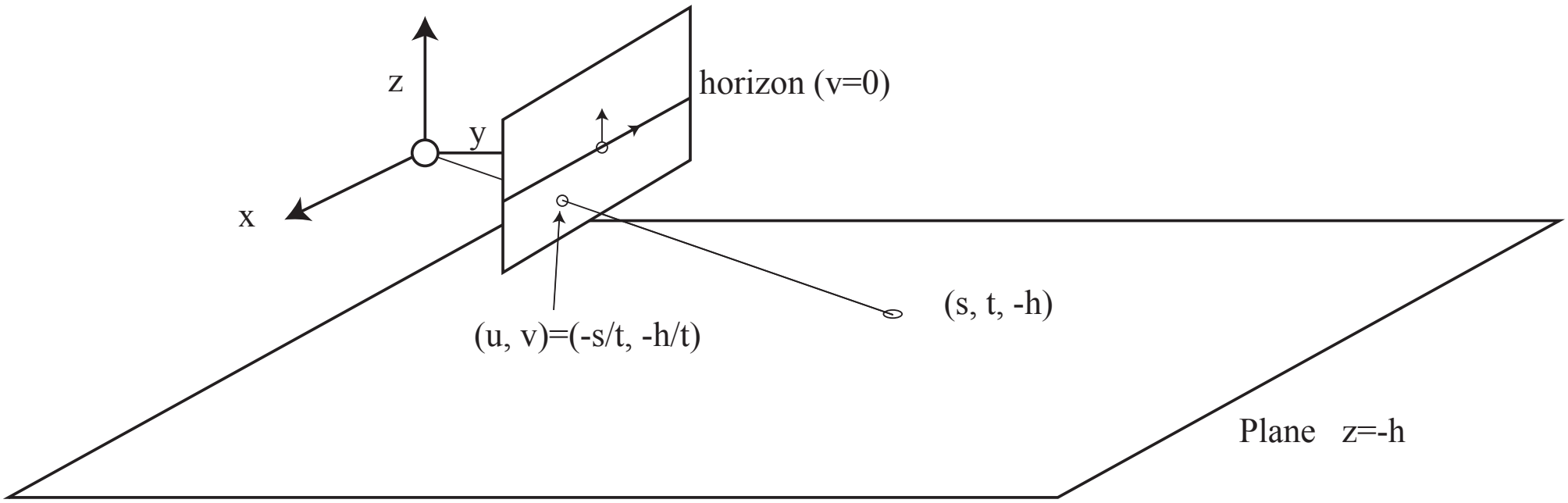
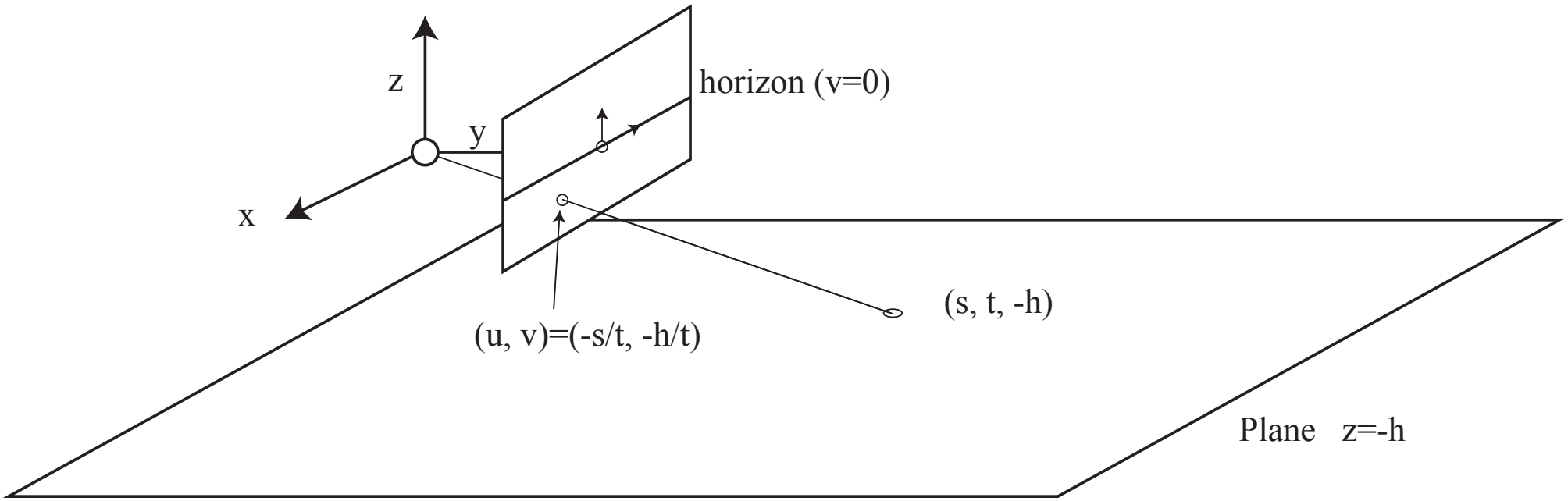
Direct Slam

