SLAM, Visual Odometry, Structure from Motion and Multiple View Stereo

Yu Huang

yu.huang07@gmail.com

Sunnyvale, California

Outline

- Calibration
- Camera pose estimation
- Triangulation
- Bundle adjustment
- Filtering-based SLAM
- Key frame-based SLAM
- PTAM/DTAM
- SLD-SLAM/ORB-SLAM
- Direct Sparse Odometry
- MonFusion/MobileFusion
- Image-based Relocalization
- MVS
- Planar/Polar rectification
- Stereo matching
- Plane sweep stereo
- Multiple baseline stereo
- Volumetric Stereo/Voxel Coloring
- Space carving
- (Carved) Visual Hulls
- Region growing/depth fusion
- Patch-based MVS
- Stereo initialization, SfM and MVS
- SDF for surface representation
- Marching cubes
- Poisson surface reconstruction
- Texture mapping
- Appendix 1: Depth sensor-based reconstruction
- Appendix 2: Immersive Panoramic View

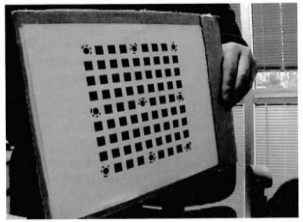
Calibration

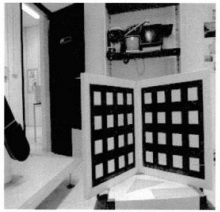
- Find the intrinsic and extrinsic parameters of a camera
  - Extrinsic parameters: the camera's location and orientation in the world.
  - Intrinsic parameters: the relationships between pixel coordinates and camera coordinates.
- Work of Roger Tsai and of Zhengyou Zhang are influential: 3-D node setting and 2-d plane
- Basic idea:
  - Given a set of world points *Pi* and their image coordinates *(ui,vi)*
  - find the projection matrix *M*
  - And then find intrinsic and extrinsic parameters.
- Calibration Techniques
  - Calibration using 3D calibration object
  - Calibration using 2D planer pattern
  - Calibration using 1D object (line-based calibration)
  - Self Calibration: no calibration objects
  - Vanishing points from for orthogonal direction

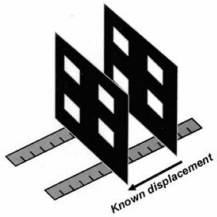
Calibration

- Calibration using 3D calibration object:
  - Calibration is performed by observing a calibration object whose geometry in 3D space is known with very good precision.
  - Calibration object usually consists of two or three planes orthogonal to each other, e.g. calibration cube
  - Calibration can also be done with a plane undergoing a precisely known translation (Tsai approach)
  - (+) most accurate calibration, simple theory
  - (-) more expensive, more elaborate setup

- 2D plane-based calibration (Zhang approach)
  - Require observation of a planar pattern at few different orientations
  - No need to know the plane motion
  - Set up is easy, most popular approach
  - Seems to be a good compromise.



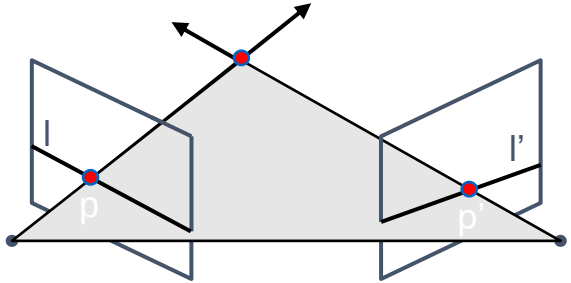




- 1D line-based calibration:
  - Relatively new technique.
  - Calibration object is a set of collinear points, e.g., two points with known distance, three collinear points with known distances, four or more…
  - Camera can be calibrated by observing a moving line around a fixed point, e.g. a string of balls hanging from the ceiling!
  - Can be used to calibrate multiple cameras at once. Good for network of cameras.

Fundamental Matrix

- Let *p* be a point in left image, *p'* in right image



- Epipolar relation
  - *p* maps to epipolar line *l'*
  - *p'* maps to epipolar line *l*

$$l' = Fp$$

$$l = p'F$$

- Epipolar mapping described by a 3x3 matrix *F*

This matrix F is called

- the "Essential Matrix"
  - when image intrinsic parameters are known

$$\boldsymbol{p}^T \mathcal{E} \boldsymbol{p}' = 0 \quad \text{with} \quad \mathcal{E} = [\boldsymbol{t}_\times]\mathcal{R}$$

- the "Fundamental Matrix"
  - more generally (uncalibrated case)

$$\boldsymbol{p}^T \mathcal{F} \boldsymbol{p}' = 0 \quad \text{with} \quad \mathcal{F} = \mathcal{K}^{-T} \mathcal{E} \mathcal{K}'^{-1}$$

Can solve for F from point correspondences

- Each (p, p') pair gives one linear equation in entries of F

- 8/5 points give enough to solve for F/E (8/5-point algo)

Camera Pose Estimation

- 2D-to-2D matching;
- Fundamental matrix estimation: 8-points, 7-points;
- Outlier removal: RANSAC (other methods like M-estimation, LMedS, Hough transform);
- Essential matrix E if camera intrinsic matrix is known: 5-points.
- Extraction of R, t from E: a scaling factor for translation t.

$$\mathbf{E} = \mathbf{U}\mathrm{diag}(1,1,0)\mathbf{V}^T$$

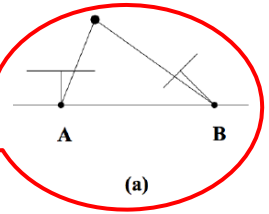
1st Camera matrix $\mathbf{P}_1 = [\mathbf{I}|\mathbf{0}]$

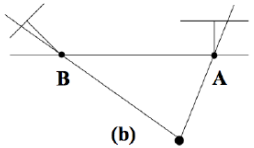
4 possible choices for 2nd camera matrix

$$\mathbf{P}_2 = [\mathbf{U}\mathbf{W}\mathbf{V}^T | +\mathbf{u}_3]$$
$$\mathbf{P}_2 = [\mathbf{U}\mathbf{W}\mathbf{V}^T | -\mathbf{u}_3]$$
$$\mathbf{P}_2 = [\mathbf{U}\mathbf{W}^T\mathbf{V}^T | +\mathbf{u}_3]$$
$$\mathbf{P}_2 = [\mathbf{U}\mathbf{W}^T\mathbf{V}^T | -\mathbf{u}_3]$$

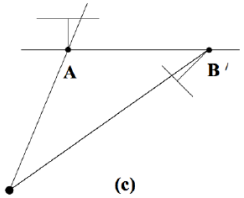
$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

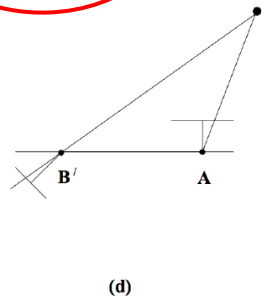
Solution





Angle btw the view direction and the direction from camera center to point





Fundamental Matrix Estimation

- Construct A (nx9) from correspondences
- Compute SVD of A: $A = USV^T$
- Unit-norm solution of Af=0 given by v_n (the right-most singular vector of A)
- Reshape f into F_1
- Compute SVD of F_1: $F_1=U_FS_FV_F^T$
- Set $S_F(3,3)=0$ to get S_F' (The enforces rank(F)=2)
- Reconstruct $F=U_FS_F'V_F^T$
- Now x^TF is epipolar line in image 2, and Fx' is epipolar line in image 1

$$\begin{aligned}(x_1, y_1) &\leftrightarrow (x_1', y_1')\\ (x_2, y_2) &\leftrightarrow (x_2', y_2')\\ (x_3, y_3) &\leftrightarrow (x_3', y_3')\\ &\vdots\\ (x_n, y_n) &\leftrightarrow (x_n', y_n')\end{aligned}$$

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & x_1'y_1 & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ x_2x_2' & x_2y_2' & x_2 & x_2'y_2 & y_2y_2' & y_2 & x_2' & y_2' & 1 \\ x_3x_3' & x_3y_3' & x_3 & x_3'y_3 & y_3y_3' & y_3 & x_3' & y_3' & 1 \\ & & & & \vdots & & & & \\ x_nx_n' & x_ny_n' & x_n & x_n'y_n & y_ny_n' & y_n & x_n' & y_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

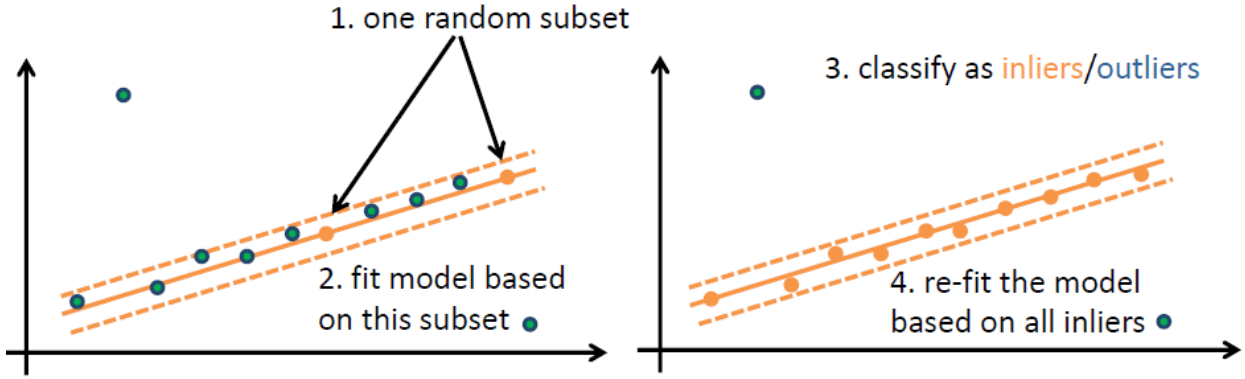
$$A\mathbf{f} = 0$$

RANSAC: RANdom SAmple Consensus

Goal: Robustly fit a model to a data set which contains outliers

Algorithm:

1. Randomly select a (minimal) subset of data points and instantiate the model;
2. Using this model, classify the all data points as inliers or outliers
3. Repeat 1&2 for iterations
4. Select the largest inlier set, and re-estimate the model from all points in it.



