Parallel Tracking and Mapping for Small AR Workspaces

The Evolution of Motion Estimation and Real-time 3D Reconstruction

Iuliia Skobleva

Main Idea

estimate camera's position in an unknown setting

tracking camera motion

producing a 3D map

Previous Methods

- MonoSLAM and Scalable Monocular SLAM were state-of-the-art
- Mostly used for robots where it receives odometery and can be driven slowly
- Data-association becomes a problem when tracking a hand-held camera
- Neither provided enough robustness for AR applications

- Separate Tracking and Mapping
- No need for processing every frame when mapping

4. Ewok rampage

Here the camera is used to aim Darth Vader's laser pistol. Movement is controlled with the keyboard.

Timeline

Augmented Reality

The enabling technology for this access interface is a heads-up (see-thru) display head set (we call it the "HUDset"), combined with head position sensing and workplace registration systems. This technology is used to "augment" the visual field of the user with information necessary in the performance of the current task, and therefore we refer to the technology as "augmented reality" (AR).

Parallel Tracking and Mapping for Small AR Workspaces

Georg Klein* David Murray†

1968

1992

29th June 2007

13th Nov 2007

2016

Now

Pokémon GO

Algorithm

Algorithm Overview

1. The map is densely initialised from a stereo pair
2. New points are initialised with an epipolar search
3. Mapping is based on keyframes, which are processed using batch techniques

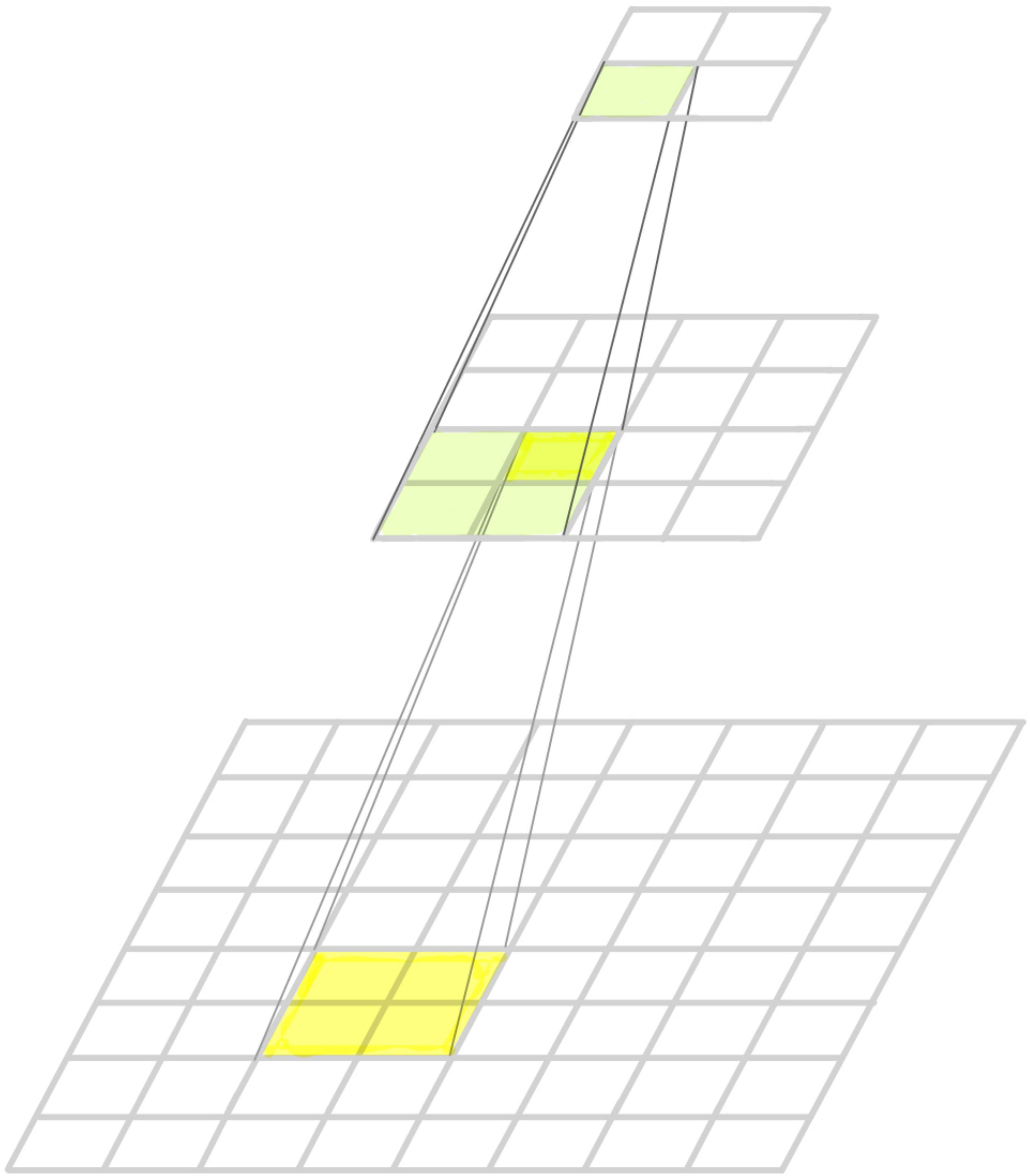
the scene must be mostly static

the scene must be mostly small

Notation

Before we get started

- The map contains N keyframes
- Each keyframe stores a four-level pyramid of greyscale images
- The map consists of M point features in the world coordinate frame
- Each point feature represents a patch



