Review of Dense 3D Surface Reconstruction

Alex Hagiopol

Outline

- Motivation
- Camera Calibration
- Stereo Vision
- Disparity Estimation
- Results

Motivation

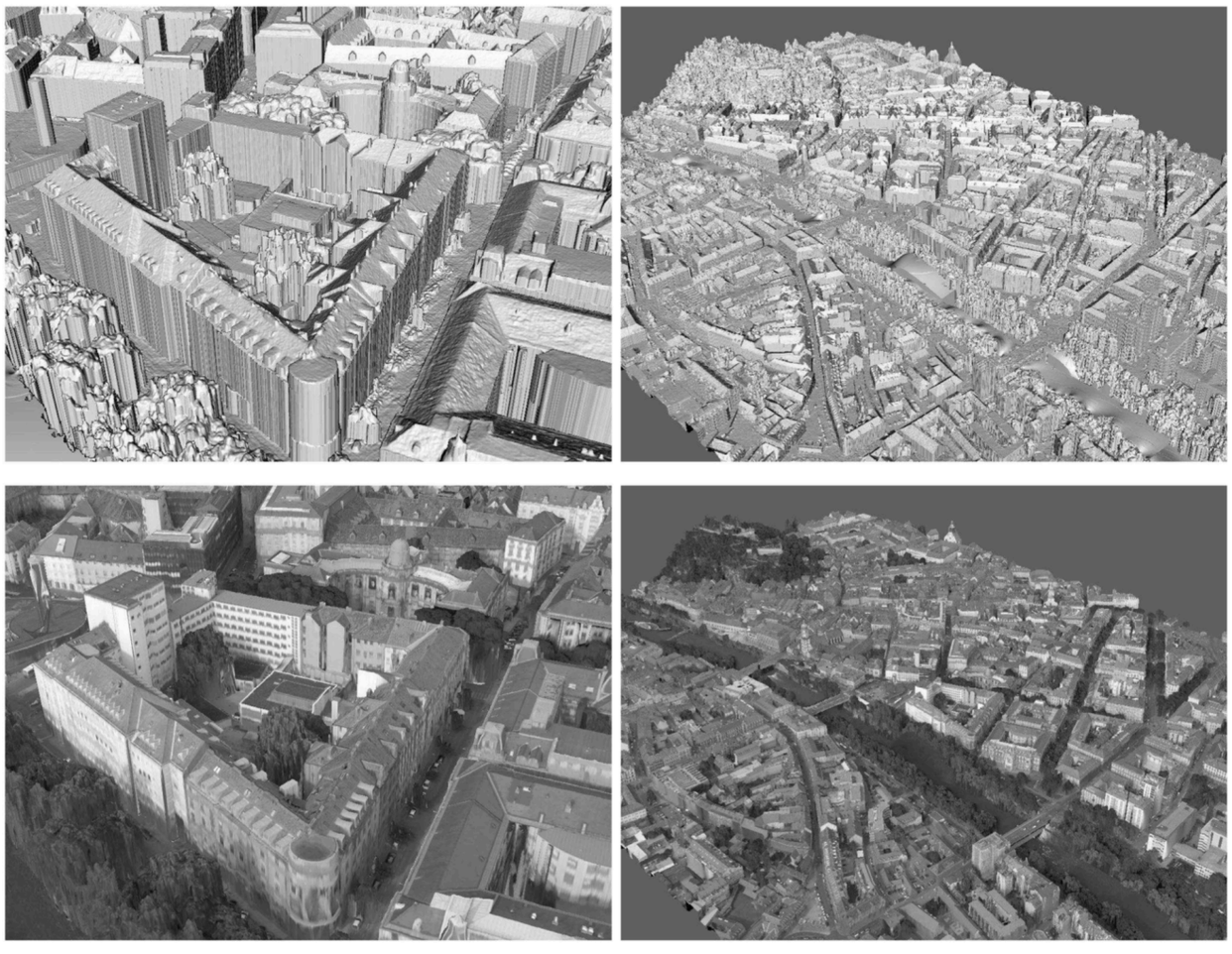
Motivation

- What can dense 3D surface reconstruction accomplish?

Motivation

- Mapping: Reconstruct large scale scenes for autonomous navigation, quality inspection, historical preservation.

City-scale 3D reconstruction of Graz, Austria using aerial images. [6]:

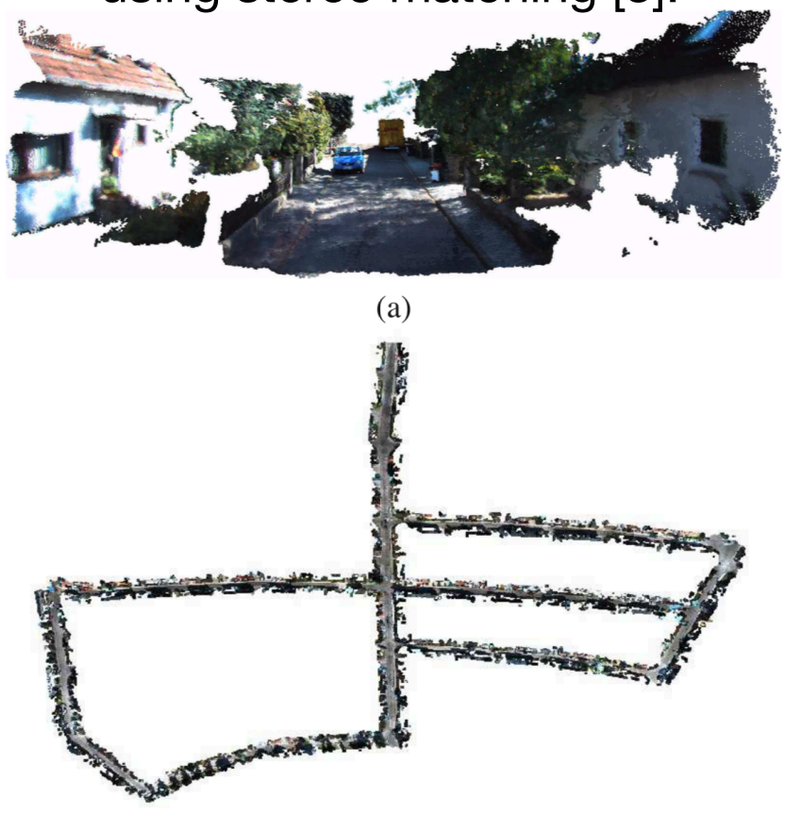


[5] H. Hirschmuller, ***Stereo Processing by Semiglobal Matching and Mutual Information***, (2008).

Motivation

- Mapping: Reconstruct large scale scenes for autonomous navigation, quality inspection, historical preservation.

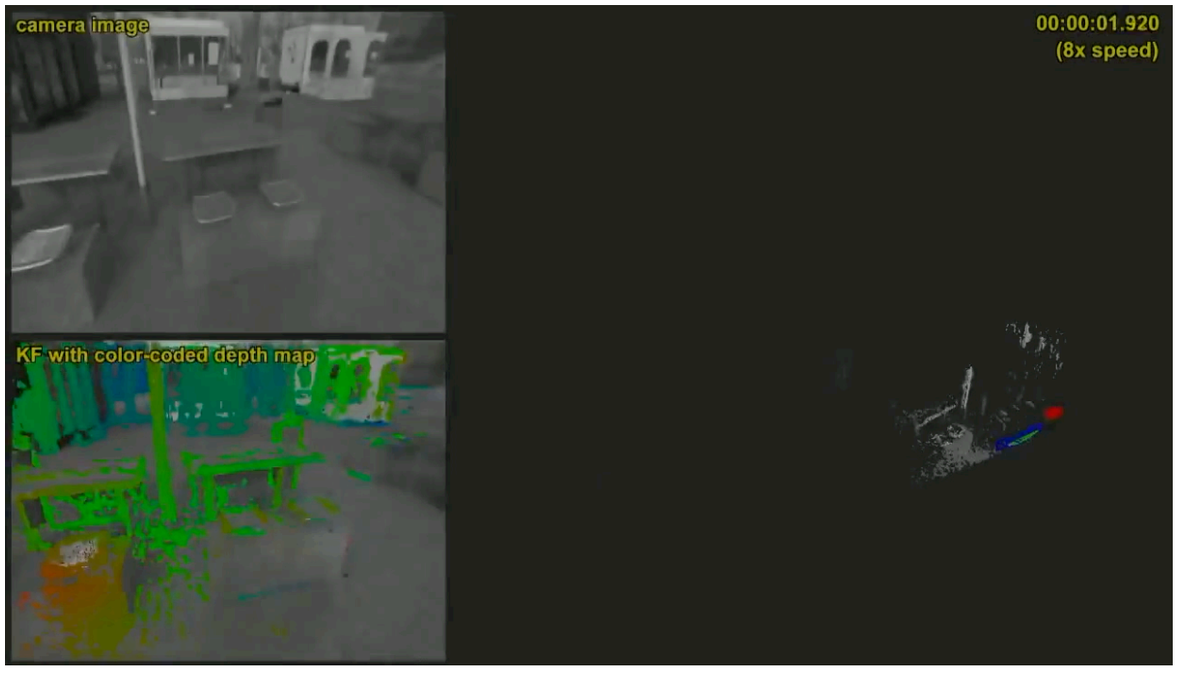
KITTI dataset point cloud reconstruction using stereo matching [3]:



(a)

(b)

Large-Scale Direct Monocular SLAM [4]:

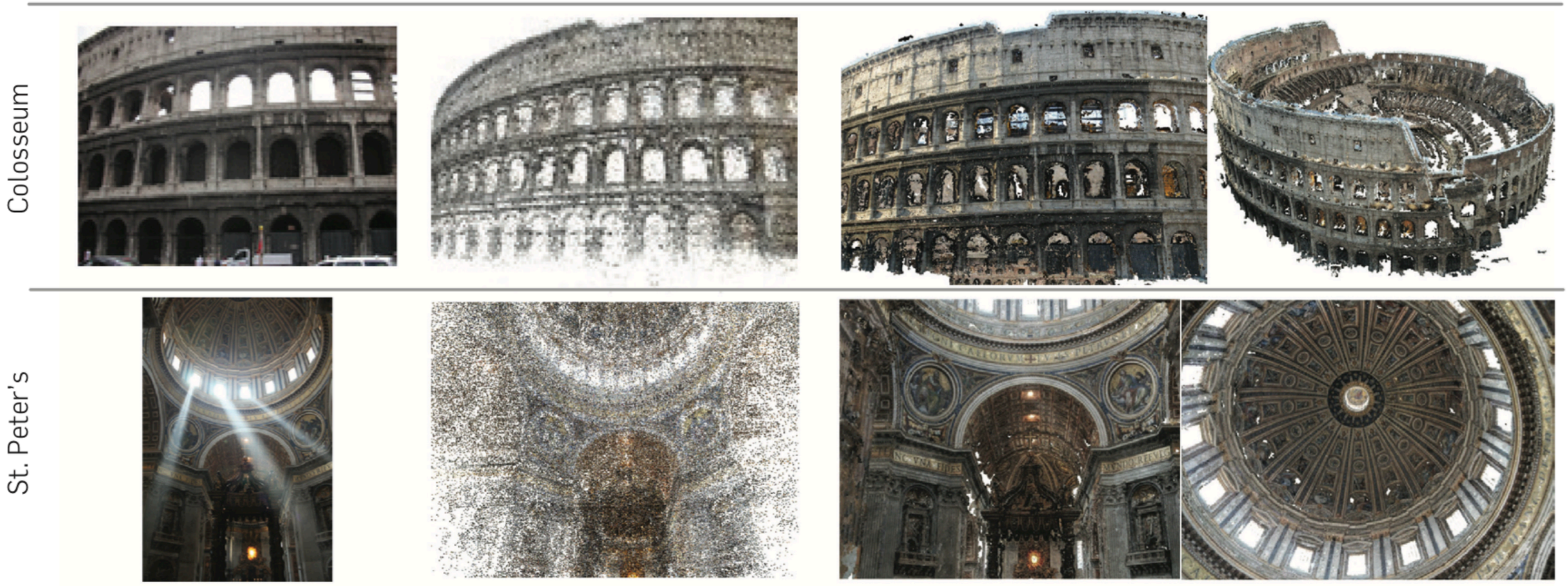


[3] P. Alcantarilla, C. Beall, and F. Dellaert, ***Large-Scale Dense 3D Reconstruction from Stereo Imagery,*** (2013).
[4] J. Engel, T. Schops, and D. Cremers, ***LSD-SLAM: Large-Scale Direct Monocular SLAM,*** (2014).