Triangulation: A Linear Solution

- Generally, rays C→x and C'→x' will not exactly intersect
- Can solve via SVD, finding a least squares solution to a system of equations

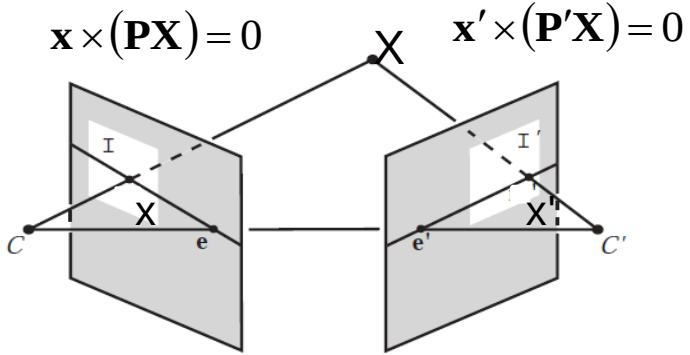
Given **P**, **P'** for cameras, **x**, **x'** for matched points

- Precondition points and projection matrices
- Create matrix **A**
- [U, S, V] = svd(A)
- **X** = V(:, end)

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \mathbf{x}' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$$

Pros and Cons

- Works for any number of corresponding images
- Not projectively invariant

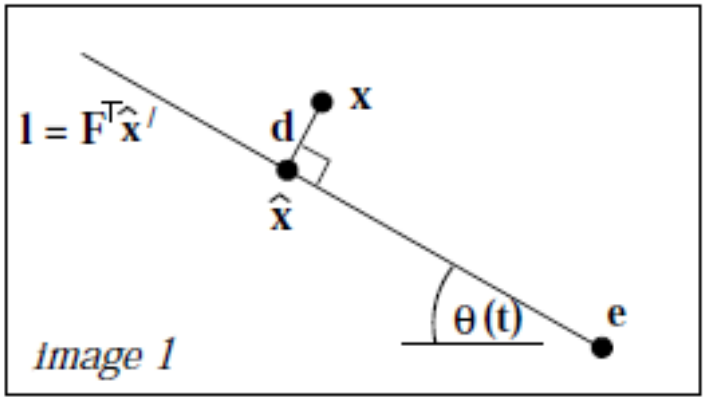


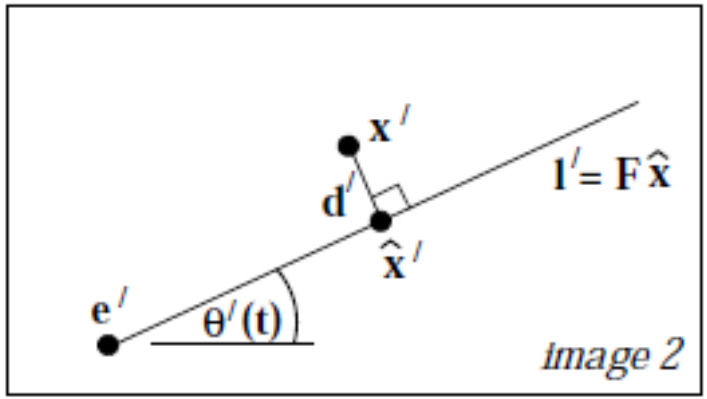
$$\mathbf{AX} = \mathbf{0} \qquad \mathbf{A} = \begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1^T \\ v\mathbf{p}_3^T - \mathbf{p}_2^T \\ u'\mathbf{p}_3'^T - \mathbf{p}_1'^T \\ v'\mathbf{p}_3'^T - \mathbf{p}_2'^T \end{bmatrix}$$

Triangulation: Non-Linear Solution

- Minimize projected error while satisfying $\mathbf{x}^T\mathbf{F}\mathbf{x} = 0$

$$C(\mathbf{X}) = d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x}}')^2$$





- Solution is a 6-degree polynomial of *t*, minimizing

$$d(\mathbf{x}, \mathbf{l}(t))^2 + d(\mathbf{x}', \mathbf{l}'(t))^2$$

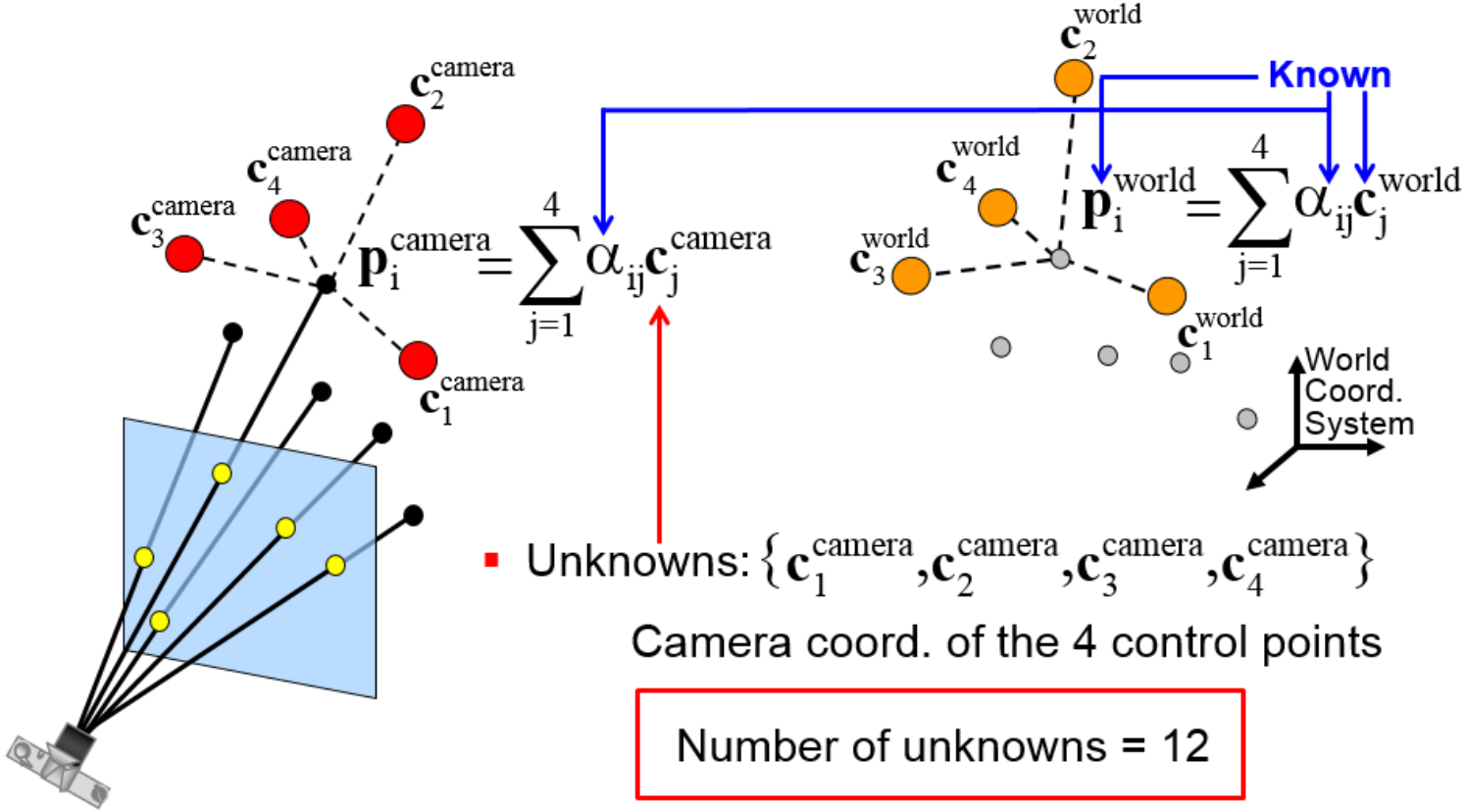
PnP (Perspective n Points): 3D-to-2D Matching

- Motion estimation from 3-D-to-2-D correspondences is more accurate than from 3-D-to-3-D correspondences because it minimizes the image reprojection error, instead of the 3-D-to-3D feature position error [Nist'04].
- Efficient PnP: an non-iterative method.

  First, 3D points expressed as a weighted sum of 4 control points, then

  1. The control points coordinates are the (12) unknowns.
  2. Project 3D points onto image→Build a linear system in control points coordinates.
  3. Express control points coordinates as linear combination of the null eigenvectors.
  4. The weights (the β_i) are the new unknowns (not more than 4).
  5. Add rigidity constraints to get quadratic equations in the β_i .
  6. Solve for the β_i depending on their number (*linearization* or *re-linearization*).

PnP (Perspective n Points): 3D-to-2D Matching



- Unknowns: $\{\mathbf{c}_1^{camera}, \mathbf{c}_2^{camera}, \mathbf{c}_3^{camera}, \mathbf{c}_4^{camera}\}$

Camera coord. of the 4 control points

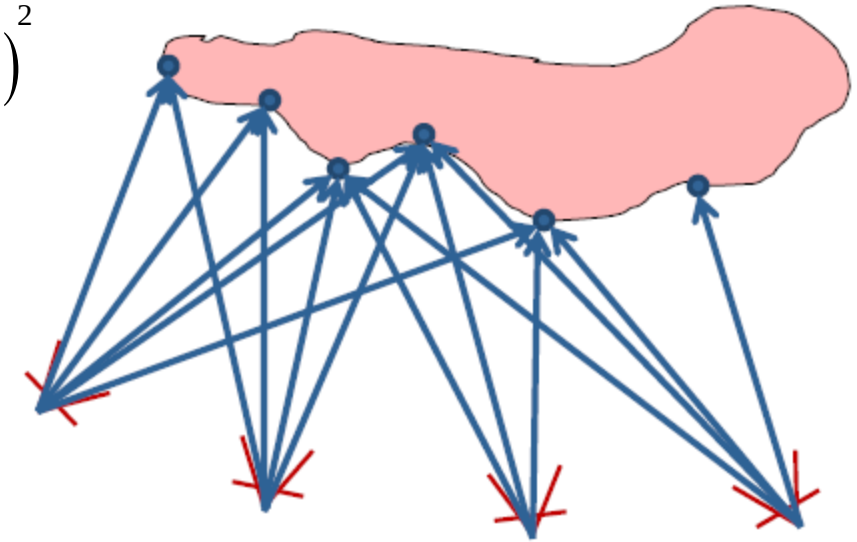
Number of unknowns = 12

Bundle Adjustment or Pose Graph Optimization: Refinement of Both Structure and Camera Pose