Tracking

Tracking

Initial Steps

1. Tracking system constructs a four-level image pyramid
2. FAST corner detector is run on each pyramid level

Tracking

Projection of Map Points

1. **A new frame is acquired from the camera, and a prior pose estimate is generated from a motion model**

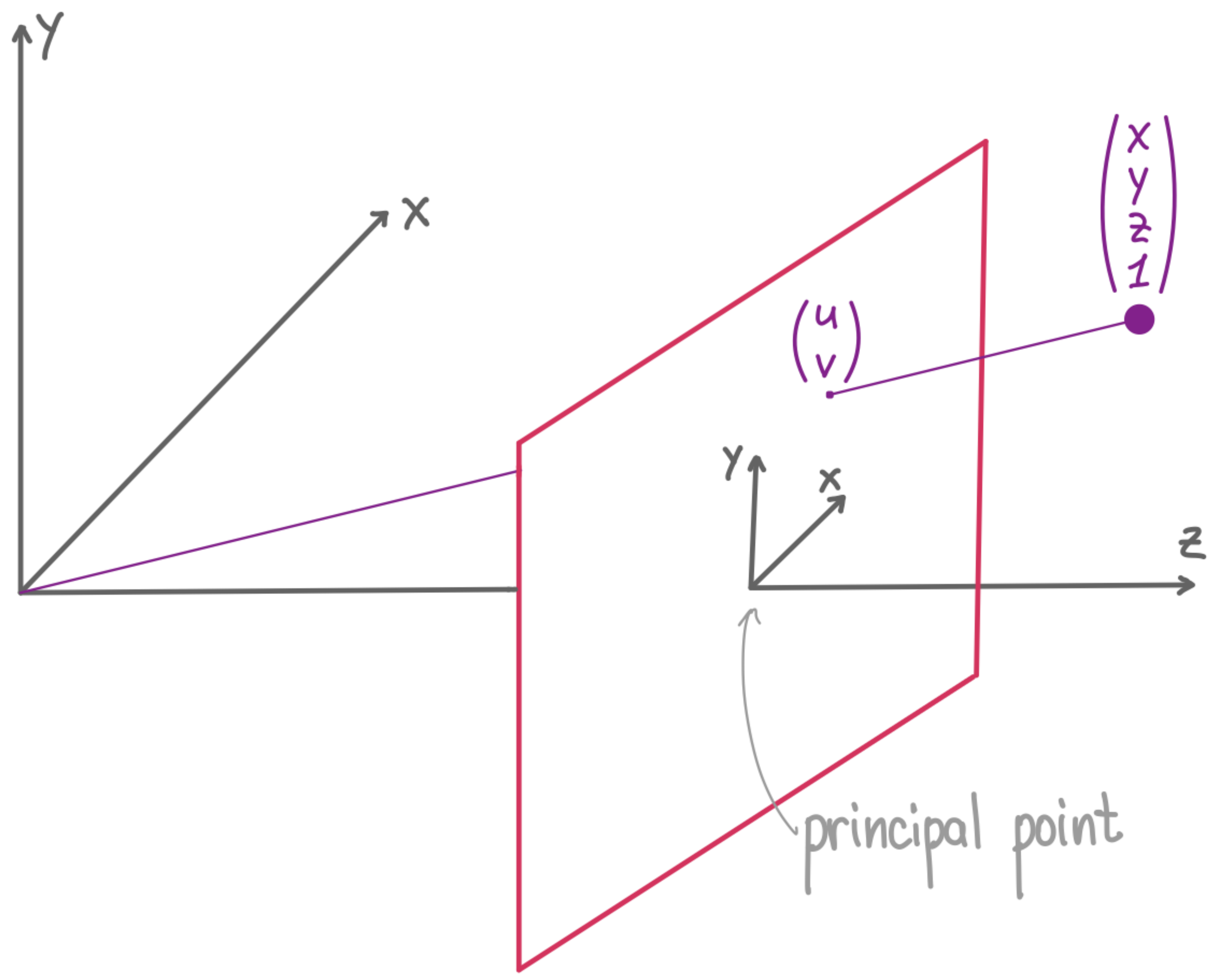
$$\mathbf{p}_{iC} = E_{CW}\mathbf{p}_{iW} \qquad E_{CW} = \exp(\hat{\boldsymbol{\mu}})$$

2. **Map points are projected onto the image according to the frame's prior pose estimate**

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \text{CamProj}(E_{CW}\mathbf{p}_{iW})$$