Motivation

- Mapping: Reconstruct large scale scenes for autonomous navigation, quality inspection, historical preservation.

City-scale 3D reconstruction using tourist photographs from Flickr. [7]:

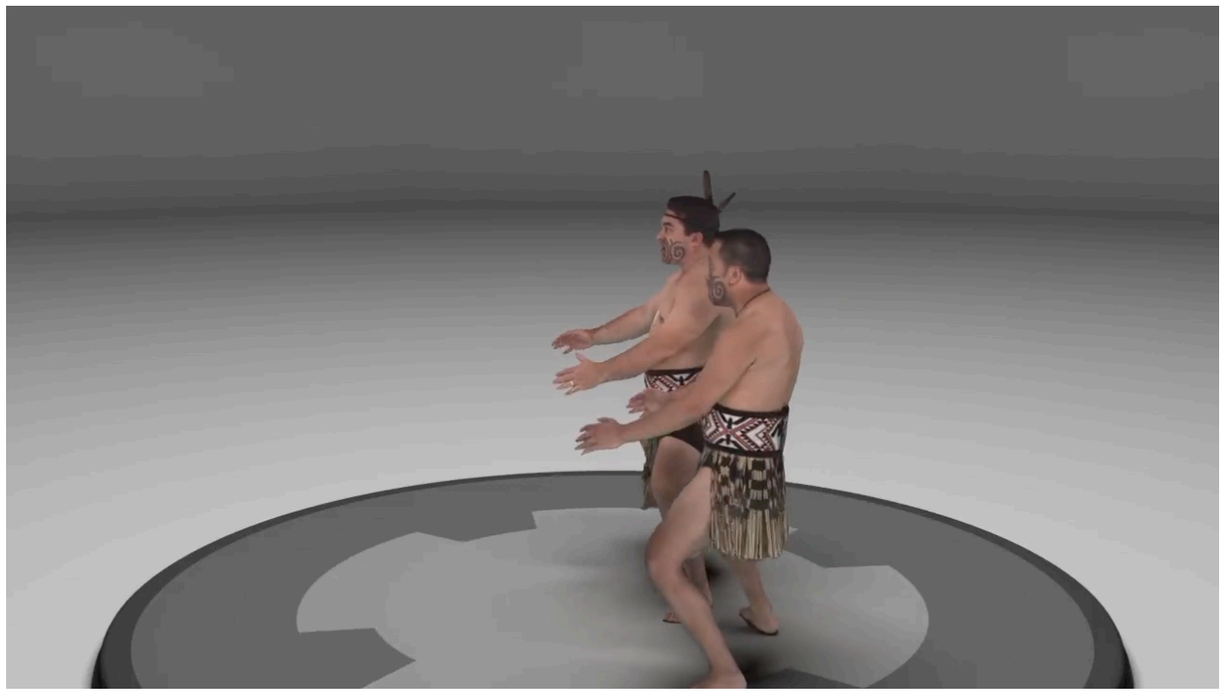


[7] S. Agarwal et al, ***Building Rome in a Day***, (2011).

Motivation

- Communication: Reconstruct human motion for high quality performance capture or real time interaction.

Offline human reconstruction [1]:



Real-time human reconstruction [2]:

[1] A. Collet et al, ***High Quality Streamable Free Viewpoint Video,*** (2015).
[2] M. Dou et al, ***Fusion4D: Real-Time Performance Capture of Challenging Scenes,*** (2016).

Motivation

- Goal: Reconstruct a collection of 3D points given 2D images from multiple views.
- Steps:

  1. Acquire images.
  2. Calibrate cameras (i.e. compute intrinsic parameters and extrinsic parameters).
  3. Rectify images
  4. Estimate disparities and triangulate points.

Image Acquisition

Image Acquisition

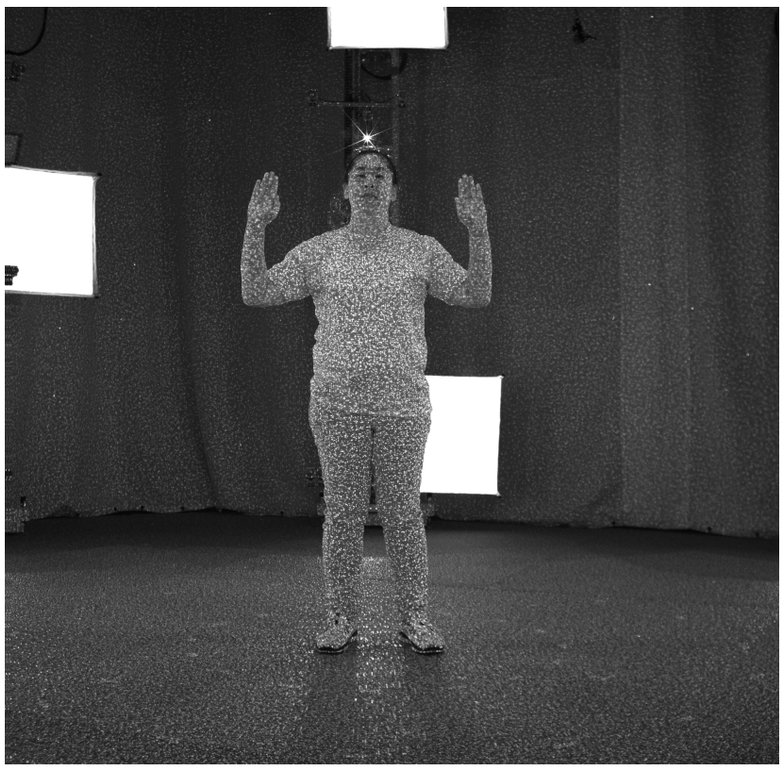
- Goal: Collect images for processing.
- Design issues:
  - Light spectrum?
  - Global vs rolling shutter?
  - Resolution (sampling density vs processing time)?
  - Frame rate?
  - Moving camera? Static camera? Static scene? Moving scene?

Image Acquisition

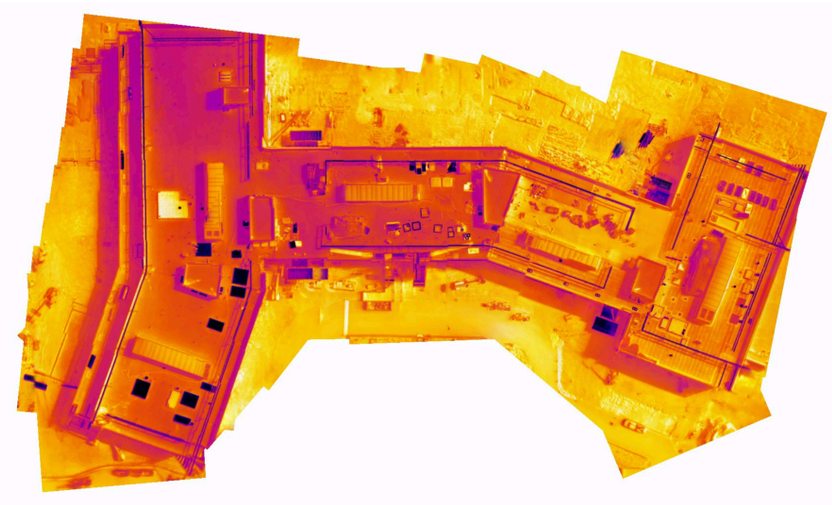
- Good results have been accomplished in the color spectrum, infrared spectrum, and thermal spectrum.

Color image acquired from the system in [1]:

Infrared image acquired from the system in [1]. Note laser dot texture projection:



View of dense map formed using thermal aerial imagery [8]:



[1] A. Collet et al, ***High Quality Streamable Free Viewpoint Video,*** 2015.
[9] ***DroneDeploy Thermal Reconstruction Documentation*** (link).

Image Acquisition

- Good results have been accomplished in the color spectrum, infrared spectrum, and thermal spectrum.

Color spectrum camera [7]:

Hologram application described in [1]:

Autonomous driving application described in [8]:

[1] A. Collet et al, ***High Quality Streamable Free Viewpoint Video,*** 2015.
[7] ***Matrix Vision mvBlueFox Technical Manual*** (link).
[8] ***Tesla Autopilot Demonstration*** (link).