- Pose graph optimization: optimize the camera pose;
- BA: Optimize both 6DOF camera poses and 3D (feature) points;

$$E(\mathbf{P},\mathbf{X}) = \sum_{i=1}^{m}\sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, \mathbf{P}_i\mathbf{X}_j\right)^2$$

  - Nonlinear;
  - Sparse;
  - Optimization.
- Algorithms:
  - Schur Complement;
  - Levenberg-Marquardt iteration;

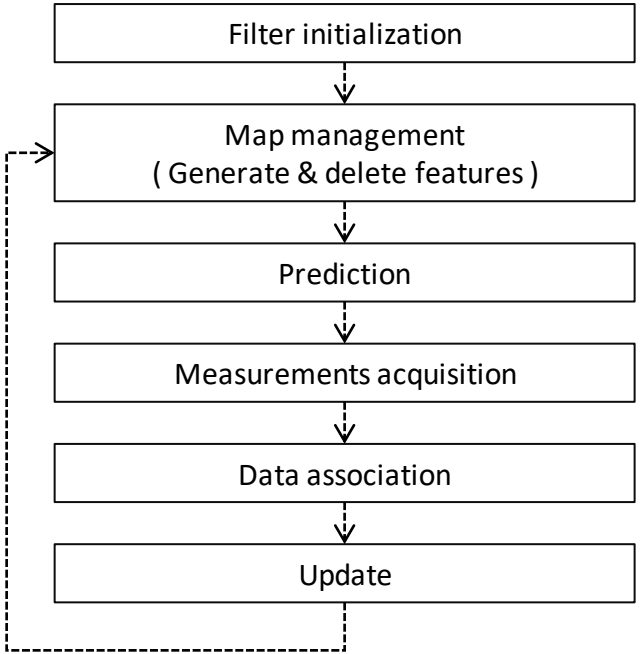


Filtering-based SLAM

- SLAM: A process by which a mobile robot can build a map of an environment and at the same time use this map to compute its own location;
- Different scenarios: indoor, outdoor, underwater, and airborne systems;
- A State Space Method (SSM): both observation and motion (state transition) models
  - Observation model describes prob. of making an observation (landmarks) when the robot location/orientation and landmark locations (map) are known;
  - Motion model is a prob. distrib. on state (location and orientation of the robot) transitions with applied control, i.e. a Markov process;
  - By recursive Bayes estimation, update robot state and environment maps simultaneously;
    - the map building problem is solved by fusing observations from different locations;
    - the localization problem is to compute the robot location by estimating its state;
- Convergence: correlation btw landmark estimates increase monotonically and more observations made;
- Typical methods: Kalman filter/Particle filter;
- Robot relative location accuracy = localization accuracy with a given map.

Filtering-based SLAM

- EKF-based



$\mathbf{S}_i$: a covariance matrix for the 2D position of i th landmark

State

$$\mathbf{x}(t-1) = \begin{bmatrix} \mathbf{x}_v(t-1) \\ \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \end{bmatrix}$$

Prediction

$$\hat{\mathbf{x}}_v(t) = \begin{bmatrix} \mathbf{r}^W(t-1) + \mathbf{v}^W(t-1)\cdot\Delta t \\ \mathbf{q}^{WR}(t-1)\times\mathbf{q}(\boldsymbol{\omega}^R(t-1)\cdot\Delta t) \\ \mathbf{v}^W(t-1) \\ \boldsymbol{\omega}^R(t-1) \end{bmatrix}$$

Dynamic System Model
(Constant Velocity Model)

Prediction of measurements

$$\mathbf{h}_i(t) = h_i(\hat{\mathbf{x}}_v(t)) = \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\mathbf{K}\ \hat{\mathbf{r}}^W(t),\ \hat{\mathbf{q}}^{WR}(t) \rightarrow \begin{bmatrix} ku \\ kv \\ k \end{bmatrix} = \mathbf{P}(t)\cdot\begin{bmatrix} \mathbf{y}_i \\ 1 \end{bmatrix}$$

Find measurements

For $\mathbf{u} = \left\{(u,v)^{\mathrm{T}} \mid \mathbf{u}\cdot\mathbf{S}_i^{-1}\cdot\mathbf{u} < \chi\right\}$
Patch matching or tracking
at $\mathbf{h}_i(t) + \mathbf{u}$
Get measurement at new pos $\mathbf{h}_i(t) + \mathbf{u}'$

$$\mathbf{z}_i(t) = \mathbf{h}_i(t) + \mathbf{u}'$$

Update

$$\mathbf{x}(t) = \hat{\mathbf{x}}(t) + \mathbf{K}(t)\cdot\begin{bmatrix} \mathbf{z}_1(t) - \mathbf{h}_1(t) \\ \vdots \\ \mathbf{z}_n(t) - \mathbf{h}_n(t) \end{bmatrix}$$

$$\mathbf{x}_v(t) = \begin{bmatrix} \mathbf{r}^W(t) \\ \mathbf{q}^{WR}(t) \\ \mathbf{v}^W(t) \\ \boldsymbol{\omega}^R(t) \end{bmatrix}$$

: 3D position vector
: orientation quaternion
: linear velocity vector
: angular velocity vector

$\mathbf{y}_i$: landmark position vector

Comparison of SFM and Filtering-based

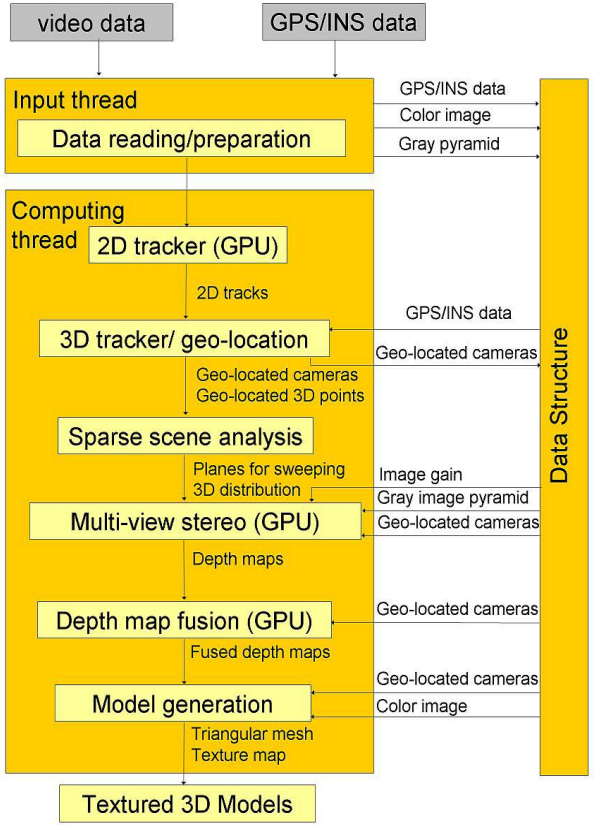
Method	SfM-based	Filtering-based
Initialization	8-point algorithm 5-point algorithm	Delayed: SfM Undelayed: Inverse depth parameterization
measurement	Feature tracking Feature matching	Feature tracking Feature matching
Estimation	Camera pose + Triangulation LBA, GBA	Kalman filtering Particle filter
Performance	Tracking 3-400 points	100 points in real-time

Key frame-based SLAM

- Bootstrapping
  - Compute an initial 3D map
  - Mostly based on concepts from two-view geometry
- Normal mode
  - Assumes a 3D map is available and incremental camera motion
  - Track points and use PnP for camera pose estimation
- Recovery mode
  - Assumes a 3D map is available, but tracking failed: no incremental motion
  - Relocalize camera pose w.r.t. previously reconstructed map
- Key-Frame BA: keep subset of frames as keyframes
  - State: 3D landmarks + camera poses for all key frames
  - BA can refine state in a thread separate from tracking component

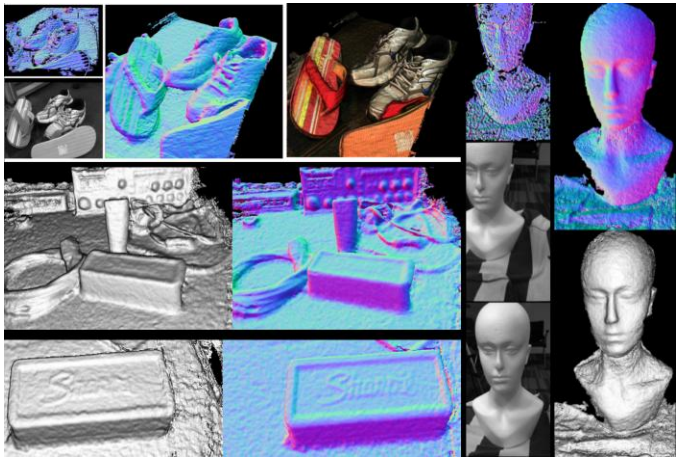
Real Time Urban 3D Reconstruction from Video

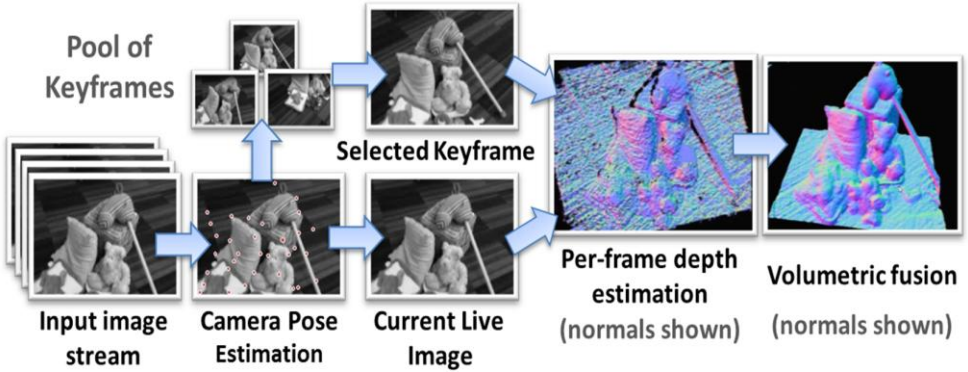
- Automatic georegistered real time reconstruction of urban scenes;
- 2D tracking: salient features, KLT;
- 3D tracking: Kalman filter for 2D and GPS/INS, then pose using SfM;
- Sparse scene analysis: 3 orthogonal sweeping directions (one for ground, two for facades);
- Stereo depth estimation: plane sweep stereo;
- Depth map fusion: handle conflicts and errors by visibility, stability and confidence-based;
- Model generation: triangular mesh and texture mapped.