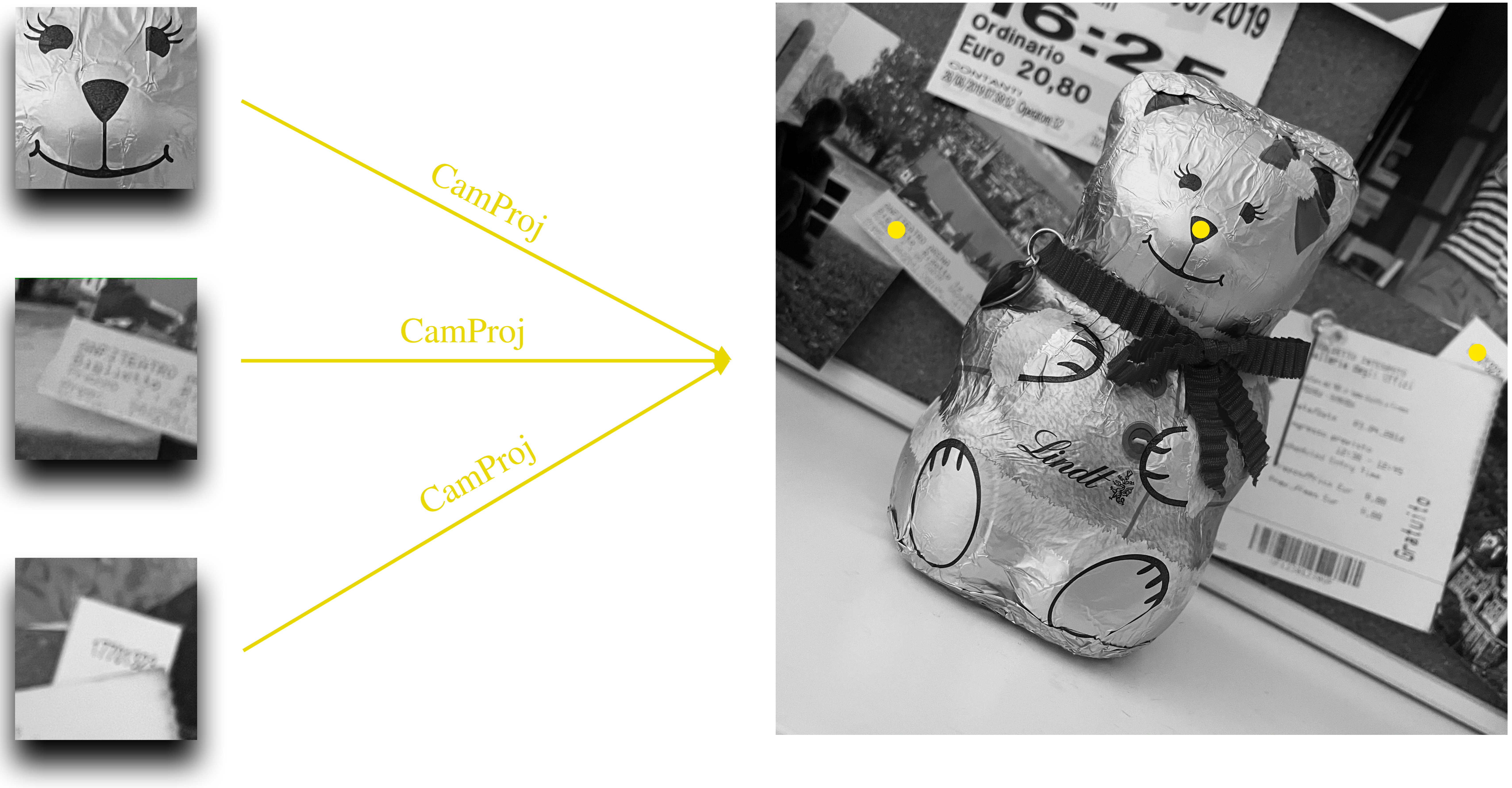
$$\text{CamProj}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \frac{r'}{r}\begin{pmatrix} f_u & 0 \\ 0 & f_v \end{pmatrix}\begin{pmatrix} \frac{x}{z} \\ \frac{y}{z} \end{pmatrix}$$

$$r = \sqrt{\frac{x^2+y^2}{z^2}} \qquad r' = \frac{1}{w}\arctan(2r\tan\frac{w}{2})$$



Tracking

Projection of Map Points



Tracking

Feature Search

3. **A small number (50) of the coarsest-scale features are searched for in the image**

- Take viewpoint changes into the account by warping *the patch* around the predicted location of a point
- Mean pixel intensity is subtracted to increase robustness against lightning changes




Tracking

Feature Search

3. A small number (50) of the coarsest-scale features are searched for in the image

- Take viewpoint changes into the account by warping *the patch* around the predicted location of a point
- Mean pixel intensity is subtracted to increase robustness against lightning changes
- Use SSD scores at FAST corner locations to find a match for the patch