D.A. Forsyth, UIUC

Basic direct method

- Imagine a camera at a fixed height
  - moving rigidly over a textured ground plane
  - bottom half of image is distorted ground plane texture
  - Q: when camera moves, how does distortion change?





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

Image coordinates

Plane texture coords

How the image distorts

- Camera moves relative to ground plane

Original texture

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix}_2 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathcal{R}_{2D} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} s \\ t \\ -h \end{pmatrix}$$

Motion

New image coords

- And we get an image transformation

Original Image

Motion

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix}_2 = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathcal{R}_{2D} & \mathbf{T} \\ \mathbf{0} & 1 \end{pmatrix} \left[\begin{pmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \right]^{-1} \begin{pmatrix} U \\ V \\ W \end{pmatrix}_1$$

New image coords

BUT - the texture hasn't changed

- So

$$I\left(\frac{U_2(\mathcal{R},\mathbf{T})}{W_2(\mathcal{R},\mathbf{T})},\frac{V_2(\mathcal{R},\mathbf{T})}{W_2(\mathcal{R},\mathbf{T})}\right) \quad \text{should be the same as} \quad I\left(\frac{U_1}{V_1},\frac{U_2}{V_2}\right)$$

- Idea:
  - minimize

$$\left[I\left(\frac{U_2(\mathcal{R},\mathbf{T})}{W_2(\mathcal{R},\mathbf{T})},\frac{V_2(\mathcal{R},\mathbf{T})}{W_2(\mathcal{R},\mathbf{T})}\right) - I\left(\frac{U_1}{V_1},\frac{U_2}{V_2}\right)\right]^2$$

  - as a function of the rotation and translation

Note: This gives odometry, and a form of map

Issues

- Minimize how?
  - Typically, Newton's method or a variant
- What about robustness?
  - use an m-estimator, as in IRLS
- How to initialize?
  - you could use interest points…
  - but if movements are small you might not need to
- Why?
  - massively improved estimates of rotation and translation IF you can min
  - because very large numbers of points contribute
- Generalize
  - camera moving in 3D and viewing a plane - easy, from drawing
  - 3D world - more interesting

Direct SLAM in a 3D world

- 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)

Direct SLAM in a 3D world - I

- Monocular cameras
  - semi-direct
    - for the moment, feature points
  - to predict image positions, we need depths
  - Discovery:
    - They're not required for every frame
      - a good R,T estimate comes from correspondence
      - keyframes and depths refine
    - and keyframe depths can be refined.