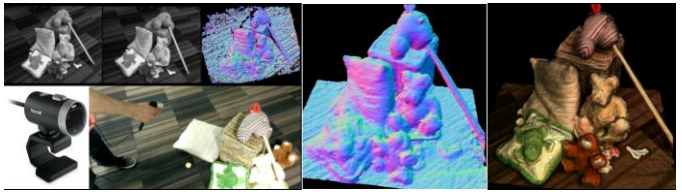


MonoFusion: Real-time 3D Reconstruction of Small Scene

- Dense 3D reconstruction by single camera;
- Sparse feature tracking for 6DoF camera pose estimation;
  - Key frame based BA;
- Dense stereo matching btw two key frames;
  - Patchmatch stereo.
- Depth fusion by SDF-based volumetric integration.







PTAM (Parallel Tracking and Mapping)

Use large number (thousands) of points in the map;

Don't update the map every frame: Keyframes;

Split the tracking and mapping into two threads;

Make pyramid levels for tracking and mapping;

- Detect Fast corners;
- Use motion model to update camera pose;

Key frames/new map points are added conditionally;

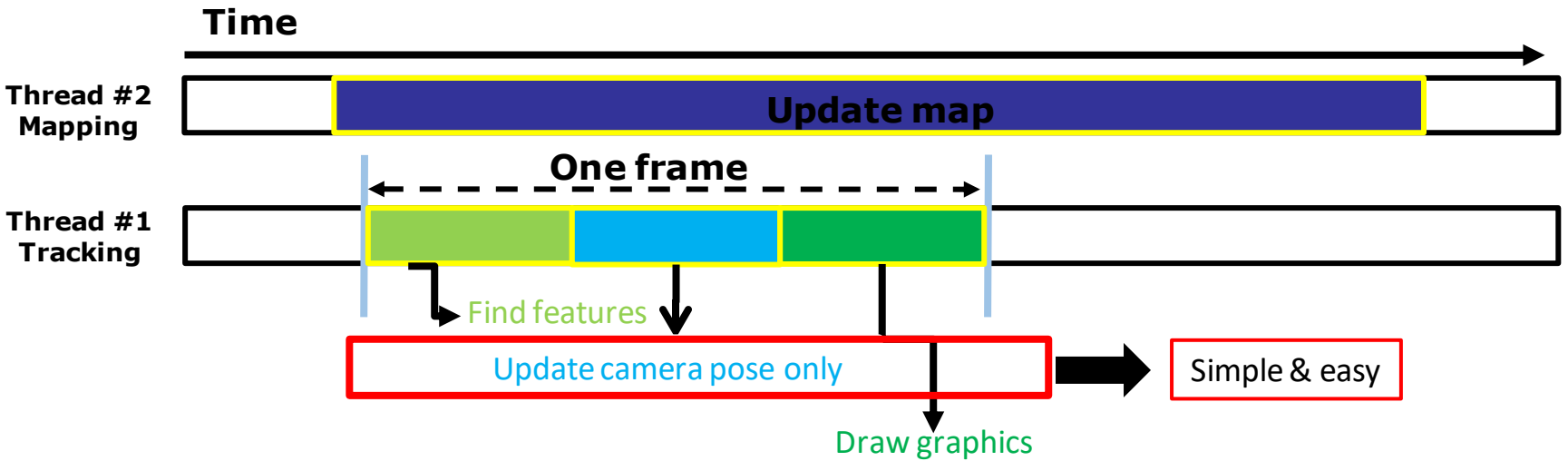
- At least sufficient baseline and good epipolar search;

Map optimization by batch method: Bundle Adjustment.

Stereo initialization densely: two clicks for key frames

- Use five-point-pose algorithm;

Map maintained with data association and outlier retest.



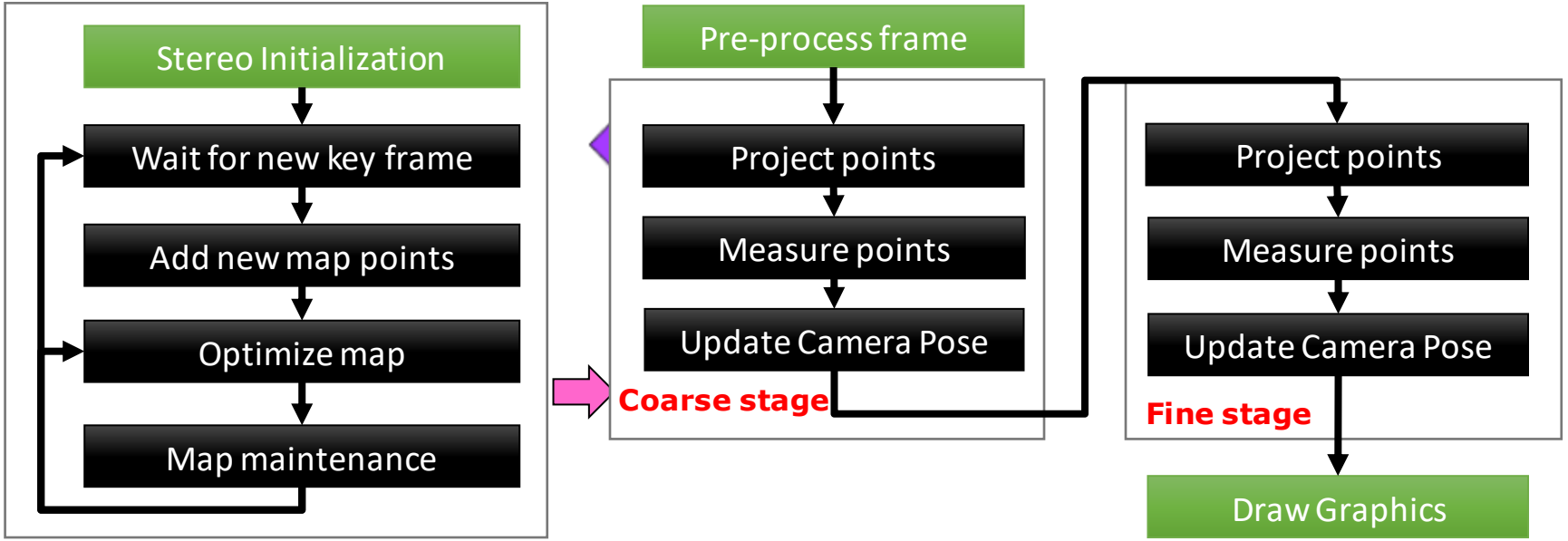
PTAM (Parallel Tracking and Mapping)

Mapping thread:

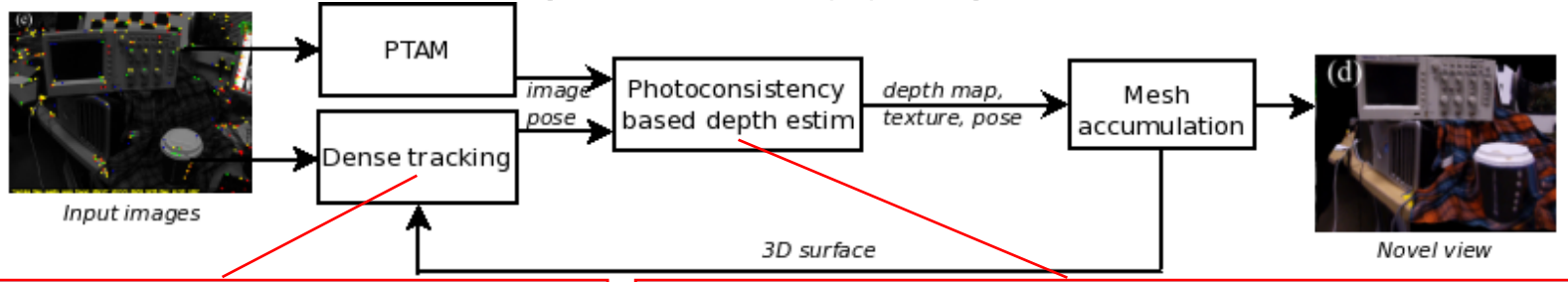
- Responsible for providing the map
- Can take lots of time per key frame
- Make as rich and accurate as possible

Tracking thread:

- Responsible estimation of camera pose and rendering augmented graphics
- Must run at 30 Hz
- Make as robust and accurate as possible



DTAM (Dense Tracking and Mapping)



Inputs:

- 3D texture model of the scene
- Pose at previous frame

Tracking as a registration problem

- First rotation estimation: previous is aligned on the current to estimate a coarse inter-frame rotation;
- Estimated pose is used to project the 3D model into 2.5D image;
- The 2.5D image is registered with the current frame to find the current camera pose;
- Two template matching problems for minimize SSD: direct and inverse compositional.