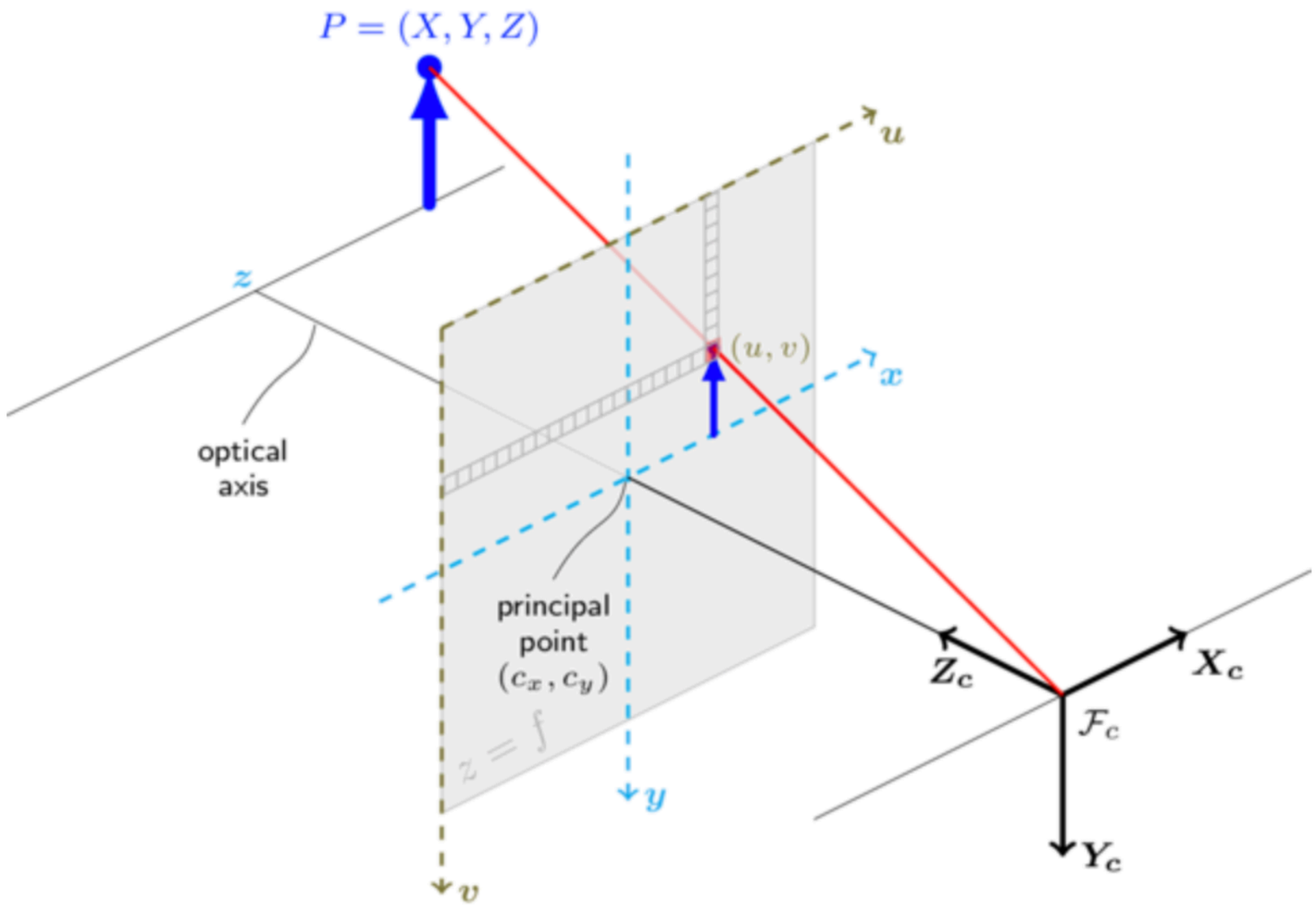
Camera Calibration

Camera Calibration

- Goal: Estimate camera parameters that mathematically model physical camera properties.
  - Intrinsic parameters (focal length, principal point, tangential distortion values, barrel distortion values).
  - Extrinsic parameters (rotation and translation).
  - Essential for triangulating 3D surface points given 2D images.
  - Heavily studied problem in computer vision and photogrammetry for past 20+ years.

Camera Calibration

- Pinhole camera model [10]:



[10] OpenCV 2.4.13.7 Documentation: ***Camera Calibration and 3D Reconstruction*** (link).

Camera Calibration

- Camera parameters [10]:

$$s\, m' = A[R|t]M'$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

(X, Y, Z) are the coordinates of a 3D point in the world coordinate space
(u, v) are the coordinates of the projection point in pixels
(cx, cy) is a principal point that is usually at the image center
fx, fy are the focal lengths expressed in pixel units.
The joint rotation-translation matrix $[R|t]$ is called a matrix of extrinsic parameters.

[10] OpenCV 2.4.13.7 Documentation: ***Camera Calibration and 3D Reconstruction*** (link).

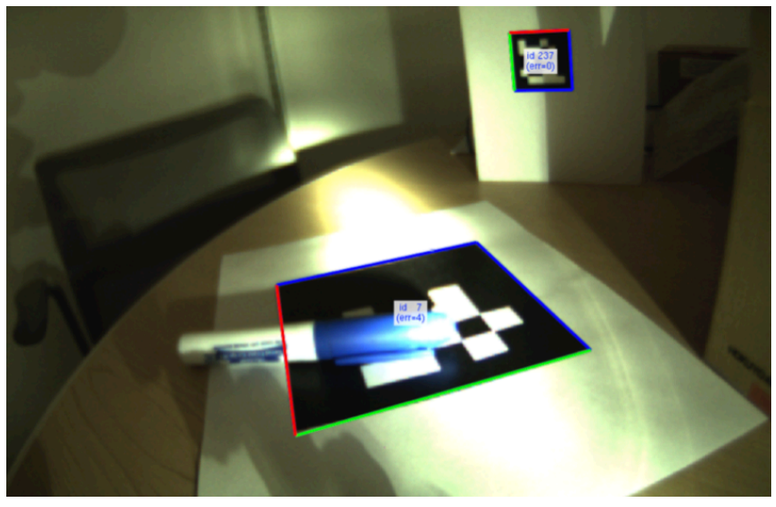
Camera Calibration

- Approaches:
  - **Marker-based**:
    - Compute camera parameters using images of calibration targets of known geometry.
    - Practical in laboratory or factory calibration scenarios where the highest accuracy is needed.
  - **Marker-less**:
    - Compute camera parameters using correspondences among images of unstructured scenes.
    - Practical in the wild.
- Marker-based can be used to estimate intrinsics in the lab. Those intrinsics can be used as input to marker-less methods for reconstruction in the wild.

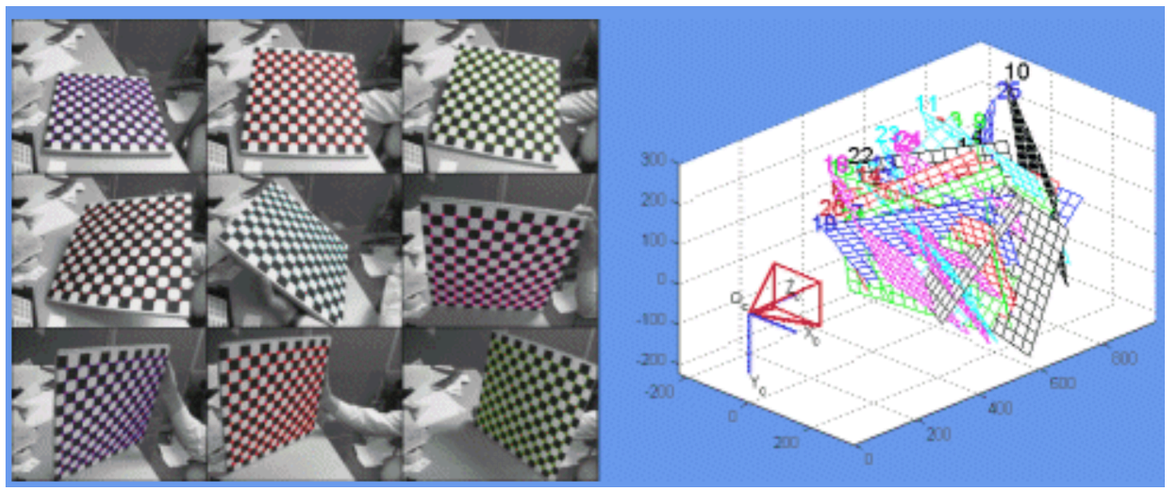
Camera Calibration

- Approaches:
  - **Marker-based**:
    - AprilTags [11] estimates 6DOF transformation between marker and camera from a single view.
    - Bouguet Calibration Toolbox [12] estimates intrinsic and extrinsic parameters from multiple views of a marker.

AprilTag detection and pose estimation from [11]:



Marker detection and pose estimation from [12]:



[11] Edwin Olson, ***AprilTag: A Robust and Flexible Visual Fiducial System,*** 2011.
[12] J. Bouguet, ***Camera Calibration Toolbox for MATLAB****,* 1999 (link).

Camera Calibration

- Approaches:
  - **Marker-less**:
    - **Structure From Motion** [13, 14]
      - Estimates camera parameters from a collection of unordered images, usually in an offline manner using correspondences.
      - Uses optimization [15] and invariant feature extraction [16].
    - **SLAM and VO** [4, 17, 18]
      - Estimate camera poses online from sequential images. Optimization and feature extraction [19, 20] are applied as well. Loop closure [21] is a key element in SLAM. Correspondences are also used.

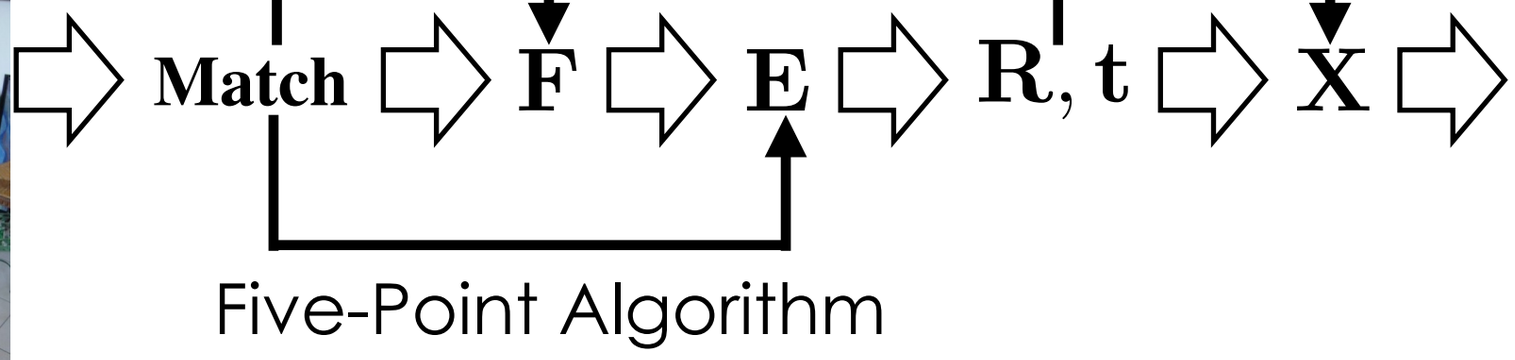
[13] B. Triggs et al, ***Bundle Adjustment - A Modern Synthesis,*** 2000.
[14] R. Hartley and A. Zisserman, ***Multiple View Geometry in Computer Vision,*** 2004.
[15] A. Ranganathan, ***The Levenberg-Marquardt Algorithm,*** 2004.
[16] D. Lowe, ***Distinctive Image Features from Scale-Invariant Keypoints,*** 2004.
[17] R. Mur-Artal et al, ***ORB-SLAM: a Versatile and Accurate Monocular SLAM System***, 2015.
[18] C. Forster et al, ***SVO: Fast Semi-Direct Monocular Visual Odometry***, 2014.
[19] E. Rublee et al, ***ORB: An efficient alternative to SIFT or SURF***, 2011.
[20] E. Rosten and T. Drummond, ***Machine Learning for High-Speed Corner Detection***, 2006.
[21] R. Mur-Artal and J. Tardos, ***Fast Relocalization and Loop Closing in Keyframe-Based SLAM***, 2014.

Numerical descriptors of extracted features must be unique enough for features to be re-identified as the camera moves.
Features on the left video frame are matched with their corresponding features on the right video frame. Parallel lines indicate correct matches. Intersecting lines indicate mistaken matches.
RANSAC is often used to improve match quality.