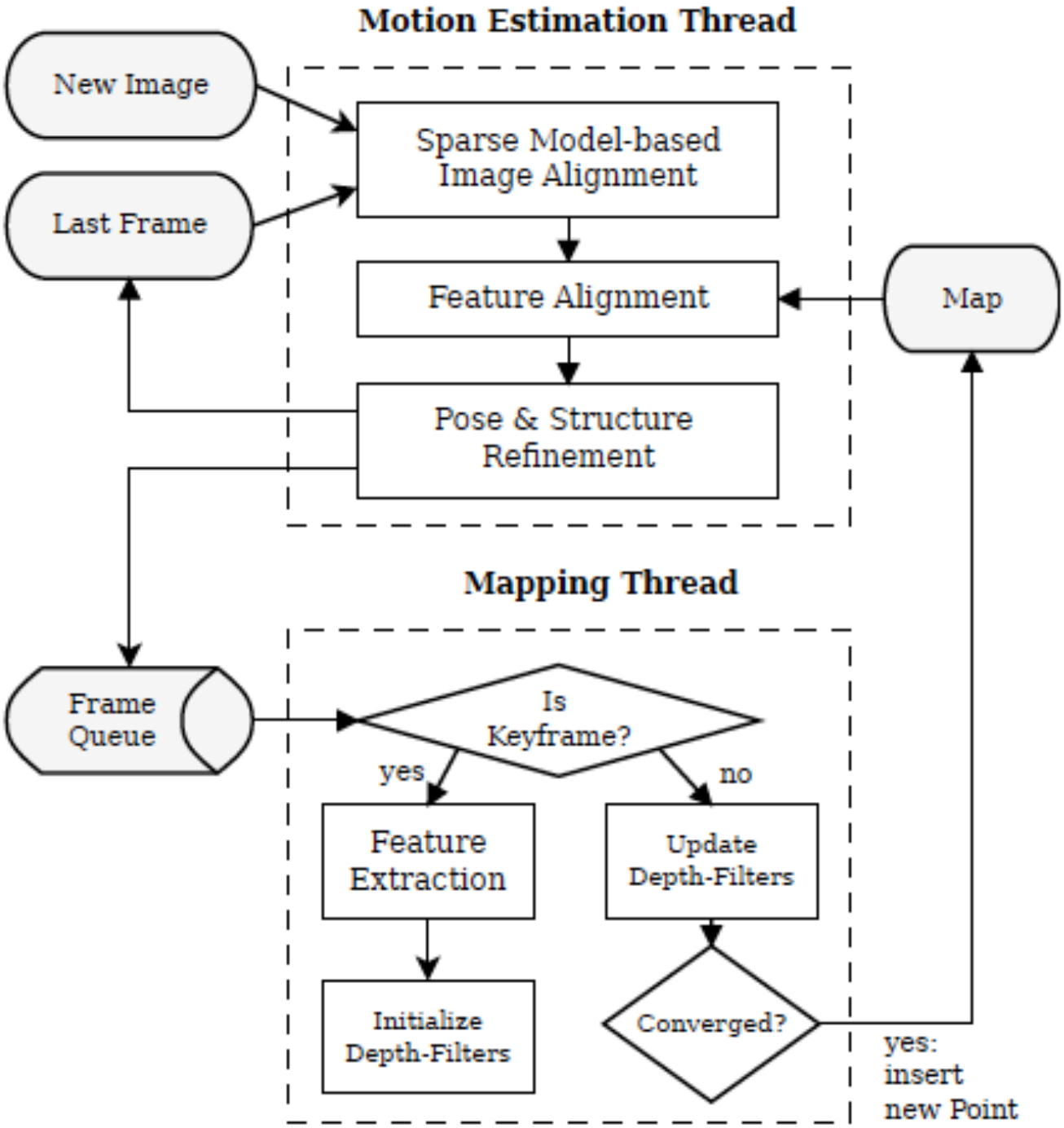


Fig. 1: Tracking and mapping pipeline

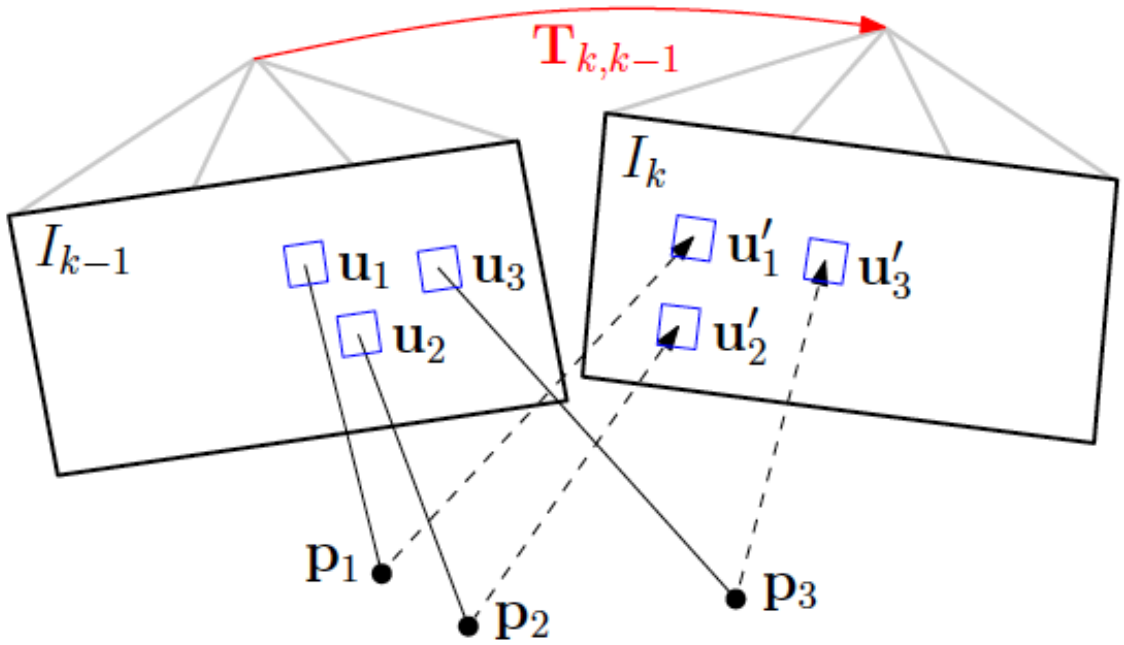


Fig. 2: Changing the relative pose $\mathbf{T}_{k,k-1}$ between the current and the previous frame implicitly moves the position of the reprojected points in the new image $\mathbf{u}'_i$. Sparse image alignment seeks to find $\mathbf{T}_{k,k-1}$ that minimizes the photometric