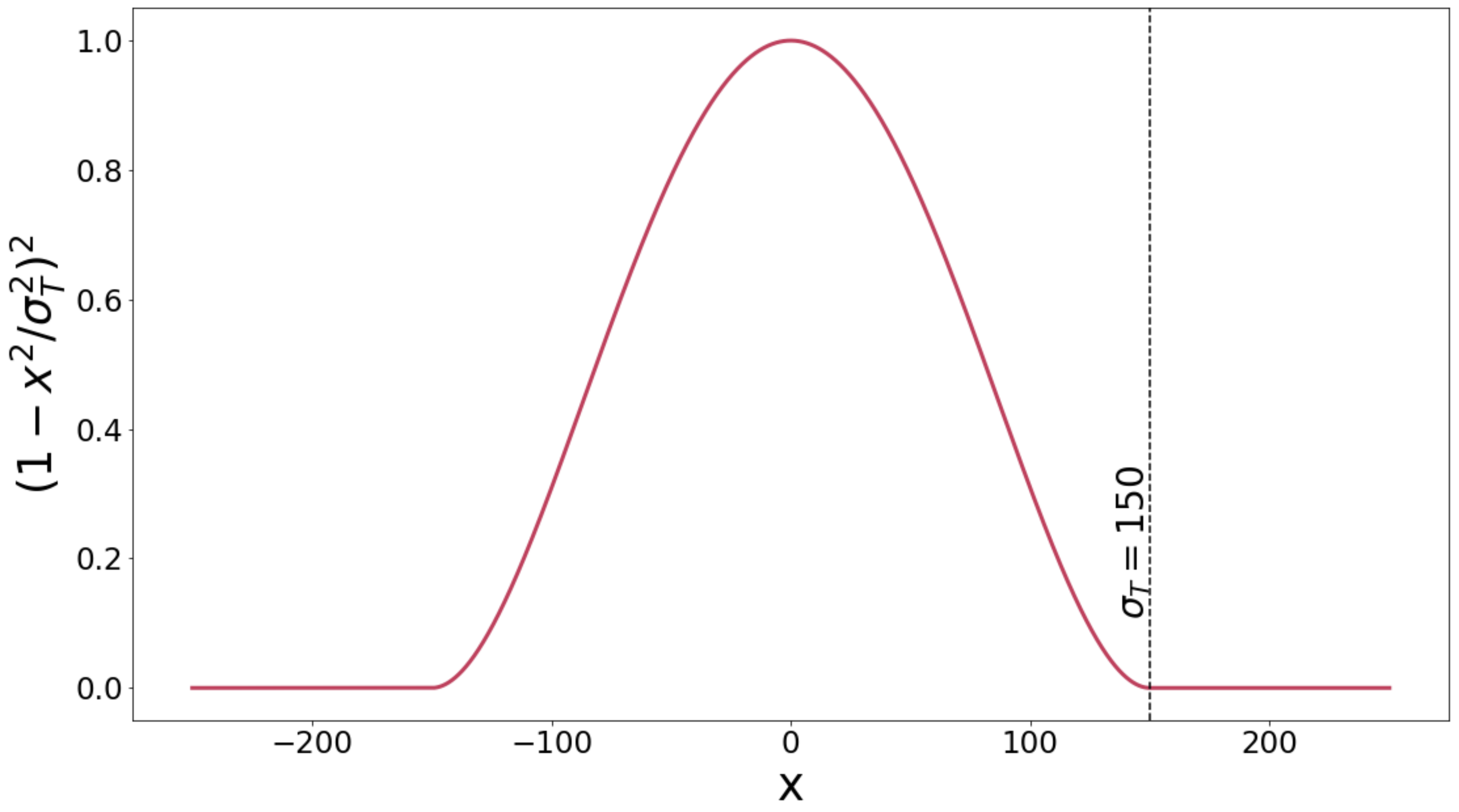
Tracking

Pose Update

4. **Camera pose is updated from these coarse matches**

$$\boldsymbol{\mu}' = \operatorname{argmin} \sum_{i \in S} \mathrm{Obj}\left(\frac{|\boldsymbol{e}_i|}{\sigma_i}, \sigma_T\right)$$

$$\boldsymbol{e}_i = \begin{pmatrix} \hat{u}_i \\ \hat{v}_i \end{pmatrix} - \mathrm{CamProj}(\exp(\boldsymbol{\mu}) E_{CW} \boldsymbol{p}_i)$$



$(\hat{u}_i \quad \hat{v}_i)$ is the found patch position

$\mathrm{Obj}(\,\cdot\,, \sigma_T)$ is the Tukey objective bi-weight function

σ_T is a robust (median-based) estimate of the distribution's standard deviation derived from all the residuals

Tracking

Search & Pose Update for more Points

5. A large number of points (1000) is re-projected and searched for in the image
6. A final pose estimate for the frame is computed from all matches found

Mapping

Building a Map

Initialisation

- User takes pic #1
- User rotates and translates the camera
- User takes pic #2
- 1000 points are searched for using FAST corners
- Using 5-point algorithm
- To scale the map, assume the camera has moved 10cm

Building a Map

Keyframes

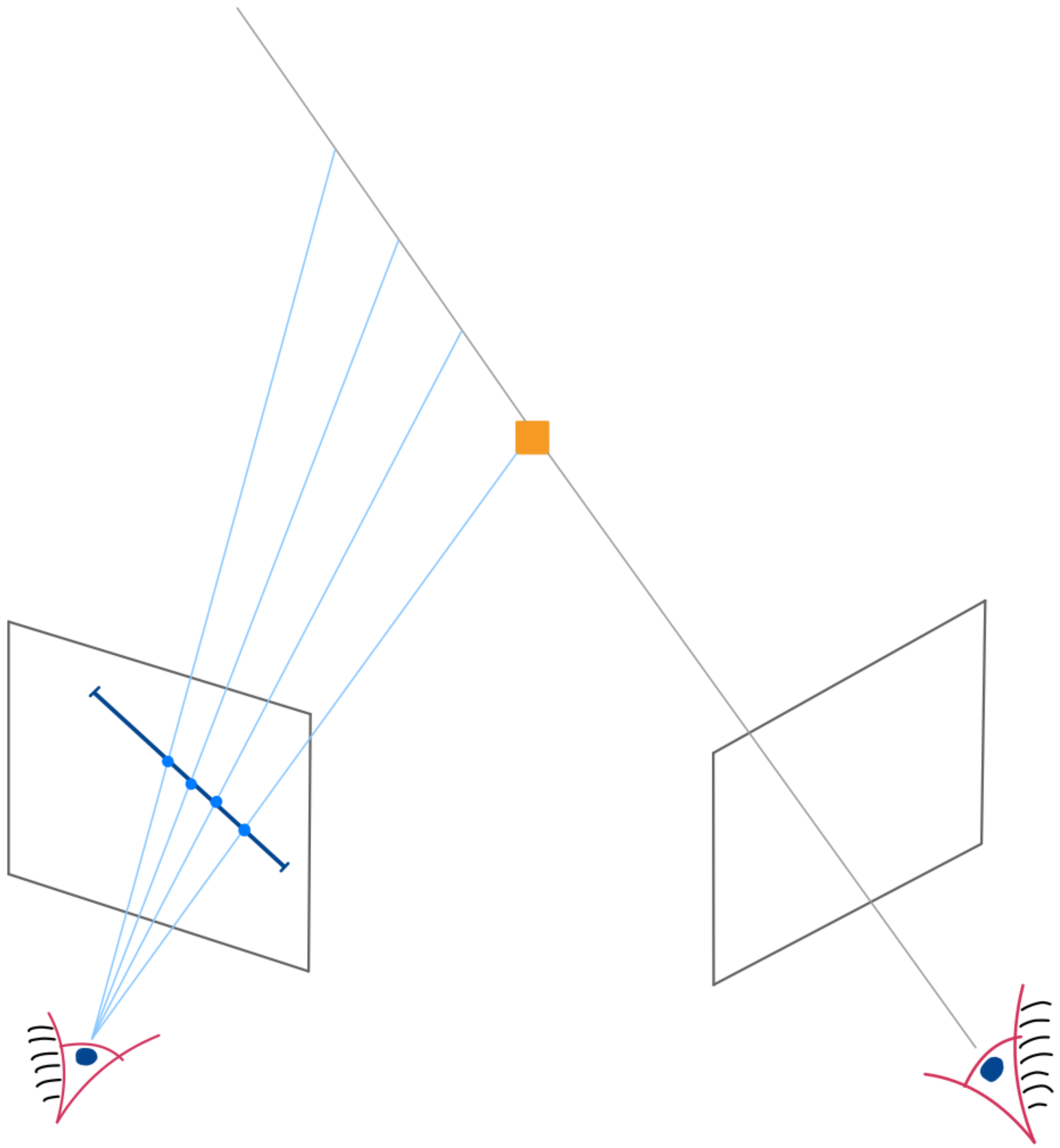
New keyframes are added when:

- Tracking quality is good *(fraction of successful feature observations)*
- 0.67 seconds have passed since taking the last keyframe
- Camera must be away a certain distance from nearest keypoint in the map
    - Distance depends on average depth of the observed features

Building a Map

Epipolar search

- Use FAST corners, which have already been calculated by the tracking system
- Use Shi-Tomasi scores to narrow down the set
- Select "new" points *(discard salient points near successful observations of existing features)*
- Acquire depth information using triangulation with the nearest keyframe
- In case of a successful match *(point exists in two frames)*, add the point to the map



Building a Map

Bundle Adjustment

- Use Levenberg-Marquardt Bundle Adjustment algorithm to refine the map for all keyframes:

$$\{\{\boldsymbol{\mu}_1 \ldots \boldsymbol{\mu}_N\}, \{\boldsymbol{p}'_1 \ldots \boldsymbol{p}'_M\}\} = \underset{\{\boldsymbol{\mu}\}, \{\boldsymbol{p}\}}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j \in S_i} \operatorname{Obj}\left(\frac{|\boldsymbol{e}_{ji}|}{\sigma_{ji}}, \sigma_T\right)$$

- Perform local Bundle Adjustment to decrease computation time:

$$\{\{\boldsymbol{\mu}_{x \in X}\}, \{\boldsymbol{p}'_{z \in Z}\}\} = \underset{\{\boldsymbol{\mu}\}, \{\boldsymbol{p}\}}{\operatorname{argmin}} \sum_{i \in X \cup Y}^{N} \sum_{j \in Z \cap S_i} \operatorname{Obj}\left(i, j\right)$$

X is the set of keyframes which are being adjusted (newest frame + four nearest ones)

Z is the set of all 3D points visible in X

Y is the further set of keyframes with projections of points from Z

Building a Map

Data Association Refinement

- Performed when bundle adjustment has converged
- Make new measurements in old keyframes
- Re-measure outliers *(frequently happens in regions with repeated patterns)*