Principle:

- **S** depth hypothesis are considered for each pixel of the reference image $\mathbf{I}_r$
- Each corresponding 3D point is projected onto a bundle of images $\mathbf{I}_m$
- Keep the depth hypothesis that best respects the color consistency from the reference to the bundle of images

Formulation:

$$\mathbf{C}_r(\mathbf{u},d) = \frac{1}{|\mathcal{I}(r)|} \sum_{m \in \mathcal{I}(r)} \|\rho_r\left(\mathbf{I}_m, \mathbf{u}, d\right)\|_1 \qquad \pi(\mathbf{x}_c) = (x/z, y/z)^T \qquad \pi^{-1}(\mathbf{u},d) = \frac{1}{d}K^{-1}\dot{\mathbf{u}}$$

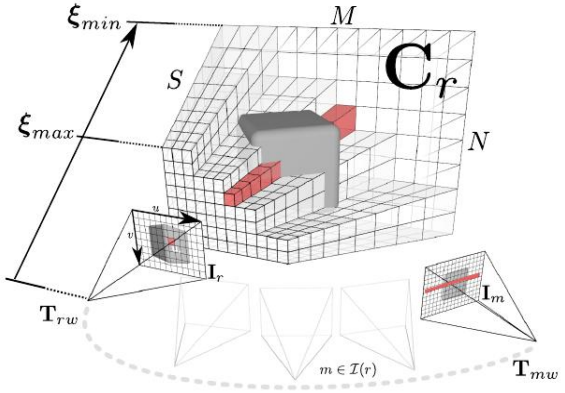
- $\mathbf{u}, d$: pixel position and depth hypothesis
- $|\mathcal{I}(r)|$: number of valid reprojection of the pixel in the bundle
- ρ_r : photometric error between reference and current image

$$\rho_r\left(\mathbf{I}_m, \mathbf{u}, d\right) = \mathbf{I}_r\left(\mathbf{u}\right) - \mathbf{I}_m\left(\pi\left(\mathrm{K}\mathrm{T}_{mr}\pi^{-1}\left(\mathbf{u},d\right)\right)\right)$$

DTAM (Dense Tracking and Mapping)

Problem (Depth Estimation): Uniform regions in reference image do not give discriminative enough photometric error;

Solution (Primal Dual): Assume that depth is smooth on uniform regions, use total variation approach where depth map is the functional to optimize, where photometric error defines the data term and the smoothness constraint defines the regularization.



Problem in SSD minimization:

Align template T(x) with input I(x).

Formulation:

Find transform $\mathbf{W}(\mathbf{x};\mathbf{p})$ that best maps the pixels of the templates into the ones of the current image, minimize:

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

Formulation of a variational approach:

$$E_{\boldsymbol{\xi}} = \int_{\Omega} \left\{ g(\mathbf{u}) \|\nabla \boldsymbol{\xi}(\mathbf{u})\|_{\epsilon} + \lambda \mathbf{C}\left(\mathbf{u}, \boldsymbol{\xi}(\mathbf{u})\right) \right\} \mathrm{d}\mathbf{u} \quad g(\mathbf{u}) = e^{-\alpha \|\nabla \mathbf{I}_r(\mathbf{u})\|_2^{\beta}}$$

- First term as regularizer with Huber norm, second term as photometric error;
- Problem: optimizing this equation directly requires linearising of cost volume, expensive and cost volume has many local minima.

Approximation:

$$\mathbf{E}_{\boldsymbol{\xi},\boldsymbol{\alpha}} = \int_{\Omega} \left\{ g(\mathbf{u}) \|\nabla \boldsymbol{\xi}(\mathbf{u})\|_{\epsilon} + \frac{1}{2\theta} \left(\boldsymbol{\xi}(\mathbf{u}) - \boldsymbol{\alpha}(\mathbf{u})\right)^2 + \lambda \mathbf{C}\left(\mathbf{u}, \boldsymbol{\alpha}(\mathbf{u})\right) \right\} \mathrm{d}\mathbf{u} \,.$$

- Introduce $\boldsymbol{\alpha}(\mathbf{u})$ as an auxiliary variable, can optimized with heuristic search
- Second terms brings original and auxiliary variable together

Reformulation of regularization with primal dual method

- Dual variable $\mathbf{p}$ is introduced to compute the TV norm:

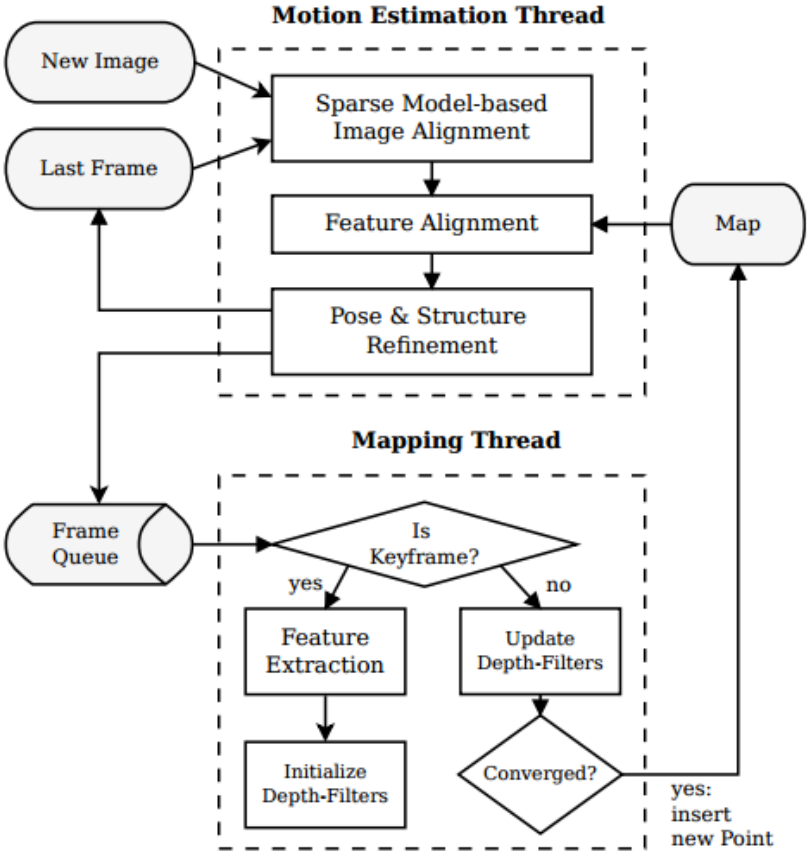
$$\boldsymbol{p} \cdot \nabla u = \frac{\nabla u}{|\nabla u|} \nabla u = |\nabla u|$$

$$TV(u) = \max \left\{ \int_{\Omega} \boldsymbol{p} \cdot \nabla u \, d\Omega : \|\boldsymbol{p}\| \leq 1 \right\} \text{ for } \nabla u \neq 0, \; \|\boldsymbol{p}\| = \left\| \frac{\nabla u}{|\nabla u|} \right\| = 1$$

REMODE: Regularized Monocular Depth Estimation

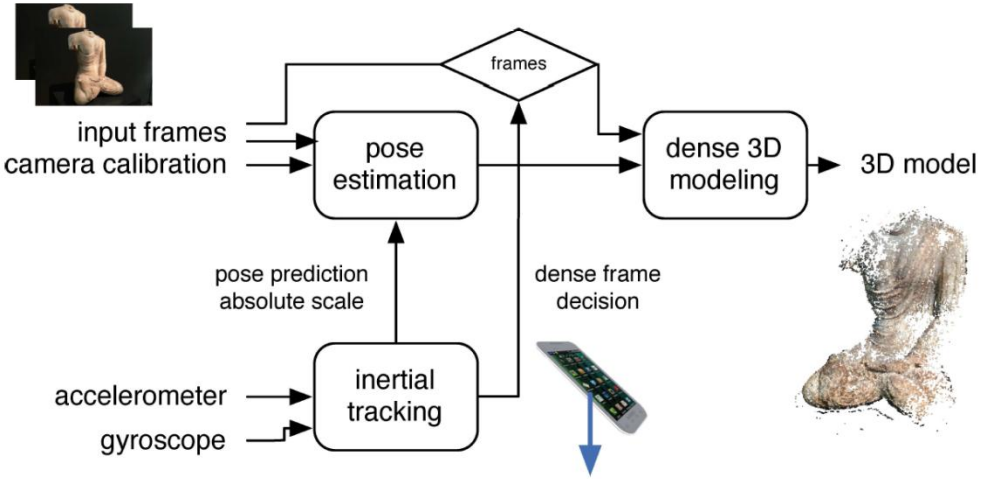
- Combine Bayesian estimation and convex optimization for processing;
  - Probabilistic depth map handles the uncertainty and allows sensor fusion.
- Depth estimated by triangulating a reference view and the last view;
  - Smoothness of depth map enforced by a regularization term (TV).
- Camera pose estimation: **Semi-direct monocular visual Odometry (SVO)**;
  - Particle/Kalman filter or LS optimization (such as key frames-based or batch Bundle Adjustment) solution;
  - **SVO**: Feature correspondence from direct motion estimation with sparse depth map estimate (motion estimation and mapping together), still a sparse model-based alignment method, related to model-based dense method;
    - Initial guess by direct method and then followed by feature-based refinement with reprojection error;
    - A prob. depth filter initialized by sparse features, then posterior **depth estimate** by **Bayesian** rule and update sequentially.
- Scene reconstruction: update after initial estimation with two frames.
  - Region growing (low textured, sparse data sets) or occlusion-robust photo consistency (dense stereo, 3-d segment.);
  - **Inverse depth**: A 6-d vector for parallax, a relation btw disparity and depth.
    - The ray from the 1st camera position, the current camera orientation and the inverse depth;
    - Switch from inverse depth to XYZ (3-d point) for high parallax features.
  - Extend SVO with dense map estimate within Bayesian framework.