difference between image 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.

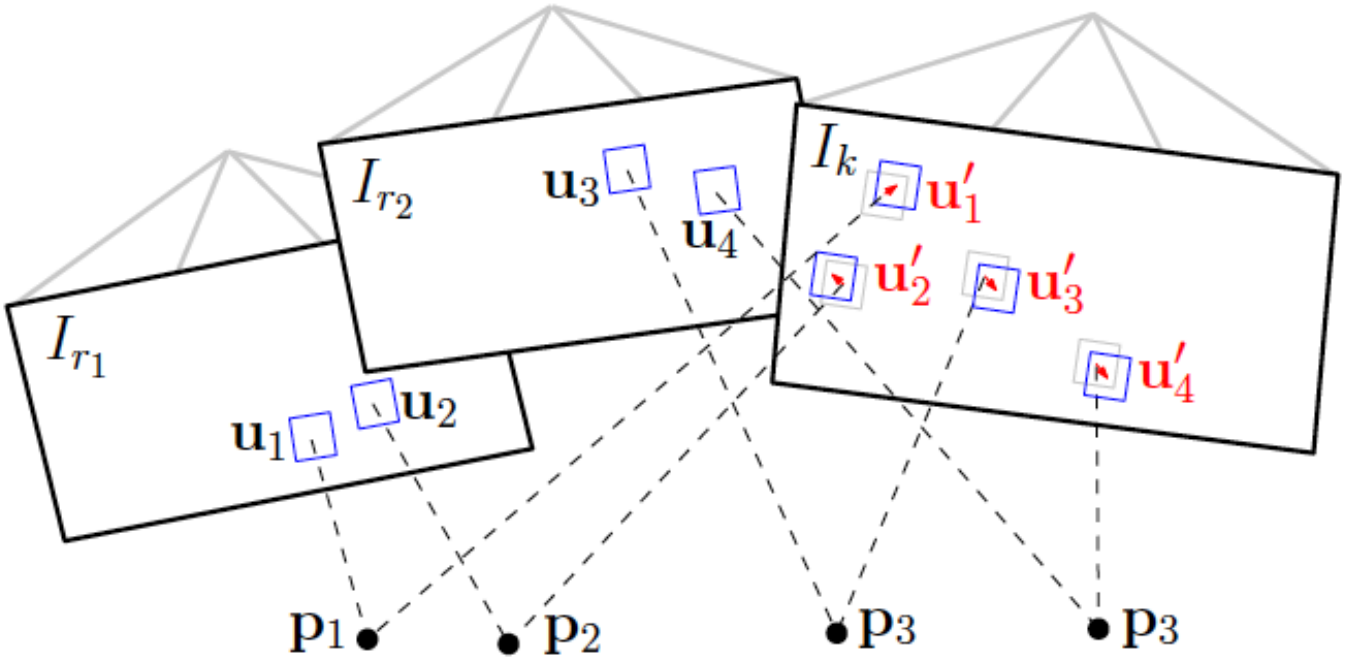


Fig. 3: Due to inaccuracies in the 3D point and camera pose estimation, the photometric error between corresponding patches (blue squares) in the current frame and previous keyframes r_i can further be minimised by optimising the 2D position of each patch individually.