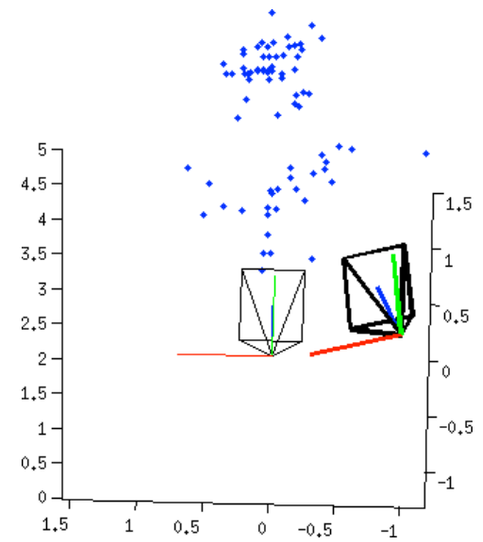
Camera Calibration

Eight-Point Algorithm

Triangulation



Five-Point Algorithm

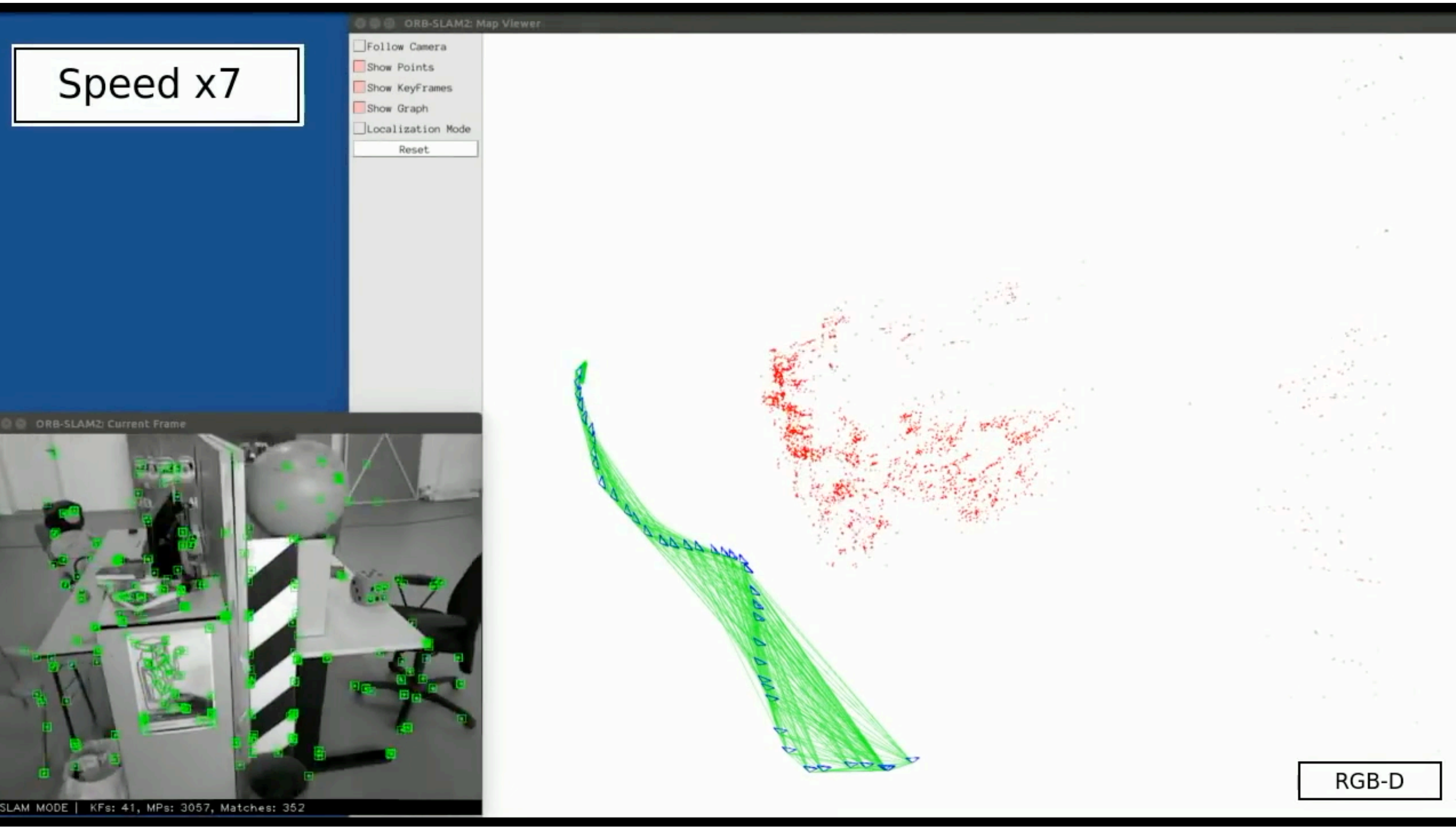


Bundle Adjustment

- Structure from motion process:
  1. Estimate intrinsics matrix **K** using marker-based calibration.
  2. In the wild, detect features and descriptors using SIFT [16].
  3. Estimate fundamental matrix **F** using RANSAC algorithm.
  4. Compute essential matrix **E** using relationship $\mathbf{E} = \mathbf{K}'^{\mathsf{T}}\mathbf{F}\mathbf{K}$
  5. Compute rotation and translation using relationship $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$
  6. Triangulate sparse 3D point cloud of SIFT features.
  7. Optimize poses: Compare reprojected features versus observations to construct error term. Apply an optimization technique (e.g. [15]) that minimizes error.

Slide adapted from [22] Fisher Yu, ***Multiple View 3D Reconstruction***, 2014 (link).

Camera Calibration



[17] R. Mur-Artal et al, ***ORB-SLAM: a Versatile and Accurate Monocular SLAM System***, 2015.

Stereo Vision

Stereo Vision

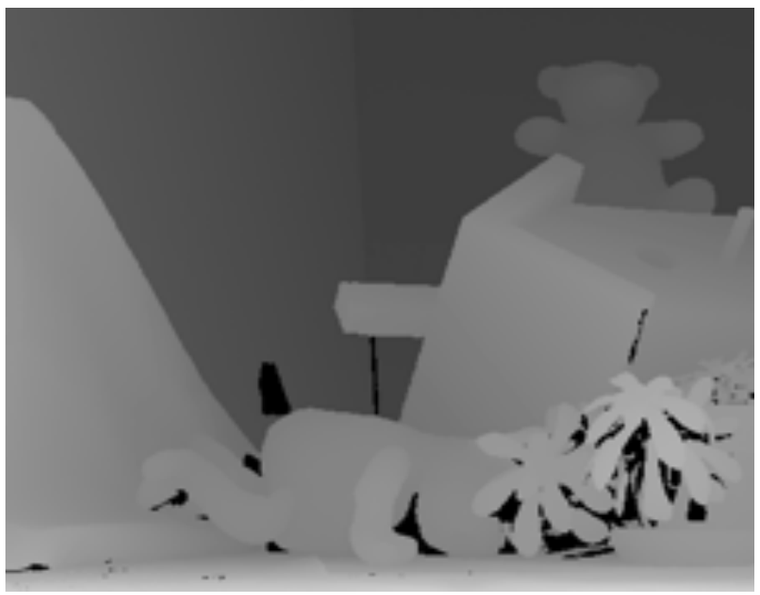
- Using previous work, we've acquired **images** and **camera parameters**. But how do we estimate a 3D surface?
- Goal: Triangulate a **dense point cloud** using **images** and **camera parameters**.

Disparity Estimation

Left 2D Image

Right 2D Image

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$



Disparity Map

$$Z = \frac{fT}{d}$$

3D View

Slide credit: [23] M. Bleyer, ***Fundamentals of Stereo Vision***.

Stereo Vision

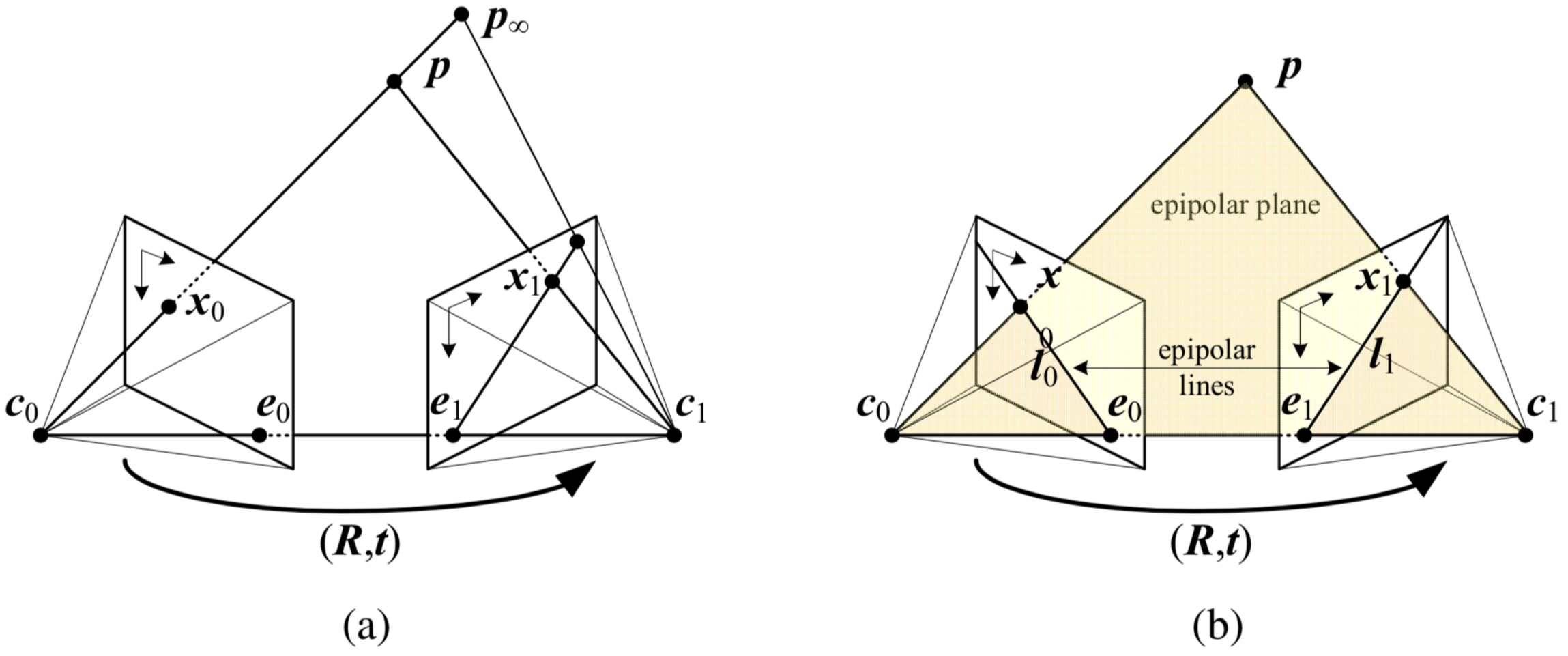
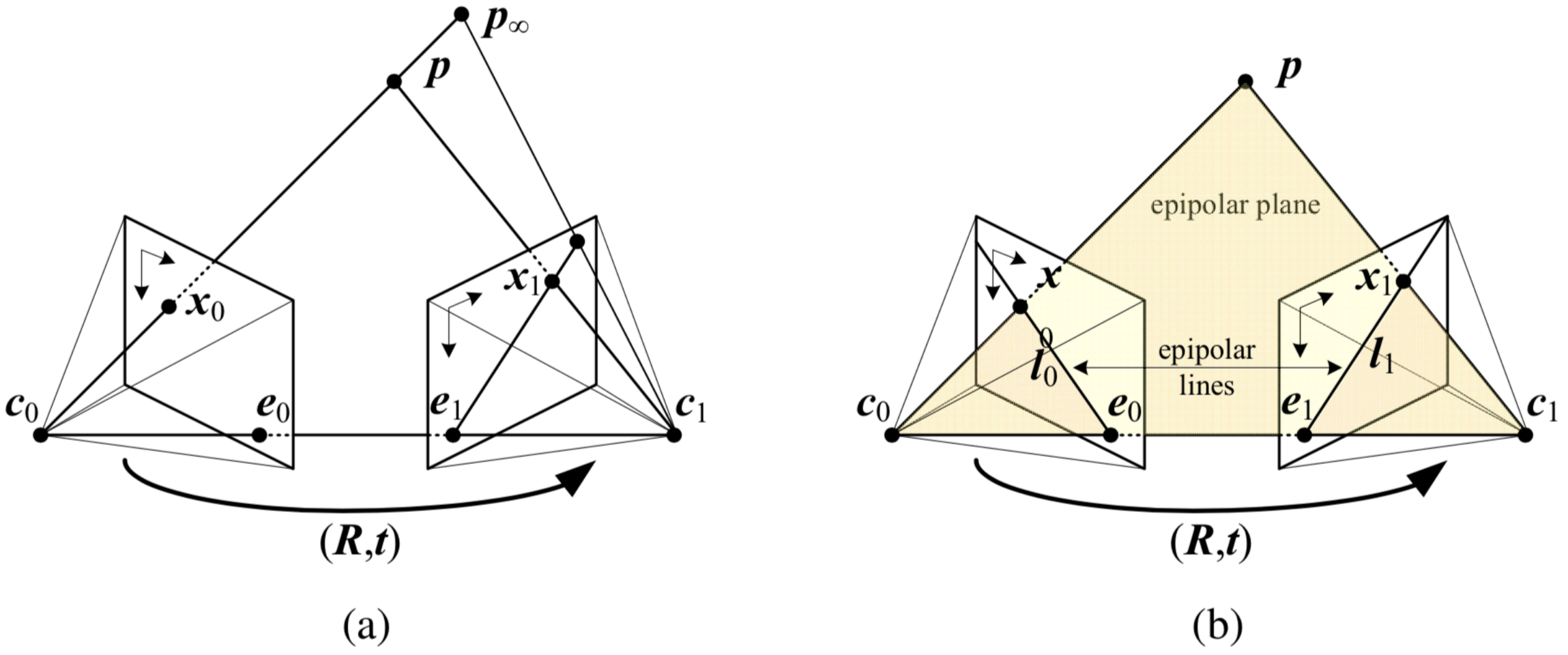