SVO: Semi-direct Monocular Visual Odometry



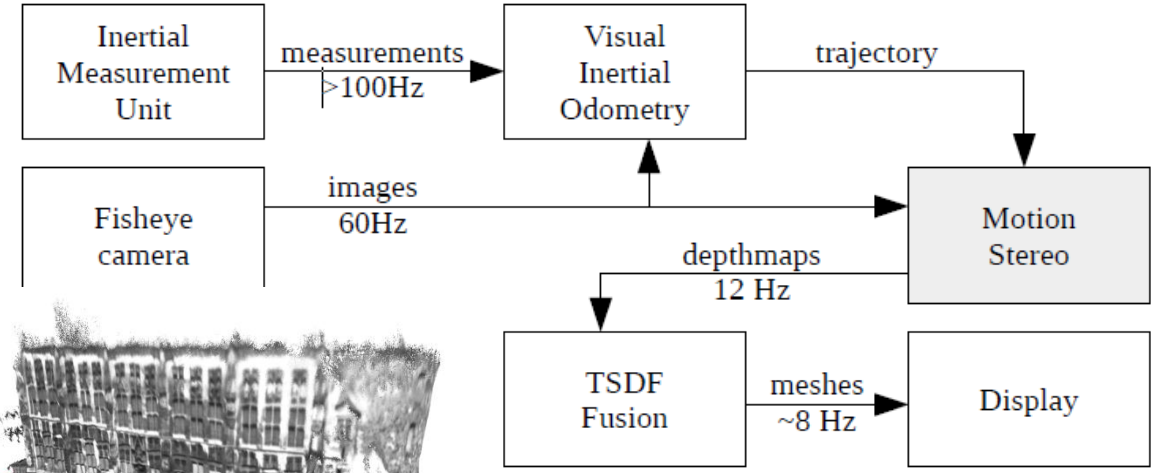
Live Metric 3D Reconstruction with A Mobile Camera

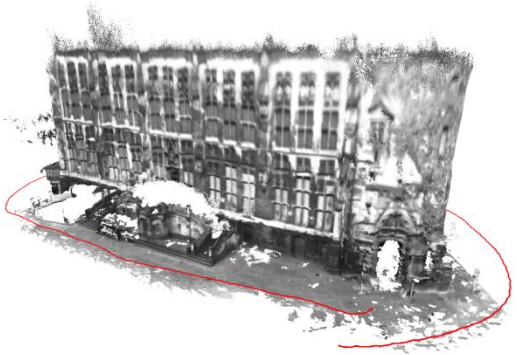
- Absolute scale on site for user interactive feedback;
  - Inertial sensors for scale estimation;
- Feature based tracking and mapping + sensor (gyro, accelero);
  - Two view initialization;
  - BA for drifting correction;
  - Automatically select keyframes;
- Dense 3D modeling by mobile dev.;
  - Image mask estimation;
  - Dense stereo matching;
  - Depth map filtering.



3D Modeling on the Go: Interactive 3D Reconstruction of Large-Scale Scenes on Mobile Devices

- Reconstruct scenes “on the go” by simply walking around them;
- Use monochrome fisheye images (not active depth cameras);
- Compute depth maps using plane sweep stereo (GPUs);
- Fuse the depth maps into a global model of the environment represented as a truncated signed distance function in a spatially hashed voxel grid;
- Importance of filtering outliers in the depth maps;



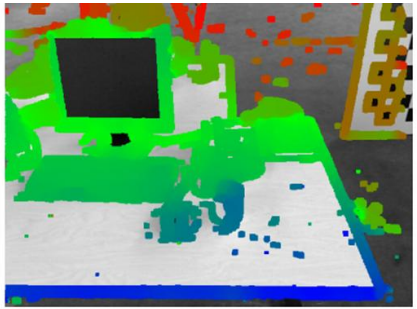


Semi-Dense SLAM

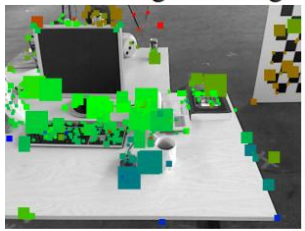
- Continuously estimate a semi-dense inverse depth map for the current frame, which in turn is used to track the motion of the camera using dense image alignment.
  - Estimate the depth of all pixels which have a non-negligible image gradient.
- Use the oldest frame, where the disparity range and the view angle within a certain threshold;
  - If a disparity search is unsuccessful, the pixel's "age" is increased, such that subsequent disparity searches use newer frames where the pixel is likely to be still visible.
- Each estimate is represented as a Gaussian probability distribution over the inverse depth;
  - Minimize photometric and geometric disparity errors with respect to pixel-to-inverse depth ratio.
- Propagate this information over time, update with new measurements as new images arrive.
  - Based on the inverse depth estimate for a pixel, the corresponding 3D point is calculated and projected into the new frame, providing an inverse depth estimate in the new frame;
  - The hypothesis is assigned to the closest integer pixel position – to eliminate discretization errors, the sub-pixel image location of the projected point is kept, and re-used for the next propagation step;
  - Assign inverse depth value the average of the surrounding inverse depths, weighted by respective inverse variance;
  - Keep track of validity of each inverse depth hypothesis to handle outliers (removed if needed).

Semi-Dense SLAM

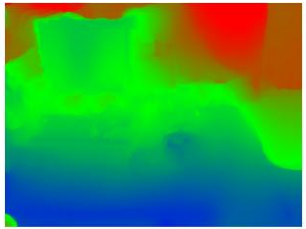
original image



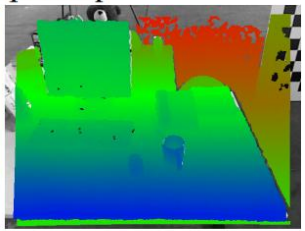
semi-dense depth map



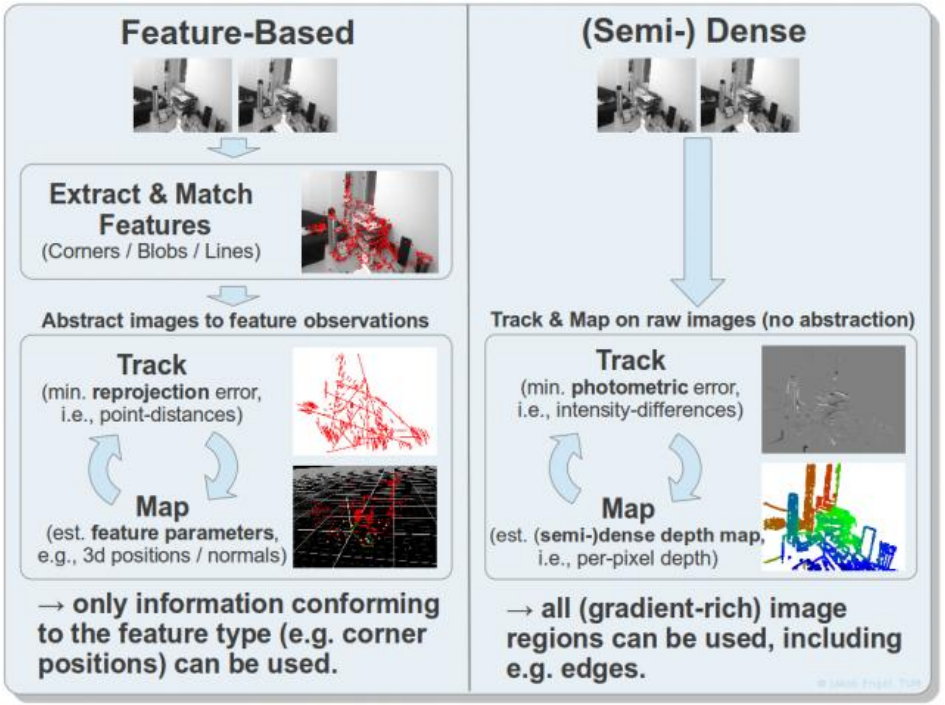
keypoint depth map



dense depth map



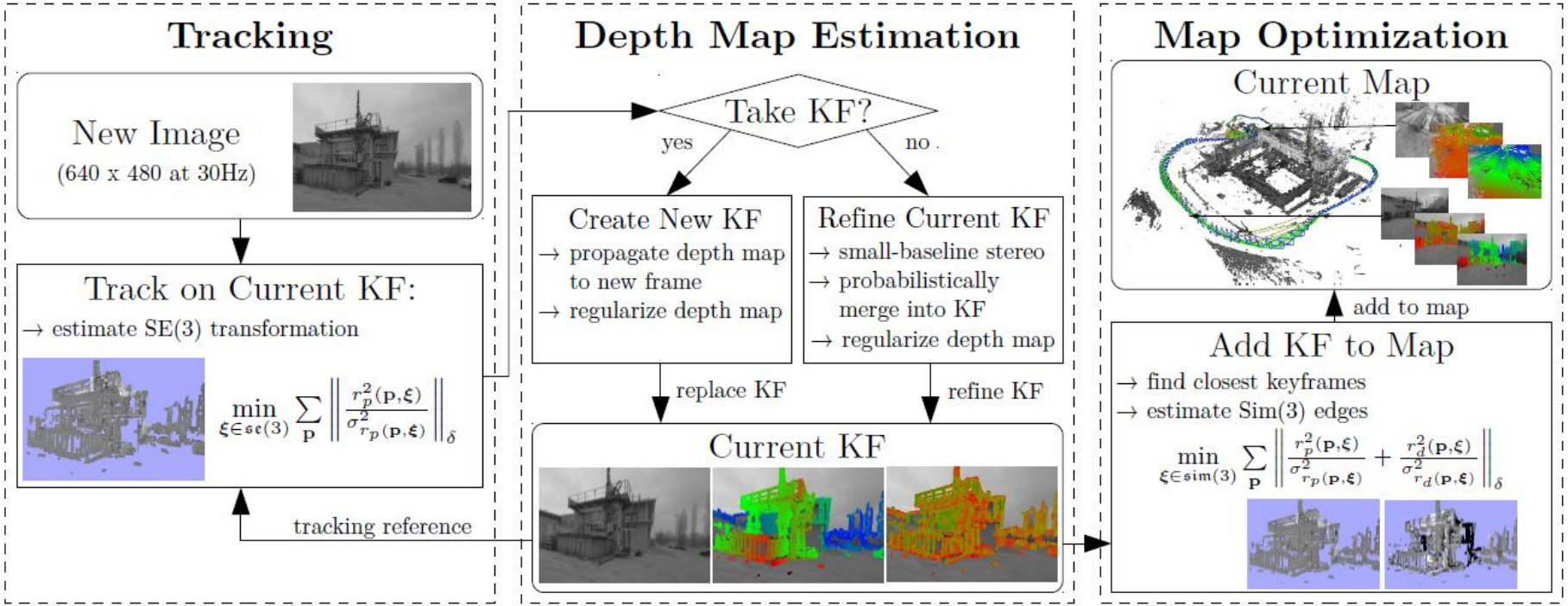
RGB-D camera



Large Scale Direct SLAM (LSD-SLAM)

- Goal: to build large-scale, consistent maps of the environment;
- Initialization: a first keyframe with a random depth map and large variance, given sufficient translational camera movement in the first seconds, converge to a correct depth configuration after a couple of keyframe propagations;
- Pose estimation based on direct scale-aware image alignment on similarity transform;
- Keyframes are replaced when the camera moves too far away from the existing map;
- Once a frame is chosen as keyframe, its depth map is initialized by projecting points from the previous keyframe into it, followed by regularization and outlier removal;
- The depth map is scaled to have a mean inverse depth of one; then refined by filtering over a large number of pixelwise small-baseline stereo;
- 3D reconstruction in real-time as pose-graph of keyframes with semi-dense (gradient) depth;
- Probabilistically consistent incorporation of uncertainty of the estimated depth;
- Loop Closure is detected by appearance-based mapping (BovW with co-occurrence statistics) .