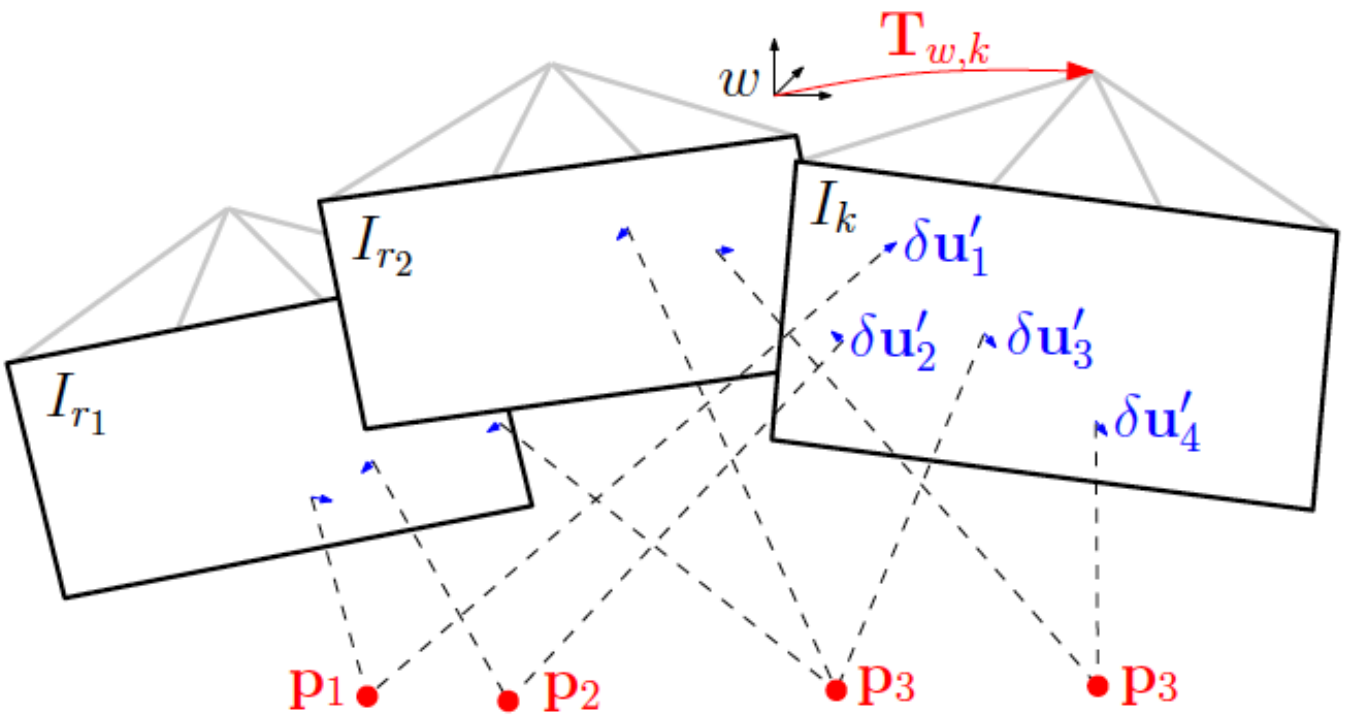


Fig. 4: In the last motion estimation step, the camera pose and the structure (3D points) are optimized to minimize the reprojection error that has been established during the previous feature-alignment step.