Figure 11.3 Epipolar geometry: (a) epipolar line segment corresponding to one ray; (b) corresponding set of epipolar lines and their epipolar plane.

[24] R. Szeliski, ***Computer Vision Algorithms and Applications***, 2010.

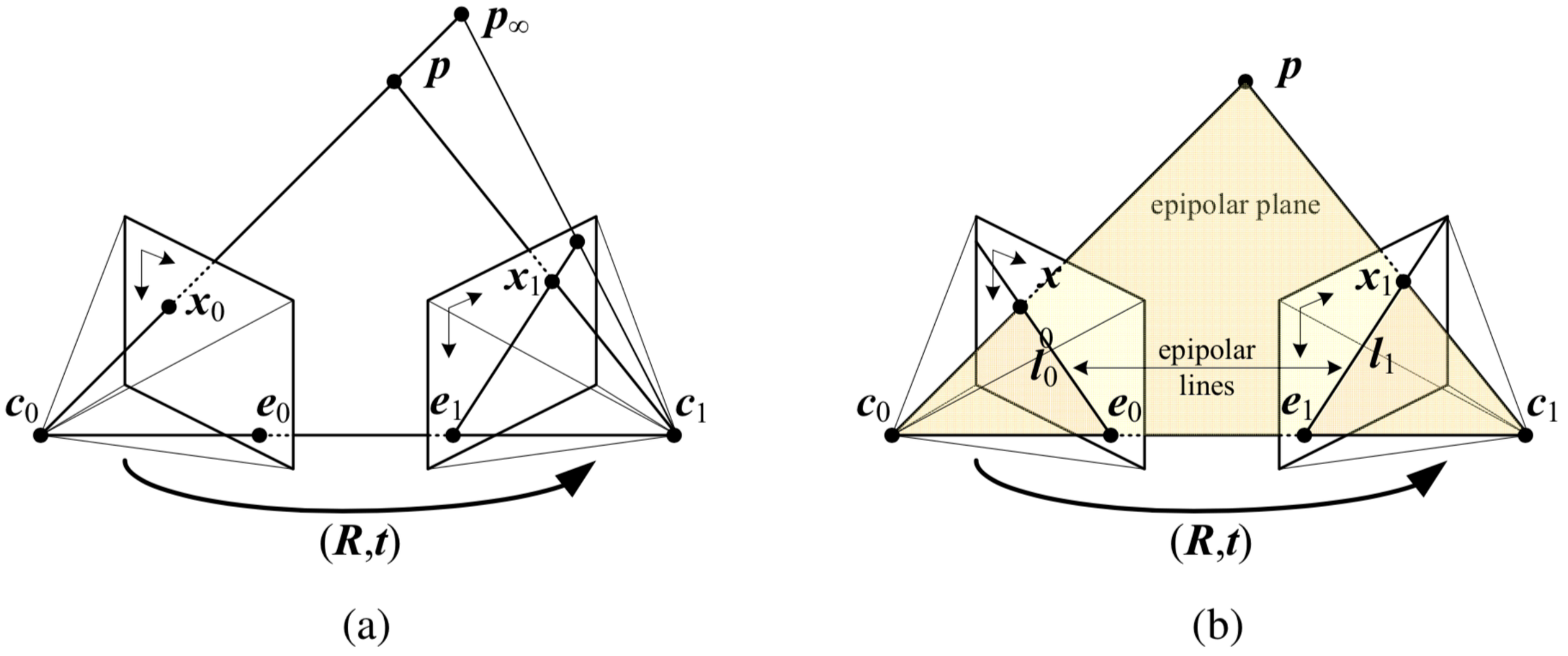
Stereo Vision



- If camera parameters are known, and 2D locations x_0 and x_1 are known, then 3D point p can be triangulated.
- If we do a trinagulation for every pixel pair match, we generate millions of 3D points from just two views. Thus, a dense surface.

[24] R. Szeliski, ***Computer Vision Algorithms and Applications***, 2010.

Stereo Vision



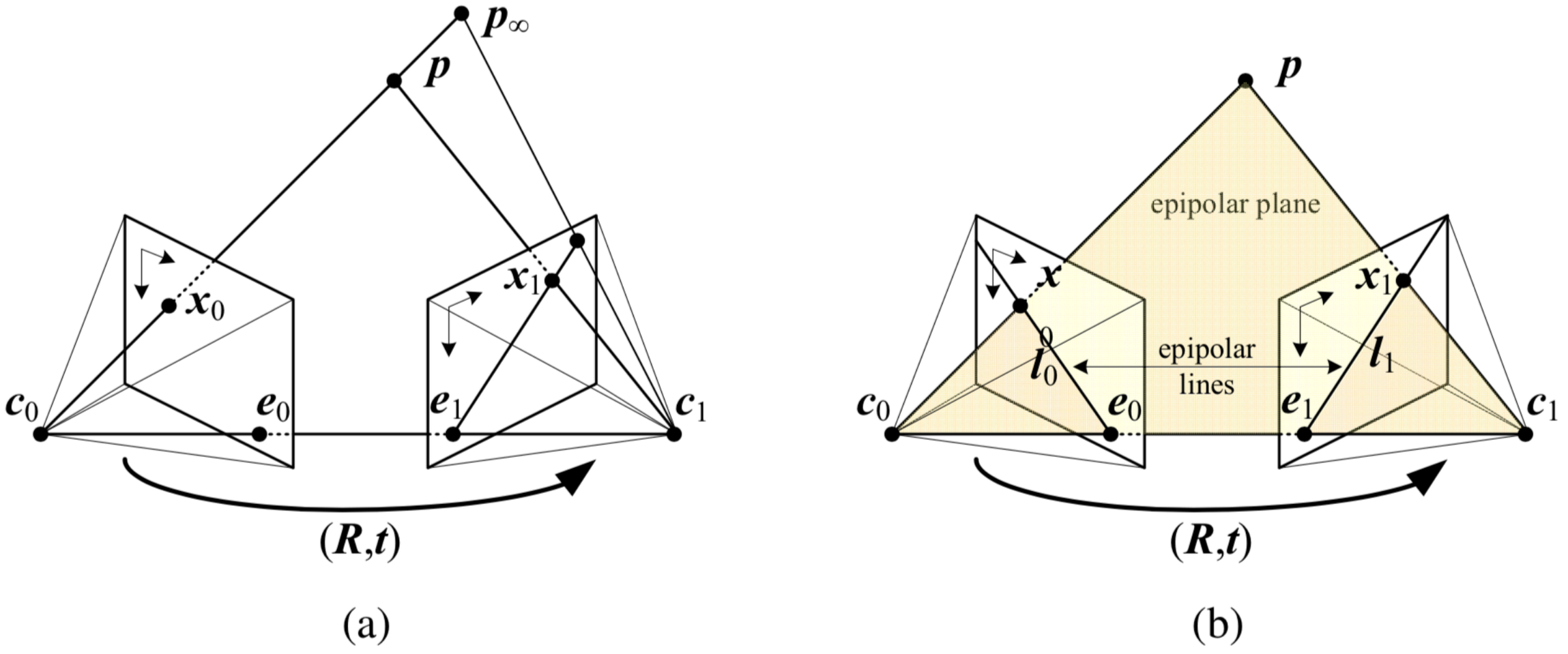
- Dense reconstruction is therefore solved if we calculate matching relationships between left and right pixels.
- The idea is to **search** for x_1 given x_0 and previous parameters. Dense reconstruction becomes a search problem.

[24] R. Szeliski, ***Computer Vision Algorithms and Applications***, 2010.

Stereo Vision

- In an N*N image, there are O(N^2) pixels in the search space.