Large Scale Direct SLAM (LSD-SLAM)



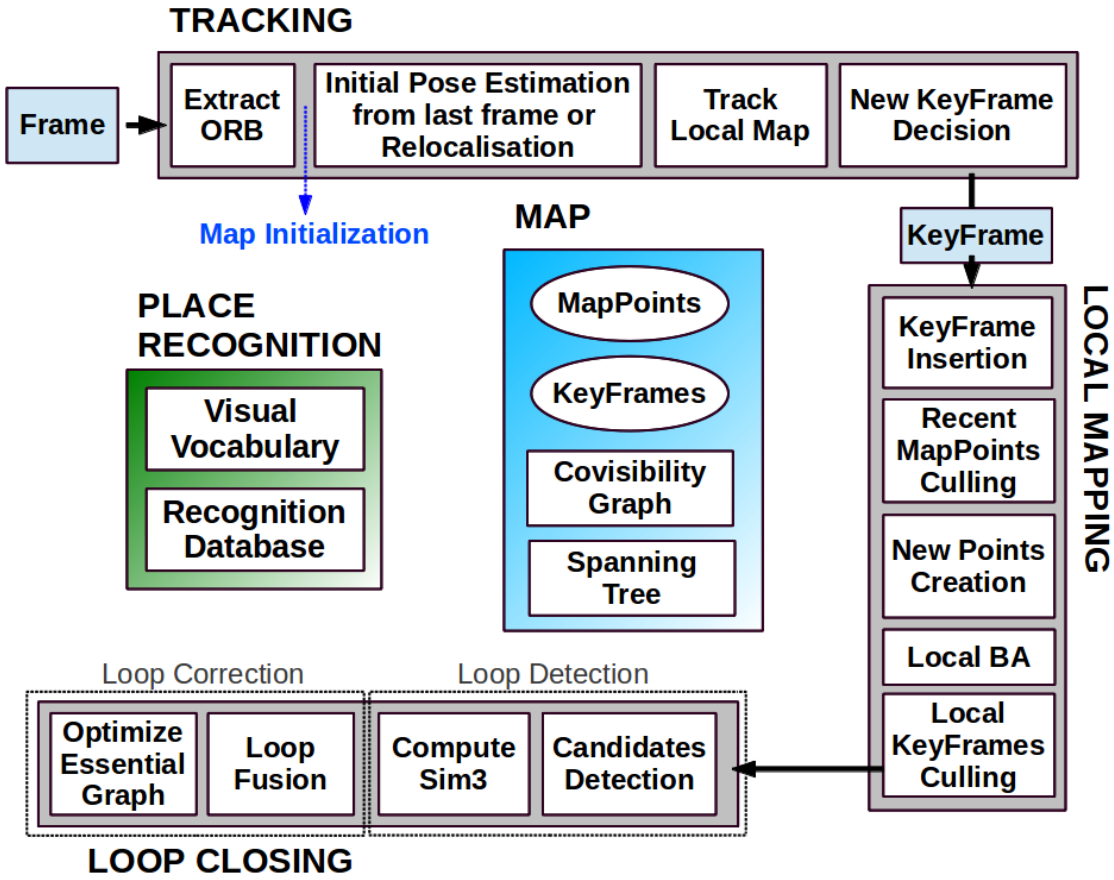
MobileFusion

- Real-time volumetric surface reconstruction and dense 6DoF camera tracking running ;
- Using the RGB camera, scan objects of varying shape, size, and appearance in seconds;
- Point-based 3D models and a connected 3D surface model ;
- Dense 6DoF tracking, key-frame selection, and dense per-frame stereo matching;
- Depth maps are fused volumetrically using a method akin to KinectFusion.

ORB-SLAM: Real-Time Monocular SLAM

- Feature-based real time mono SLAM in small/large, indoor/outdoor environments;
- Use of the same features for all tasks: tracking, mapping, relocalization and loop closing;
- Robust to motion clutter, wide baseline loop closing, relocalization, auto initialization;
- Real time loop closing based on optimization of pose graph called the Essential Graph;
- Real time camera relocalization with significant invariance to viewpoint and illumination;
- Automatic initialization based on model selection that permits to create an initial map of planar and non-planar scenes;
- A survival of the fittest strategy selects the points and keyframes of the reconstruction.

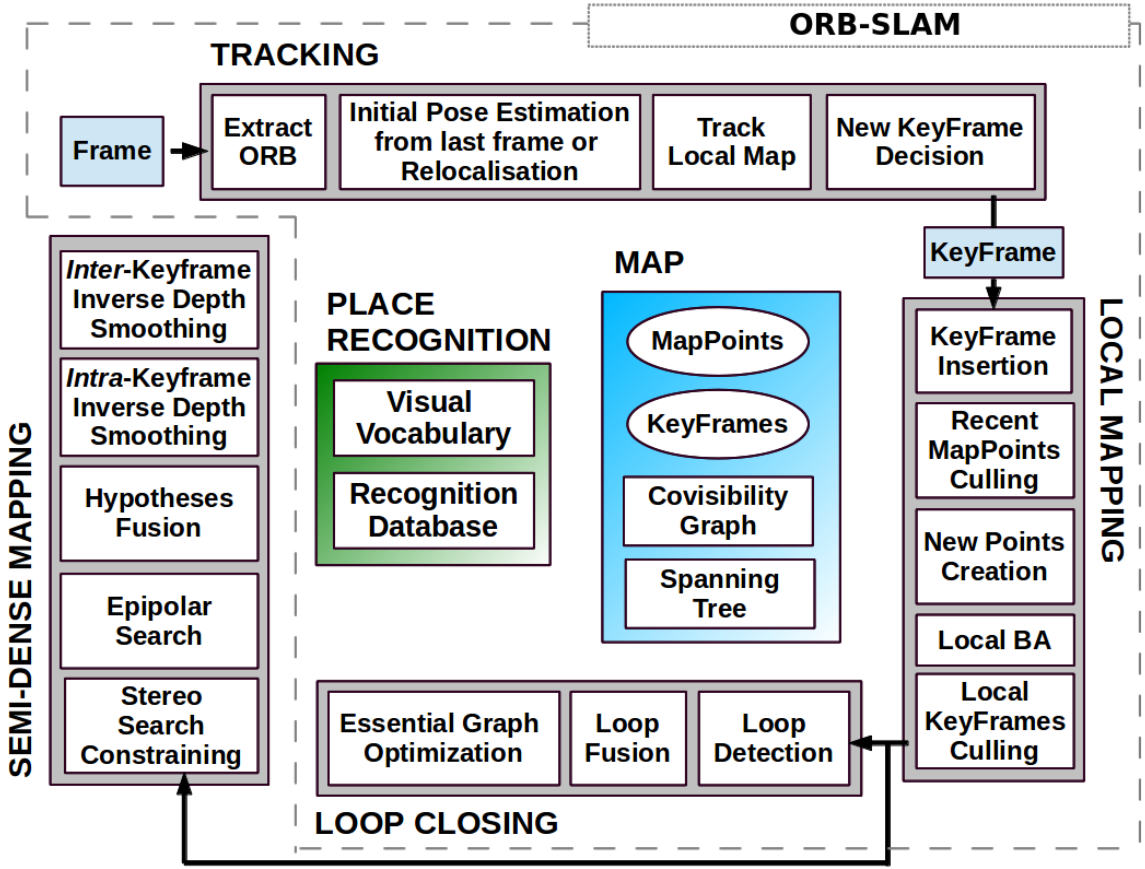
ORB-SLAM: Real-Time Monocular SLAM



ORB-SLAM v.s. LSD-SLAM

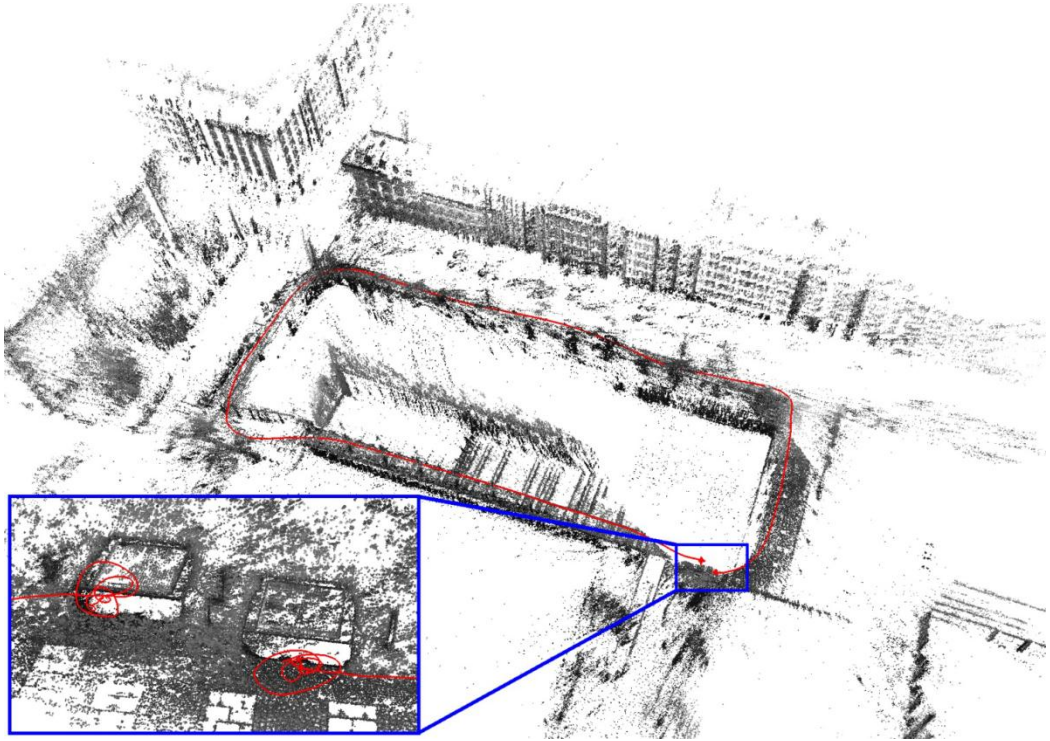
	Feature-Based SLAM (ORB-SLAM)	**Direct SLAM** (LSD-SLAM)
Matching	Illumination/viewpoint invariance *Wide baseline*	Photometric consistency *Narrow baseline*
Map Optimization	Local BA + Global Pose Graph	Global Pose Graph
Loop Detection	Integrated Place Recognition	Need Features (FabMap)
# Points	~300	~100K
Strengths	Excellent accuracy Robust in dynamic scenes Robust initialization	Robust in low texture areas Robust under defocus/motion blur
Weakness	Low texture areas Motion blur	Dynamic Objects Strong viewpoint/illumination changes Rolling-shutter cameras

ORB-SLAM: Adding Semi-Dense Mapping



Direct Sparse Odometry

- Combination: Sparse+Indirect, Dense+Indirect, Dense+Direct, **Sparse+Direct (proposed)**;
- Continuous optimization of the **photometric error** over a **window** of recent frames, including **geometry** (reverse depth in a reference frame) and camera **motion** (pose);
- Integrated A photometric camera model (calibrated): lens attenuation, gama correction, and known exposure times;
- Run in real-time on CPU laptop.