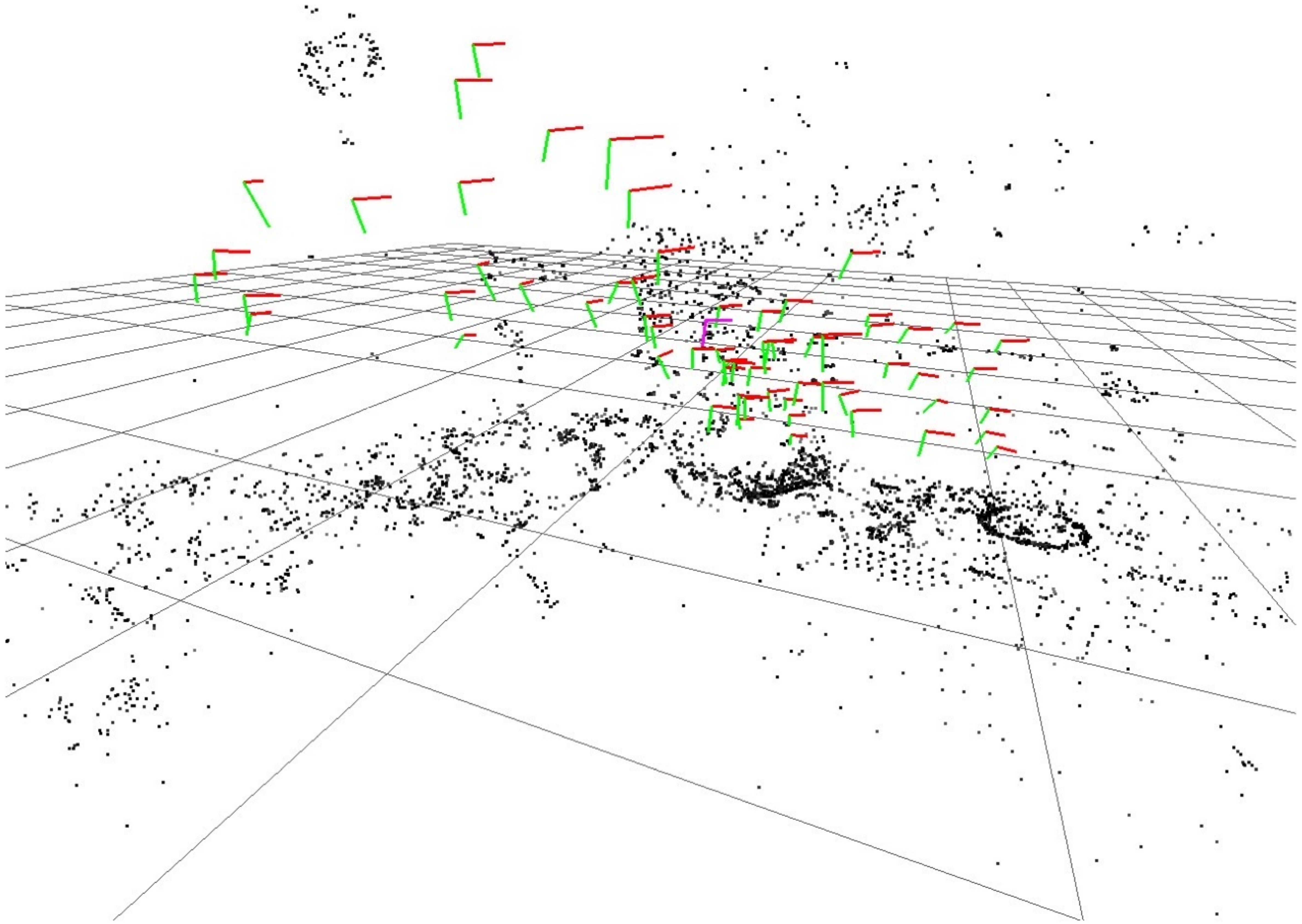
Results

Results

1. Tracking

This video shows the operation of the proposed tracking + mapping system. A map of the desktop is started from a single stereo pair and expanded as the camera moves about.

All videos are recorded live.



Results

Tracking Performance

Keyframe Preparation: frame capture, converting image to greyscale, building the image pyramid and FAST corner detection

Feature Projection: projection of map points onto image and selection of the most salient points

Task	Time
Keyframe Preparation	2.2ms
Feature Projection	3.5ms
Patch Search	9.8ms
Iterative Pose Update	3.7ms
Total	*19.2ms*

Results

Scalability

- Tracking scales linearly with increasing map size
- Mapping scales poorly with increasing number of map points and keyframes
- System remains stable under 6000 map points and 150 keyframes

	2-49 keyframes	50-99 keyframes	100-149 keyframes
Local Bundle Adjustment	170ms	270ms	440ms
Global Bundle Adjustment	380ms	1.7s	6.9s