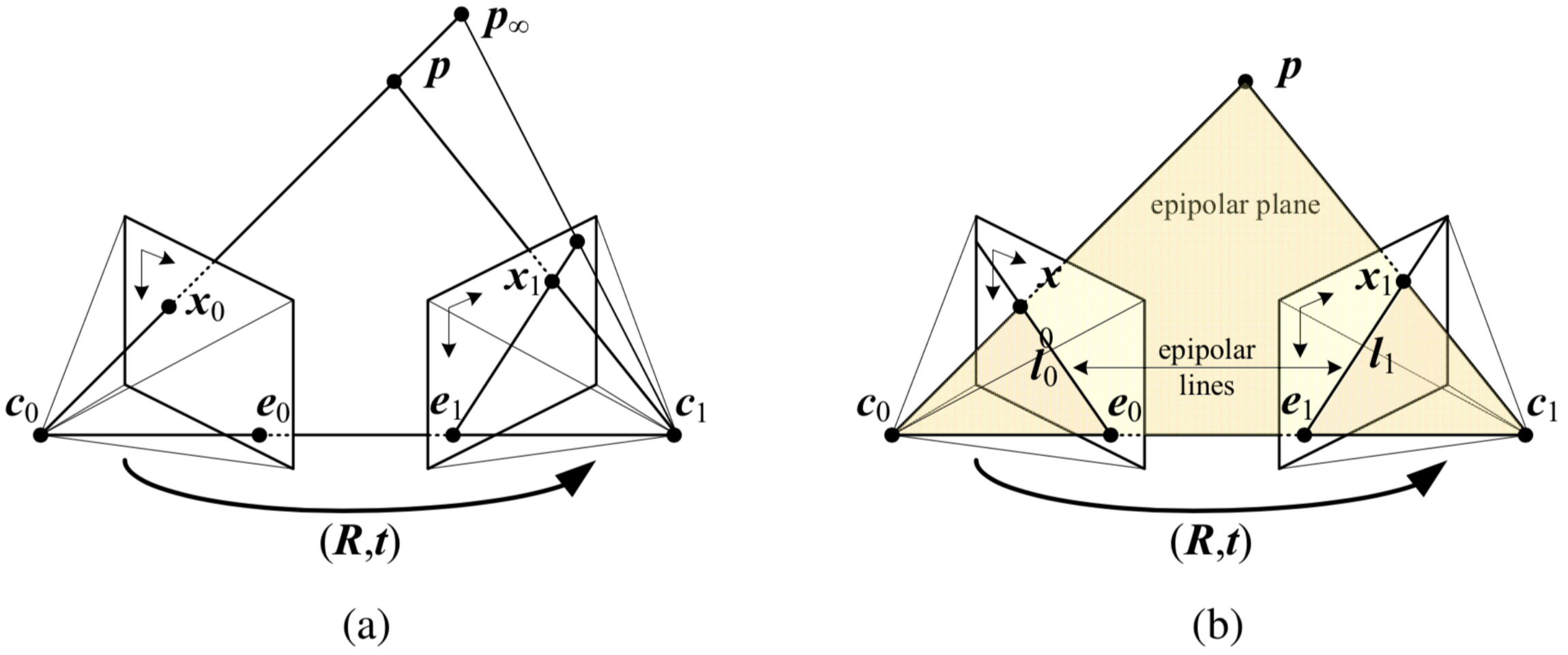
Stereo Vision



- Ray from c_0 to x_0 and ray from c_0 to c_1 define an epipolar plane. This plane intersects with the right image plane to define a right epipolar line.

[24] R. Szeliski, ***Computer Vision Algorithms and Applications***, 2010.

Stereo Vision



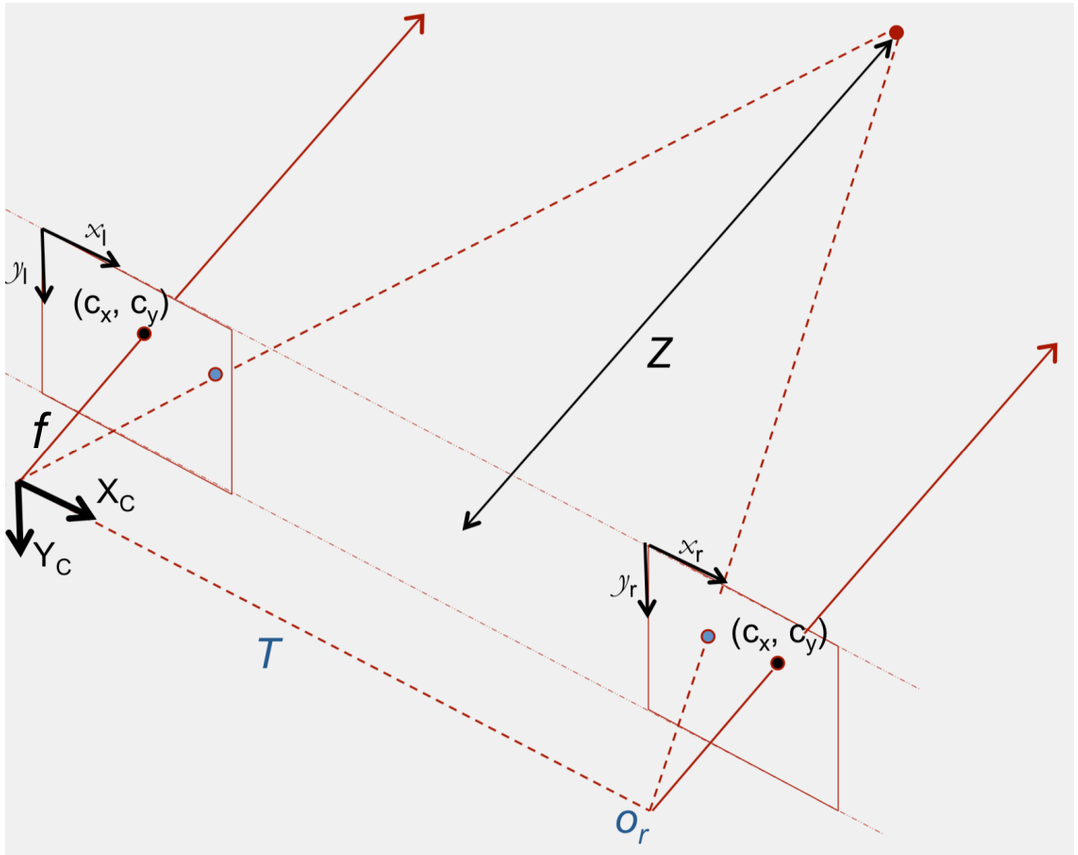
- We observe that given c_0, x_0, and c_1, the search for x_1 is therefore constrained: x_1 must lie on the right epipolar line.
- Search space is now O(N) instead of $O(N^2)$!

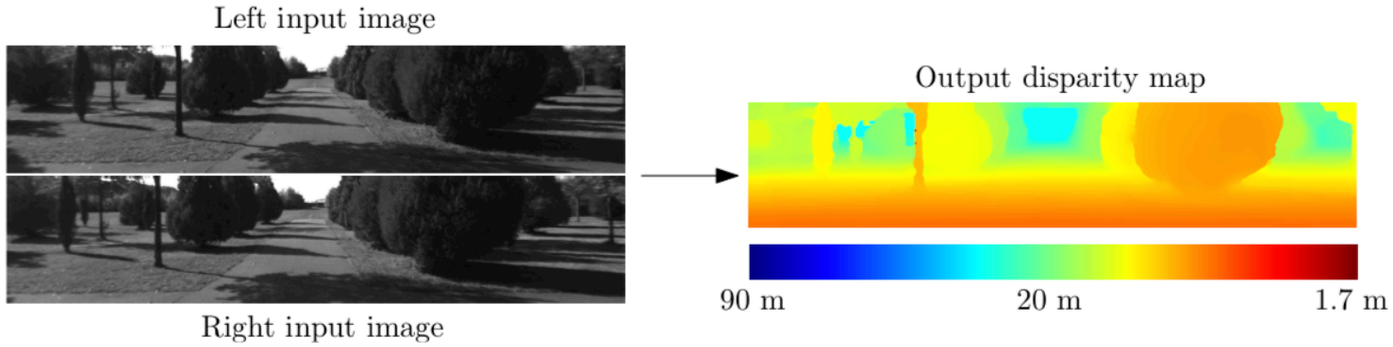
[24] R. Szeliski, ***Computer Vision Algorithms and Applications***, 2010.

Stereo Vision

- Any other optimizations?

Stereo Vision



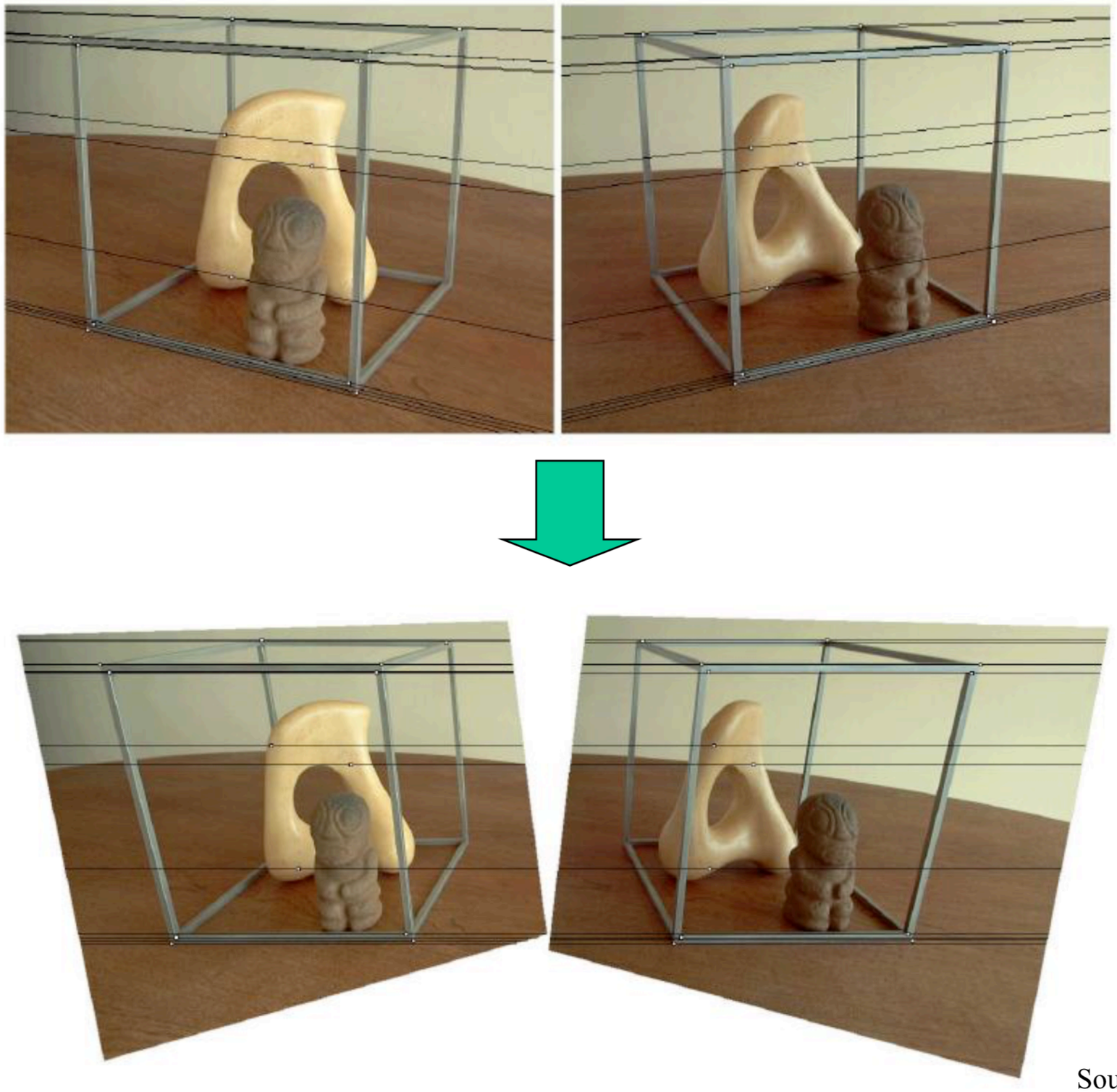


$$Z = \frac{fT}{d}$$

- If the image planes are coplanar and epipolar lines are parallel, search space is now a horizontal row of pixels.
- Greatly simplifies implementation. Search is reduced to calculating difference between left x-coordinate x_l and right x-coordinate x_r i.e. the **disparity, *d***.
- Y-coordinates of matching points are equal.