Quick and efficient

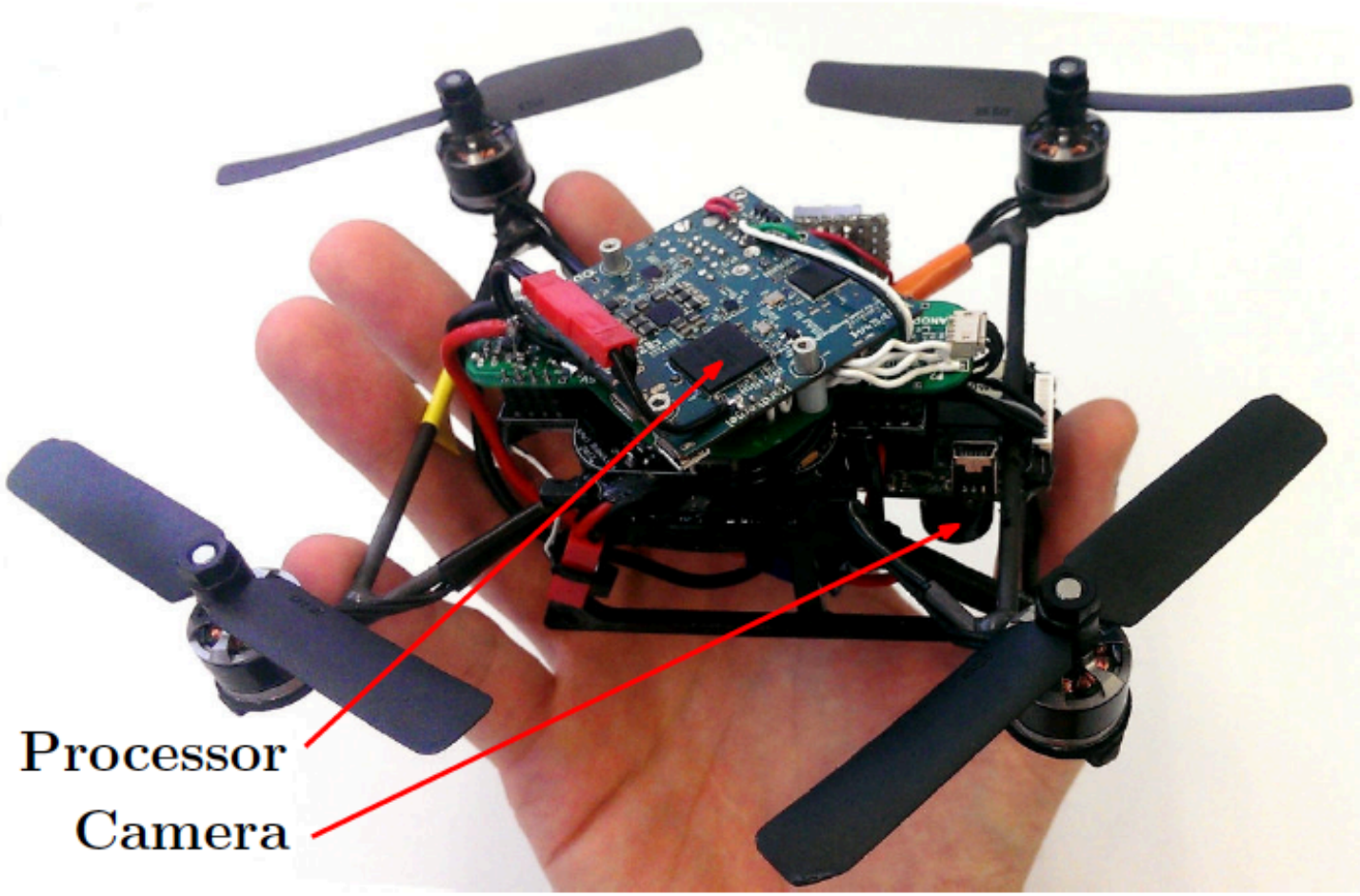


Fig. 17: "Nano+" by KMel Robotics, customized with embedded processor and downward-looking camera. SVO runs at 55 frames per second on the platform and is used for stabilization and control.

Direct SLAM in a 3D world - I

- 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)