Overview

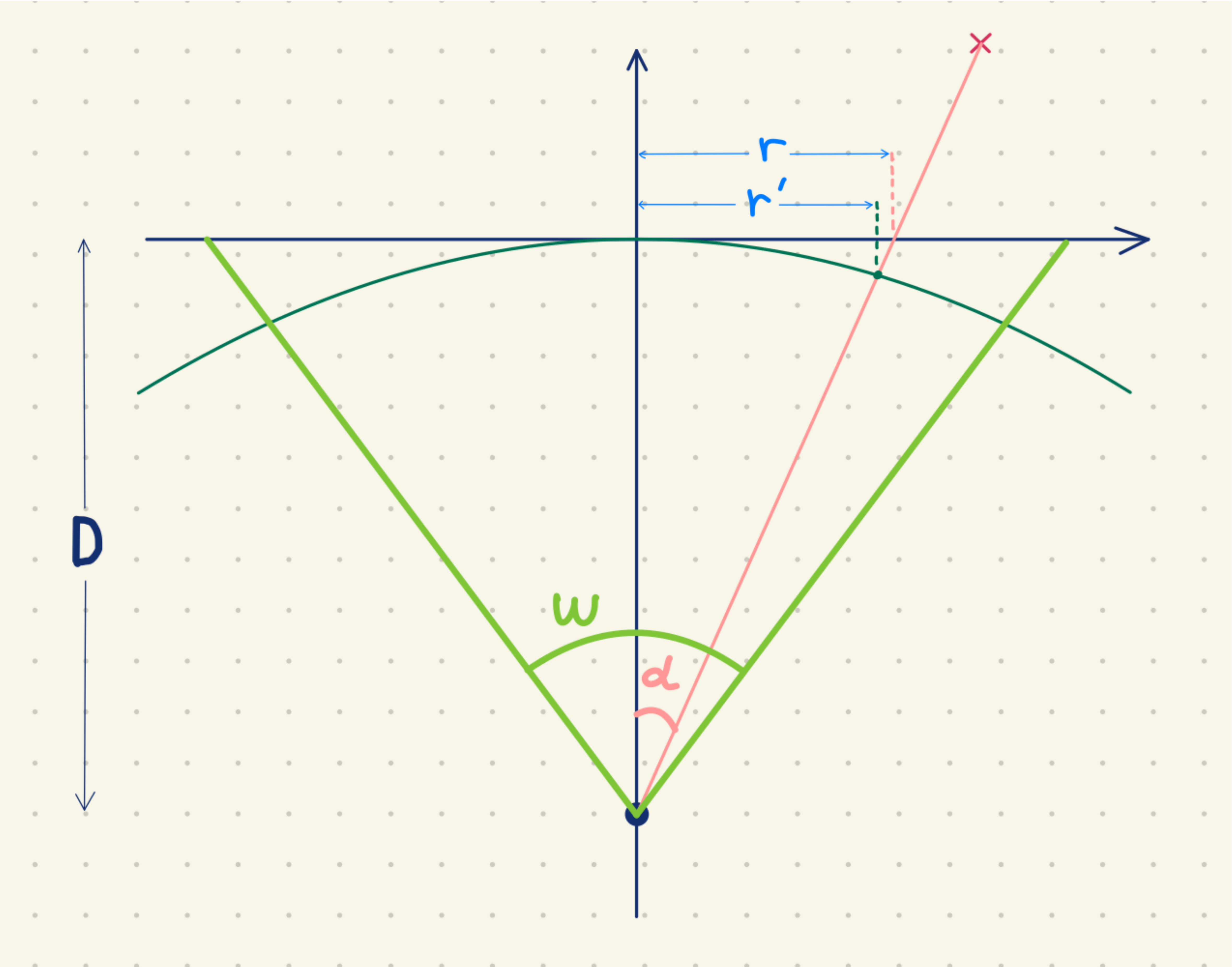
Problems & Future Work

- Could only be run on a computer and not on mobile platforms
- Requires user interaction
- The map bears no geometrical meaning
- Not possible to interact with features in the images
- Limited to small spaces
- Cannot handle loop closure

Thanks ☺️

Extra Slides

FOV Model



$$\alpha = \omega r'$$

$$\tan\frac{\omega}{2} = \frac{1}{2D}$$

$$\tan\alpha = \frac{r}{D}$$

$$\tan\alpha = 2r\tan\frac{\omega}{2}$$

$$\alpha = \arctan(2r\tan\frac{\omega}{2})$$

$$r' = \frac{1}{\omega}\arctan(2r\tan\frac{\omega}{2})$$

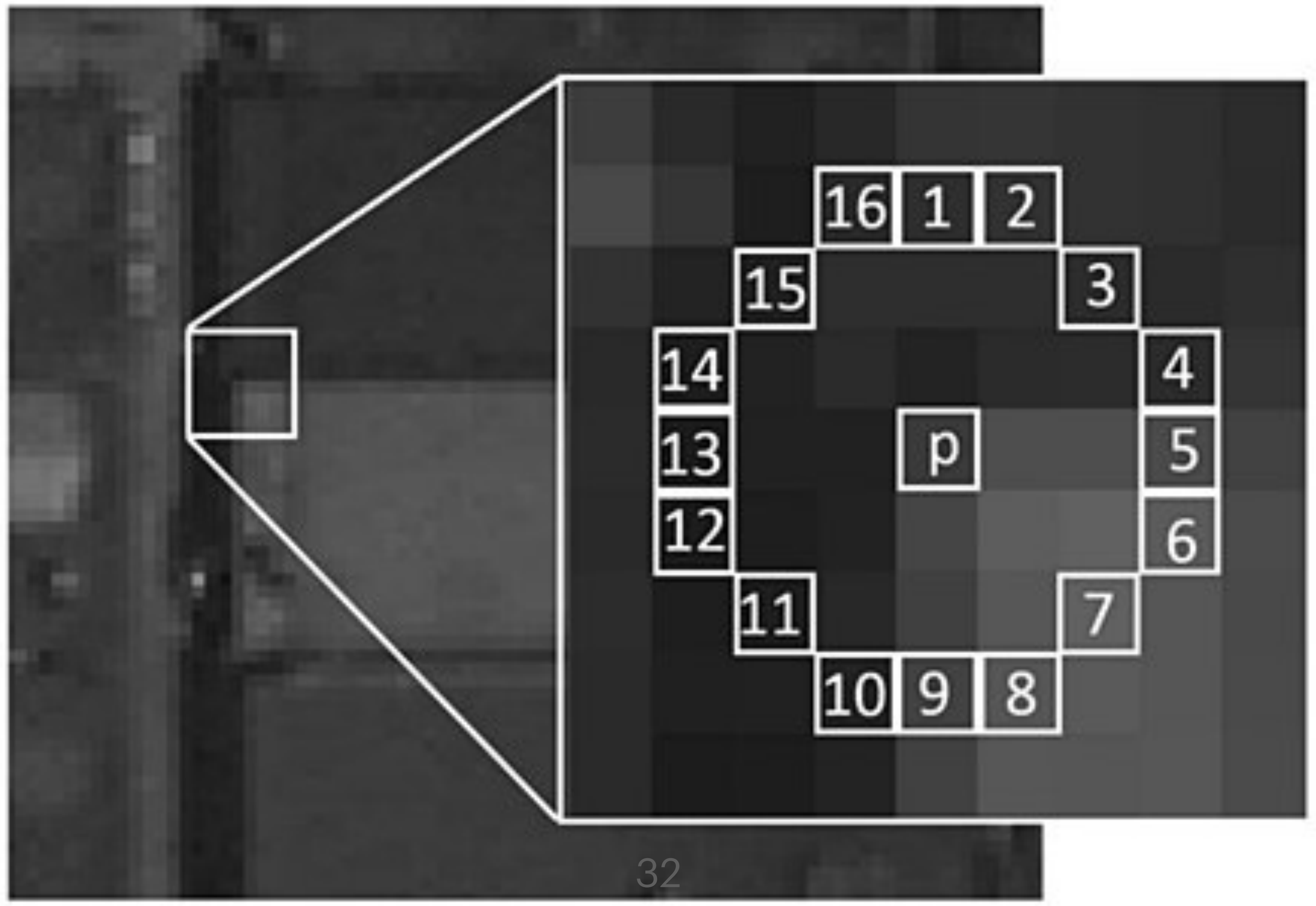
Alpha-Beta Velocity Model

- Low order approximation appropriate for simple systems
- Assuming constant velocity over a small time interval
- $x_{k+1}^{-} = x_k^{-} + v_k \Delta t$
- $v_{k+1}^{-} = v_k$
- $x_{k+1}^{+} = x_{k+1}^{-} + \alpha(x_{k+1} - x_{k+1}^{-})$
- $v_{k+1}^{+} = v_{k+1}^{-} + \dfrac{\beta}{\Delta t}(x_{k+1} - x_{k+1}^{-})$
- Values for α and β are adjusted experimentally