[25] D.J. Lee, ***Stereo Calibration and Rectification***, 2012.

Stereo Vision



- Rotate both cameras so that they are looking perpendicular to the line joining the camera centers c_0 and c_1.
- Next, to determine the desired twist around the optical axes, make the up vector (the camera y axis) perpendicular to the camera center line.
- Re-scale the images to account for different focal lengths, magnifying the smaller image to avoid aliasing.

Source: Alyosha Efros

[24] R. Szeliski, ***Computer Vision Algorithms and Applications***, 2010.

Disparity Estimation

Disparity Estimation

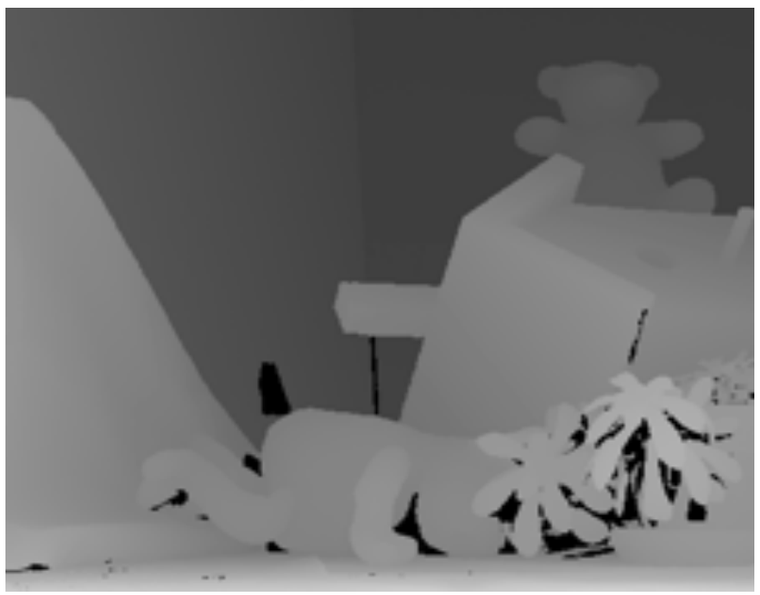
- Goal: For each pixel in a rectified reference image, compute the pixel in another rectified image that samples the exact same point in space.

Disparity Estimation

Left 2D Image

Right 2D Image

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}$$



Disparity Map

$$Z = \frac{fT}{d}$$

3D View

Slide credit: [23] M. Bleyer, ***Fundamentals of Stereo Vision***.

Disparity Estimation

- The PatchMatch Stereo [26] algorithm for disparity estimation:
  - Good results: industry standard in hologram capture.
  - Compact implementation.
  - Readily parallelizeable.

[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.

Disparity Estimation

- Other Choices:
  - Semi-Global Matching [5]
    - Also popular.
    - Accurate, but slow.
  - Deep learning based methods [27, 28, 29]:
    - Very promising, lots of progress.
    - Difficult to create ground truth dataset.
    - Difficult to generalize results from training dataset.

[5] H. Hirschmuller, ***Stereo Processing by Semiglobal Matching and Mutual Information***, 2008.
[27] J. Zbontar and Y. LeCun, ***Stereo Matching by Training a CNN to Compare Image Patches***, 2016.
[28] J.R. Chang and Y.S. Chen, ***Pyramid Stereo Matching Network***, 2018.
[29] S. Duggal et al, ***DeepPruner: Learning Efficient Stereo Matching via Differentiable PatchMatch,*** 2019.

Disparity Estimation

- PatchMatch Stereo algorithm model:
  - At each pixel p, store the parameters $a_p, b_p, c_p, n_{p_1}, n_{p_2}, n_{p_3}$ of a disparity plane.
  - Accounts for matching cost calculation on non-fronto-parallel surfaces.
    - Query disparity at any location according to plane parameters:
    - $d_p = a_p x + b_p y + c_p$
    - Where d_p is the disparity at position (x, y) based on the plane parameters assigned to that pixel.

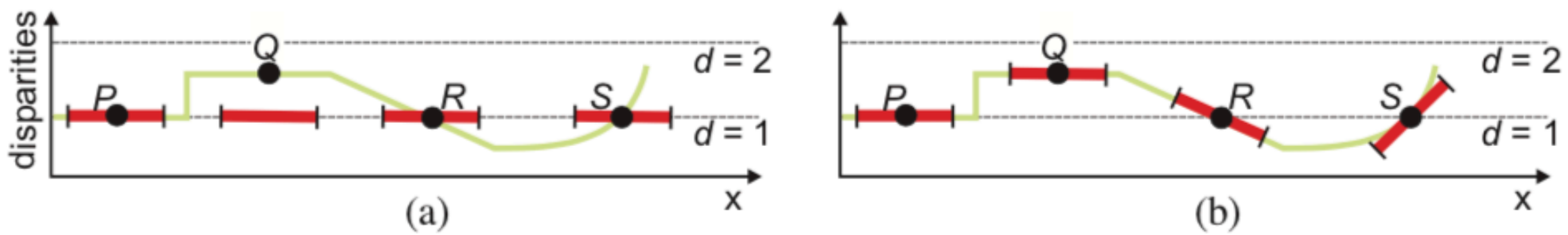


Figure 1: Support Regions (in 1D). The points of the green surface shall be reconstructed. Support regions are shown by red bars. (a) Fronto-parallel windows at integer disparities as used in standard methods. (b) Our support regions. We estimate a 3D plane at each point.

[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.

Disparity Estimation

- PatchMatch Stereo algorithm model:
  - $d_p = a_p x + b_p y + c_p$
  - This formulation allows for the plane parameters of one pixel at (x_i, y_j) to be used to compute a disparity for its neighbors (x_{i-1}, y_j), (x_i, y_{j-1}), (x_{i+1}, y_j), $(x_i, y_{j+1})\ldots$

1. Accounts for non-fronto-parallel surface properties.
2. Enables disparity plane parameters propagation from one pixel to its neighbors.

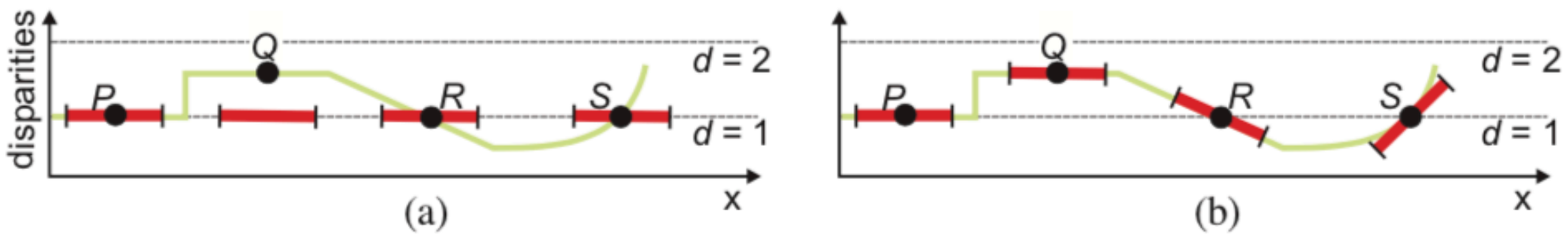


Figure 1: Support Regions (in 1D). The points of the green surface shall be reconstructed. Support regions are shown by red bars. (a) Fronto-parallel windows at integer disparities as used in standard methods. (b) Our support regions. We estimate a 3D plane at each point.

[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.

Disparity Estimation

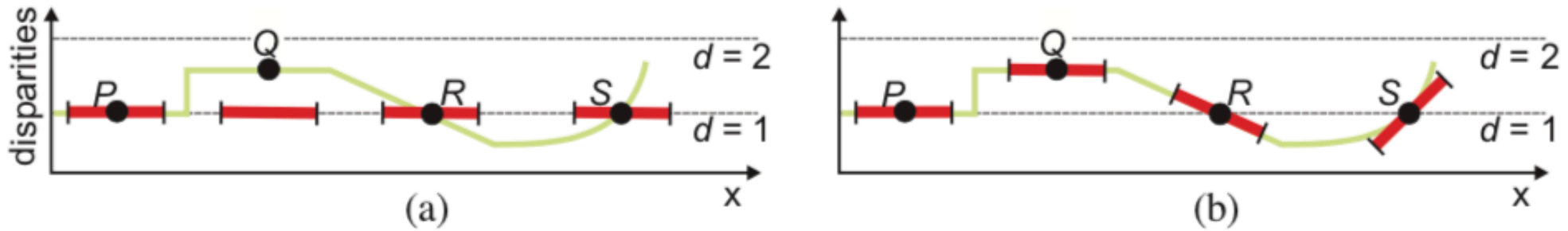


Figure 1: Support Regions (in 1D). The points of the green surface shall be reconstructed. Support regions are shown by red bars. (a) Fronto-parallel windows at integer disparities as used in standard methods. (b) Our support regions. We estimate a 3D plane at each point.

Figure 2: Advantage of our method. (a) Left image and ground truth disparities of the Corridor pair. (b) The disparity map computed with fronto-parallel windows approximates the slanted surfaces via many fronto-parallel ones. (c) Our slanted support windows correctly reconstruct the scene as a collection of slanted planar surfaces.

[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.

Disparity Estimation

- PatchMatch Stereo algorithm procedure:
  - **Random Initialization**:
    - For each pixel, assign random plane parameters $a_p, b_p, c_p, n_{p_1}, n_{p_2}, n_{p_3}$.
    - Compute cost function according to Normalized Cross Correlation in a user-defined window size around each pixel.
    - Why randomness? Law of large numbers states that with so many guesses, there is a high likelihood that *some* are good.
  - The idea is to initialize randomly, and propagate to/from a neighbor if doing so improves the NCC cost. If not, attempt more random guesses.

For this reason, *normalized cross-correlation* is more commonly used,

$$E_{\mathrm{NCC}}(\boldsymbol{u}) = \frac{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]\,[I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]}{\sqrt{\sum_i [I_0(\boldsymbol{x}_i) - \overline{I_0}]^2}\sqrt{\sum_i [I_1(\boldsymbol{x}_i + \boldsymbol{u}) - \overline{I_1}]^2}}, \tag{8.11}$$

where

$$\overline{I_0} = \frac{1}{N}\sum_i I_0(\boldsymbol{x}_i) \quad \text{and} \tag{8.12}$$

$$\overline{I_1} = \frac{1}{N}\sum_i I_1(\boldsymbol{x}_i + \boldsymbol{u}) \tag{8.13}$$

[24] R. Szeliski, ***Computer Vision Algorithms and Applications***, 2010.
[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.

Disparity Estimation

- PatchMatch Stereo algorithm procedure:
  - **Propagation**:
    - Check if propagating plane parameters from one pixel to its neighbors will lower the cost value for each neighbor.
      - Spatial Propagation - Close neighbors in space will have similar planes.
      - View Propagation - A pixel and its match will have similar planes.
      - Temporal Propagation - The same pixel will have similar plane parameters at times t and $t + 1$.
  - **Iterative Refinement:**
    - After each propagation opportunity randomly perturb plane parameters for a set number of iterations. If the NCC cost improves, keep the new plane parameters.

[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.