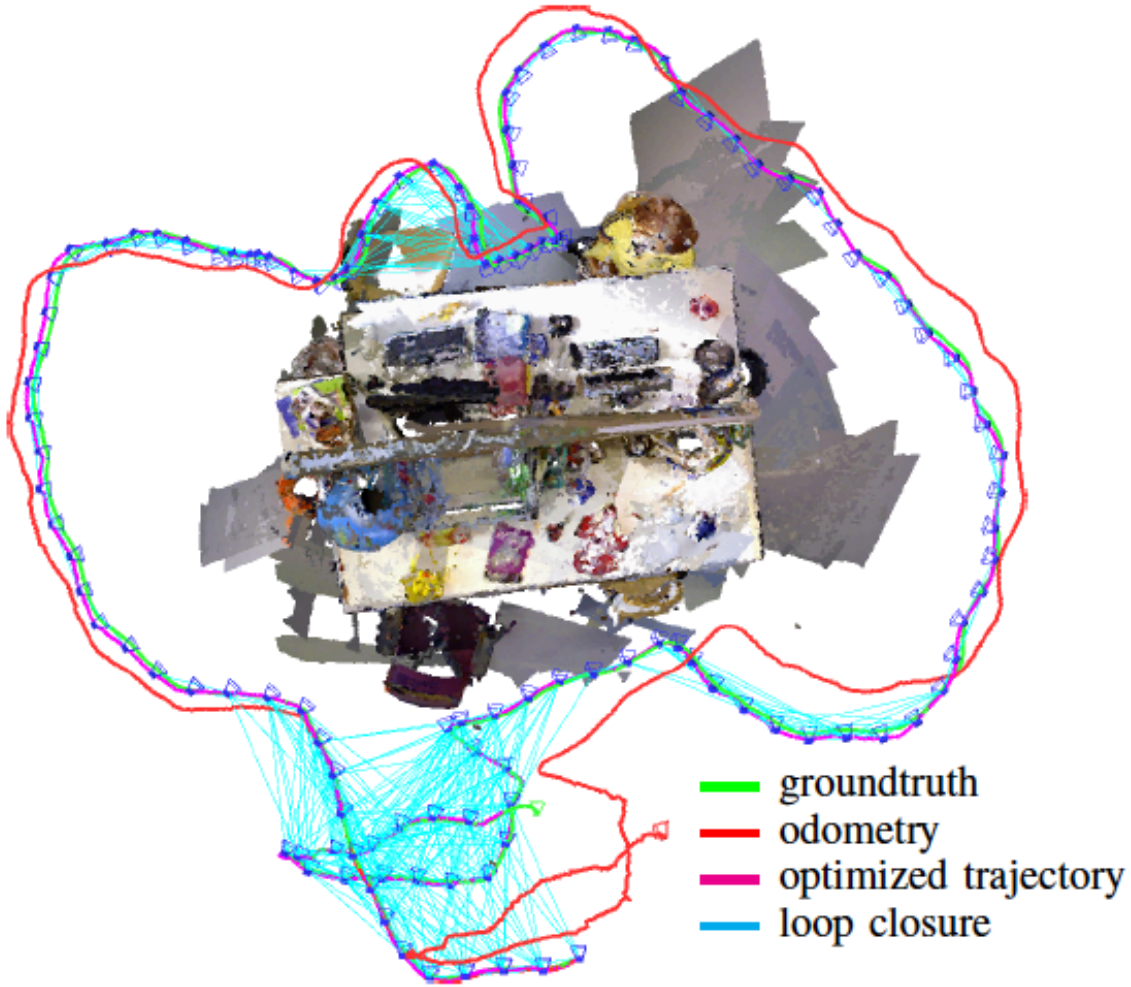


Fig. 1: We propose a dense SLAM method for RGB-D cameras that uses keyframes and an entropy-based loop closure detection to eliminate drift. The figure shows the groundtruth, frame-to-keyframe odometry, and the optimized trajectory for the fr3/office dataset.

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

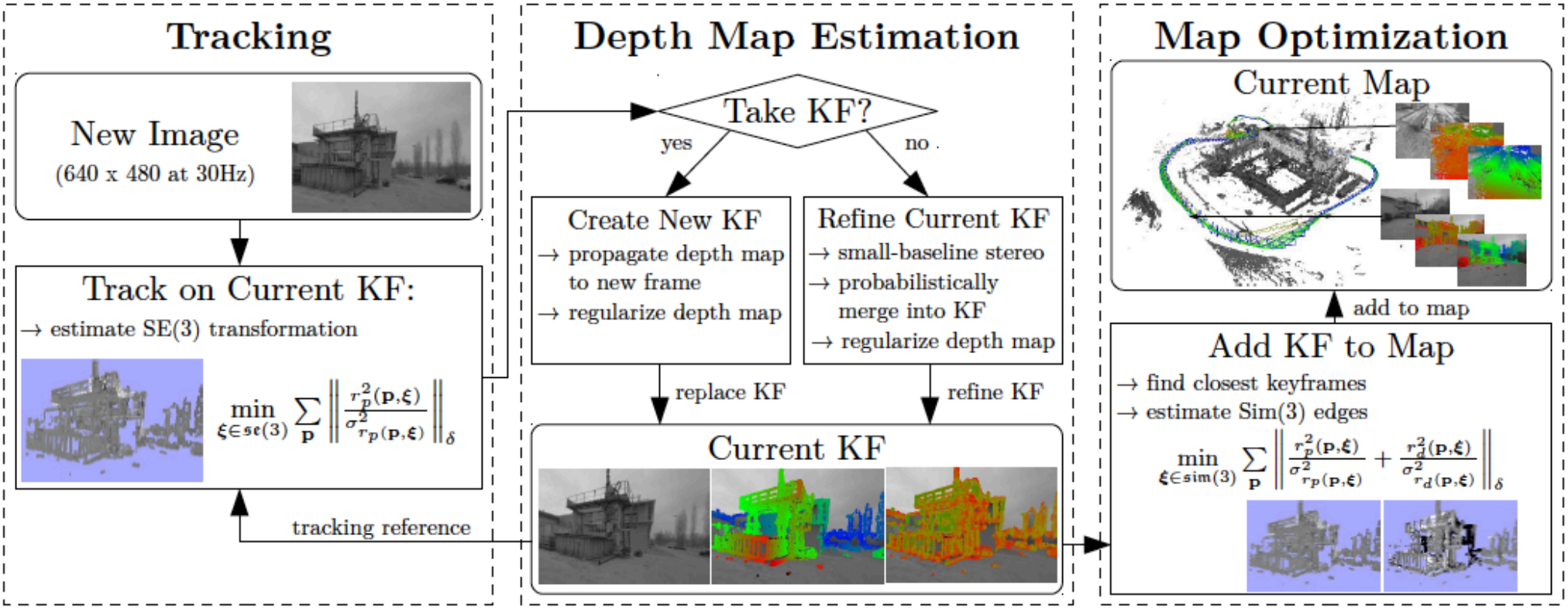


Fig. 3: Overview over the complete LSD-SLAM algorithm.