FAST

Features from Accelerated Segment Test corner detection

- Suitable for real-time feature estimation because of computational efficiency
- Select 16 pixels around candidate **p**
- N (usually 12) contiguous pixels are either all brighter or all darker than **p**



Shi-Tomasi Score

Detection of Corners

- If λ_1 and λ_2 are eigenvalues of M, then $C(x) = \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$
- Shi-Tomasi proposed: $C(x) = \min(\lambda_1, \lambda_2)$

Robust Feature Point Extraction

Even $\det M(x) \neq 0$ does not guarantee robust estimates of velocity — the inverse of $M(x)$ may not be very stable if, for example, the determinant of M is very small. Thus locations with $\det M \neq 0$ are not always reliable features for tracking.

One of the classical feature detectors was developed by Moravec '80, Förstner '84, '87 and Harris & Stephens '88.

It is based on the structure tensor

more weight to the centre of the image (or certain region)

$$M(x) \equiv G_\sigma * \nabla I \nabla I^\top = \int G_\sigma(x - x') \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix} (x')\, dx',$$

where rather than simple summing over the window $W(x)$ we perform a summation weighted by a Gaussian G of width σ.

Harris and Stephens propose the following expression:

$$C(x) = \det(M) - \kappa \operatorname{trace}^2(M),$$

and select points for which $C(x) > \theta$ with a threshold $\theta > 0$.

Estimating Point Correspondence

Prof. Daniel Cremers

8- and 4-Point Algorithms

Eight Point Algorithm (Longuet-Higgins '81)

Given a set of $n = 8$ or more point pairs $\boldsymbol{x}_1^i, \boldsymbol{x}_2^i$:

- Compute an approximation of the essential matrix. Construct the matrix $\chi = (a^1, a^2, \dots, a^n)^\top$, where $\chi \in \mathbb{R}^{n\times 9}$ $a^i = \boldsymbol{x}_1^i \otimes \boldsymbol{x}_2^i$. Find the vector $E^s \in \mathbb{R}^9$ which minimizes $\|\chi E^s\|$ as the ninth column of V_χ in the SVD $\chi = U_\chi \Sigma_\chi V_\chi^\top$. Unstack E^s into 3×3-matrix E.
  $[U_\chi, D_\chi, V_\chi] = \text{svd}(\chi)$, $E = (V_\chi(:,9),(3,3))$ → project onto essential space
- Project onto essential space. Compute the SVD $E = U \operatorname{diag}\{\sigma_1, \sigma_2, \sigma_3\} V^\top$. Since in the reconstruction, E is only defined up to a scalar, we project E onto the normalized essential space by replacing the singular values $\sigma_1, \sigma_2, \sigma_3$ with $1, 1, 0$.
- Recover the displacement from the essential matrix. The four possible solutions for rotation and translation are:

$$R = U R_Z^\top(\pm\tfrac{\pi}{2}) V^\top, \quad \widehat{T} = U R_Z(\pm\tfrac{\pi}{2}) \Sigma U^\top,$$

with a rotation by $\pm\frac{\pi}{2}$ around z:

$$R_Z^\top(\pm\tfrac{\pi}{2}) = \begin{pmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The Four Point Algorithm

Let us now assume we have $n \geq 4$ pairs of corresponding 2D points $\{\boldsymbol{x}_1^j, \boldsymbol{x}_2^j\}$, $j = 1, \dots, n$ in the two images. Each point pair induces a matrix $\boldsymbol{a}^j$, we integrate these into a larger matrix

$$\chi \equiv (\boldsymbol{a}^1, \dots, \boldsymbol{a}^n)^\top \in \mathbb{R}^{3n\times 9},$$

and obtain the system

$$\chi H^s = 0.$$

As in the case of the essential matrix, the homography matrix can be estimated up to a scale factor.

This gives rise to the four point algorithm:

- For the point pairs, compute the matrix χ.
- Compute a solution H^s for the above equation by singular value decomposition of χ.
- Extract the motion parameters from the homography matrix $H = R + \frac{1}{d} T N^\top$.

Reconstruction from Two Views: Linear Algorithms

Prof. Daniel Cremers

The Reconstruction Problem

The Epipolar Constraint

Eight-Point Algorithm

Structure Reconstruction

Four-Point Algorithm

The Uncalibrated Case

5-Point Algorithm

Rough Overview

$$\det(E) = 0$$

$$2EE^TE - \mathrm{tr}(EE^T)E = 0$$

$$\mathbf{m_1^T} F \mathbf{m_2} = 0$$

1. Epipolar constraint can be written as $\tilde{\mathbf{m}}^T \tilde{E} = 0$. Stacking $\tilde{\mathbf{m}}^T$ for all five points we obtain a 5x9 matrix. Compute four vectors, $\tilde{X}, \tilde{Y}, \tilde{Z}, \tilde{W}$, which span the null space of the matrix.
2. $E = xX + yY + zZ + wW$. Now find a solution for x, y, z by inserting the equation into the ten cubic constraints.
3. After determining E, recover R and t.
4. Use RANSAC to recover the best hypothesis.

Equipment Used

Unibrain Fire-i video camera
equipped with a 2.1mm wide-angle lens

Intel Core 2 Duo 2.66 GHz processor

Building a Map