Disparity Estimation

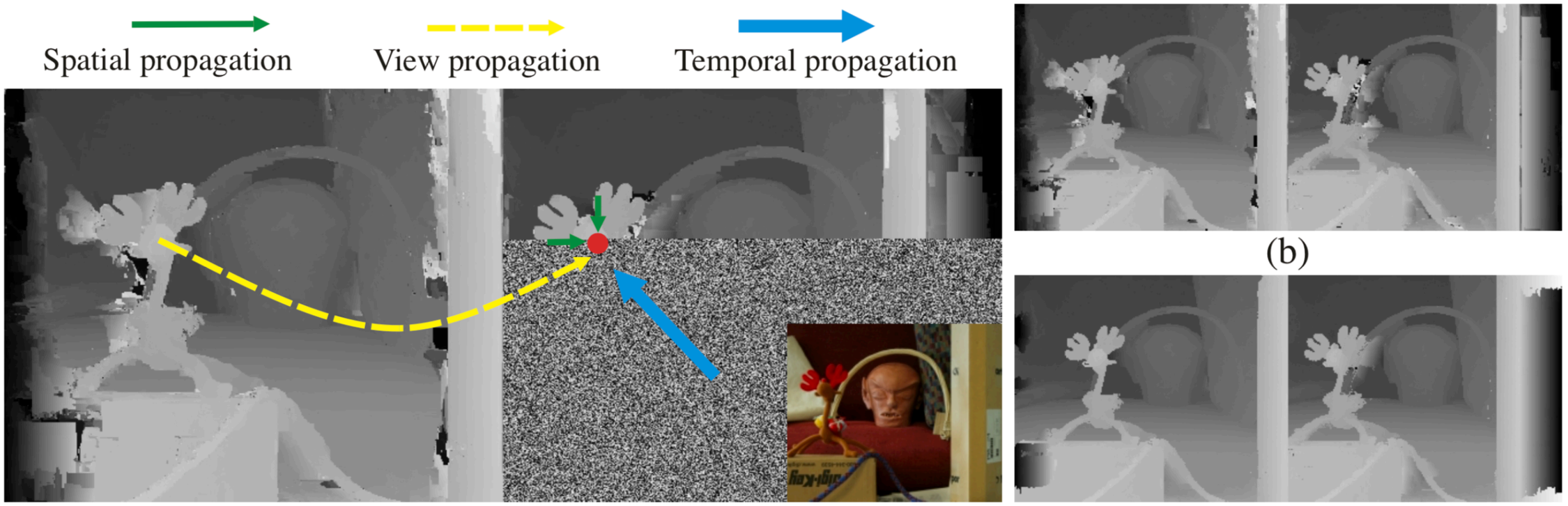


Figure 3: Different steps of the algorithm. (a) Left and right disparity maps at an intermediate step of the first iteration. All pixels up to the marked red one have already been processed. The other ones are still assigned to planes found in random initialization. We use three types of propagation illustrated by arrows. (b) Results after iteration 3. (c) Results after post-processing. We perform left/right consistency checking and occlusion filling.

[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.

Disparity Estimation

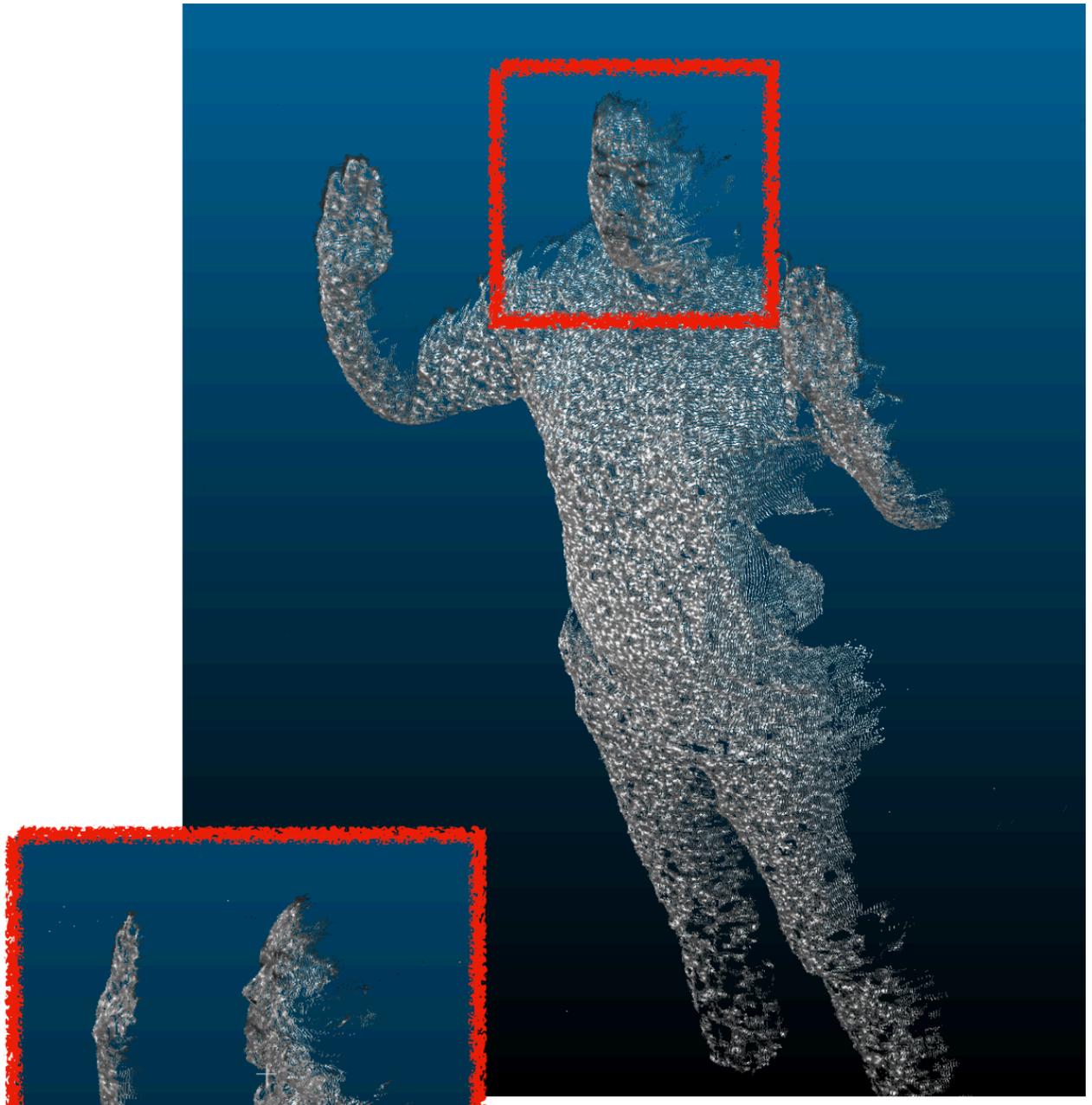
- PatchMatch Stereo algorithm implementation notes:
  - Canonical implementation propagates according to a raster pattern i.e. to pixel (x_i, y_j) we attempt to propagate from pixels (x_{i-1}, y_j), (x_{i-1}, y_{j-1}), (x_i, y_{j-1}), (x_{i+1}, y_{j-1}). This limits propagation to a single threaded procedure due to dependency relationships.
  - Parallel implementation propagates in 4 stages: right, down, left, up. In each stage, each pixel is only dependent on pixels whose coordinates differ in one direction. For example in the right propagation, each row of pixels can be processed independently. Thus an N*N image can be processed with N GPU threads.
  - View propagation is implemented using left-right consistency checking. I.e. we compute left image -> right image disparities then compute right image -> left image disparities. We then compare them and invalidate pixels that disagree. With horizontal image mirroring, this procedure enables major code reuse.
  - Areas of low texture can be handled better by multi-scale PatchMatch. I.e. we can downsample an image, compute its disparities using PatchMatch, then use those disparities to initialize a higher resolution PatchMatch run. This allows the algorithm to detect textures that may only be contained by the window size at certain scales

[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.

Results

Results

Two-View PatchMatch Surface Reconstruction (~500K points)



Results

Two-View PatchMatch Surface Reconstruction (~500K points)

Results

Multi-View PatchMatch Surface Reconstruction (millions of points)

[30] D. Cernea, ***OpenMVS***, https://github.com/cdcseacave/openMVS.

References

[1] A. Collet et al, ***High Quality Streamable Free Viewpoint Video,*** 2015.
[2] M. Dou et al, ***Fusion4D: Real-Time Performance Capture of Challenging Scenes,*** 2016.
[3] P. Alcantarilla, C. Beall, and F. Dellaert, ***Large-Scale Dense 3D Reconstruction from Stereo Imagery,*** 2013.
[4] J. Engel, T. Schops, and D. Cremers, ***LSD-SLAM: Large-Scale Direct Monocular SLAM,*** 2014.
[5] H. Hirschmuller, ***Stereo Processing by Semiglobal Matching and Mutual Information***, 2008.
[6] S. Agarwal et al, ***Building Rome in a Day***, 2011.
[7] ***Matrix Vision mvBlueFox Technical Manual*** (link).
[8] ***Tesla Autopilot Demonstration*** (link).
[9] ***DroneDeploy Thermal Reconstruction Documentation*** (link).
[10] OpenCV 2.4.13.7 Documentation: ***Camera Calibration and 3D Reconstruction*** (link).
[11] Edwin Olson, ***AprilTag: A Robust and Flexible Visual Fiducial System*** 2011.
[12] J. Bouguet, ***Camera Calibration Toolbox for MATLAB****, 1999* (link).
[13] B. Triggs et al, ***Bundle Adjustment - A Modern Synthesis,*** 2000.
[14] R. Hartley and A. Zisserman, ***Multiple View Geometry in Computer Vision,*** 2004.
[15] A. Ranganathan, ***The Levenberg-Marquardt Algorithm,*** 2004.
[16] D. Lowe, ***Distinctive Image Features from Scale-Invariant Keypoints,*** 2004.
[17] R. Mur-Artal et al, ***ORB-SLAM: a Versatile and Accurate Monocular SLAM System***, 2015.
[18] C. Forster et al, ***SVO: Fast Semi-Direct Monocular Visual Odometry***, 2014.
[19] E. Rublee et al, ***ORB: An efficient alternative to SIFT or SURF***, 2011.
[20] E. Rosten and T. Drummond, ***Machine Learning for High-Speed Corner Detection***, 2006.
[21] R. Mur-Artal and J. Tardos, ***Fast Relocalization and Loop Closing in Keyframe-Based SLAM***, 2014.
[22] Fisher Yu, ***Multiple View 3D Reconstruction***, 2014 (link).
[23] M. Bleyer, ***Fundamentals of Stereo Vision***.
[24] R. Szeliski, ***Computer Vision Algorithms and Applications***, 2010.
[25] D.J. Lee, ***Stereo Calibration and Rectification***, 2012.
[26] M. Bleyer, C. Rhemann, and C. Rother, ***PatchMatch Stereo - Stereo with Slanted Support Windows***, 2011.
[27] J. Zbontar and Y. LeCun, ***Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches***, 2016.
[28] J.R. Chang and Y.S. Chen, ***Pyramid Stereo Matching Network***, 2018.
[29] S. Duggal et al, ***DeepPruner: Learning Efficient Stereo Matching via Differentiable PatchMatch,*** 2019.
[30] D. Cernea, ***OpenMVS***, https://github.com/cdcseacave/openMVS.