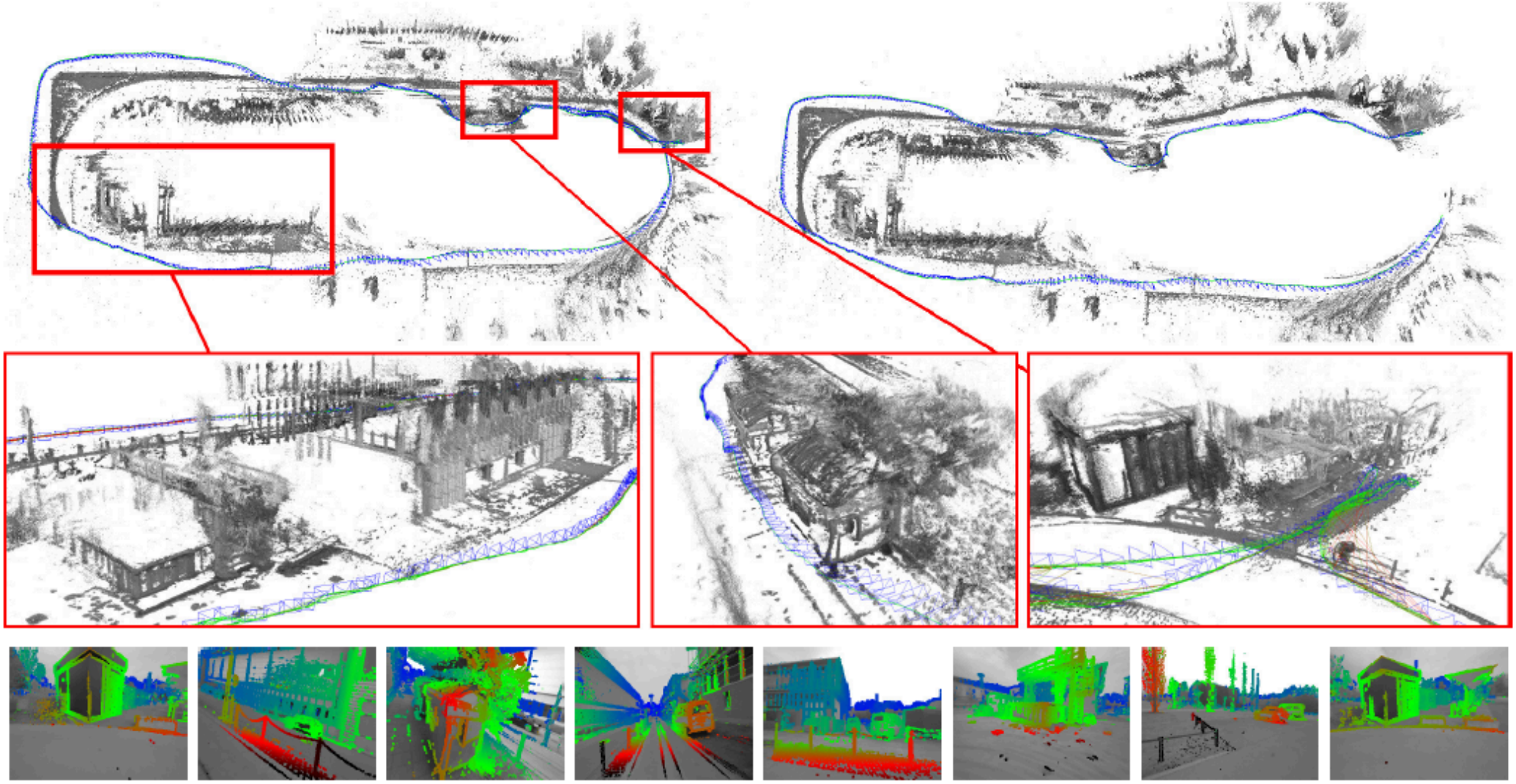


Fig. 7: Loop closure for a long and challenging outdoor trajectory (after the loop closure on the left, before on the right). Also shown are three selected close-ups of the generated pointcloud, and semi-dense depth maps for selected keyframes.

More…

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

References on web page