Direct Sparse Odometry

- Photometric error defined as weighted SSD over a small neighbor;
- Windowed optimization by **G-N** algo.;
  - Marginalization of old variables using Schur complement;
- Keep a window of up to 7 **key frames**;
- Keep a fixe number 2000 **active points equally distributed across space/frames**;
- Initialization: 2 frame image alignment;
- Key frame creation: scene change, camera translation with occlusion and dis-occlusion, exposure time changes;
- Key frame marginalization: keep latest two, remove a) <5% points visible, b) maximizing "distance score".

$$E_{\text{photo}} := \sum_{i\in\mathcal{F}} \sum_{\mathbf{p}\in\mathcal{P}_i} \sum_{j\in\text{obs}(\mathbf{p})} E_{\mathbf{p}j}. \tag{8}$$

factor graph

$$E_{\mathbf{p}j} := \sum_{\mathbf{p}\in\mathcal{N}_\mathbf{p}} w_\mathbf{p} \left\| (I_j[\mathbf{p}'] - b_j) - \frac{t_j e^{a_j}}{t_i e^{a_i}} (I_i[\mathbf{p}] - b_i) \right\|_\gamma, \tag{4}$$

exposure time

Affine brightness transfer function $e^{-a_i}(I_i - b_i)$

$$\mathbf{p}' = \Pi_\mathbf{c}\left(\mathbf{R}\,\Pi_\mathbf{c}^{-1}(\mathbf{p}, d_\mathbf{p}) + \mathbf{t}\right), \tag{5}$$

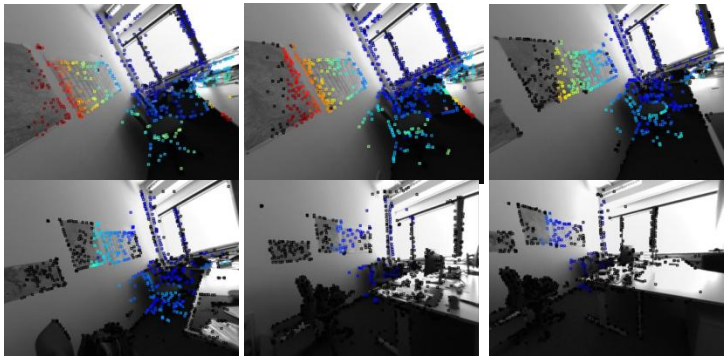
Inverse depth

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} := \mathbf{T}_j \mathbf{T}_i^{-1}. \tag{6}$$

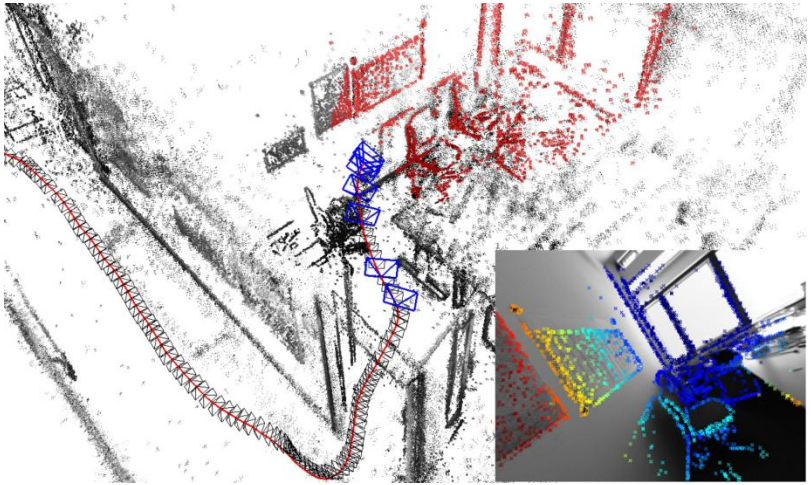
$$w_\mathbf{p} := \frac{c^2}{c^2 + \|\nabla I_i(\mathbf{p})\|_2^2}, \tag{7}$$

$$E_{\text{prior}} := \sum_{i\in\mathcal{F}} \left(\lambda_a a_i^2 + \lambda_b b_i^2\right). \tag{9}$$

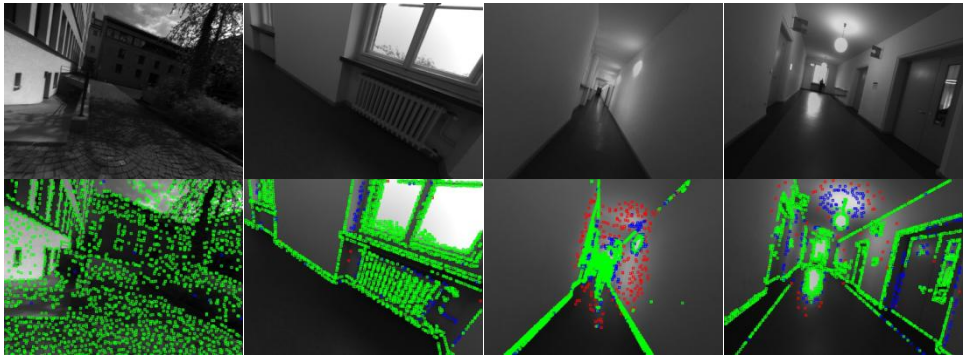
Direct Sparse Odometry



Key frame management

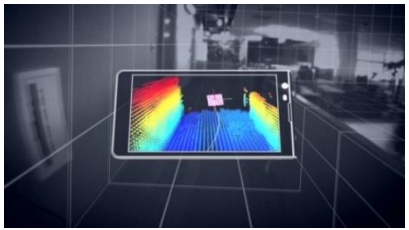


- Candidate selection:
  - Region-adaptive gradient threshold;
- Candidate point tracking:
  - Discrete search along epipolar line;
- Candidate point activation;
  - Maintain a uniform spatial distribution;
- Outlier and occlusion detection;
  - Not distinct, bigger photometric error;



SLAM in VR

- Project Tango (Google)
  - Visual-inertial odometry
  - Area learning (SLAM)
  - RGB-D sensor



- Hololens (Microsoft)
  - RGB-D sensor
- Magic Leap
  - Unknown

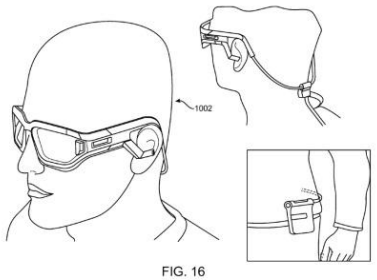
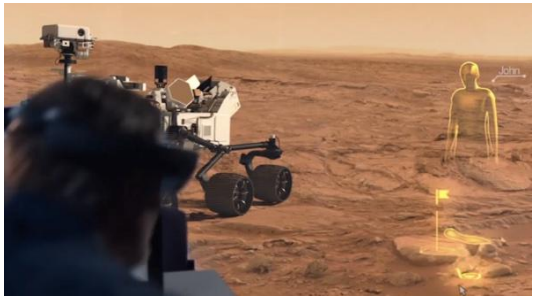
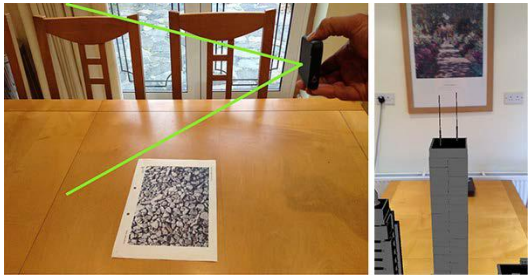


FIG. 16

- Vuforia (Qualcomm)
  - Extended tracking (monocular SLAM)
- Apple
  - Metaio
- Oculus
  - SurrealVision





Visual Odometry

Visual Odometry (V0): The process of estimating the **ego-motion** of an agent using only the input of a single or multiple camera;

VO incrementally estimates pose of the agent through examination of the changes that motion induces on the images of its onboard cameras;

VO: a special case of SfM (relative camera pose and 3-d structure);

- VO focus on 3-d motion of the camera sequentially to recover the full trajectory;

Visual SLAM vs. VO:

- VO only for recovering path incrementally, pose after pose, and potentially optimizing only over the last n poses of the path (windowed bundle adjustment);
- VO only with local consistency of trajectory, and SLAM with global consistency;
- VSAM needs understanding when a loop closure occurs and efficiently integrating this new constraint into the current map;
- VO can be used as a building block for a complete SLAM.

Image-Based Localization Pipeline

- Key Frame and its Camera Pose Stored;
  - Frame id, image feature and camera pose.
- Extract Local Features;
  - Feature original or coded.
- Establish 2D-2D or 3D-to-2D Matches;
  - Single camera: 3d-to-2d;
  - Stereo or depth camera: 2d-to-2d;
- Camera Pose Estimation.
  - Refine by 3-d point estimation;
    - 2-d features along with their depths.

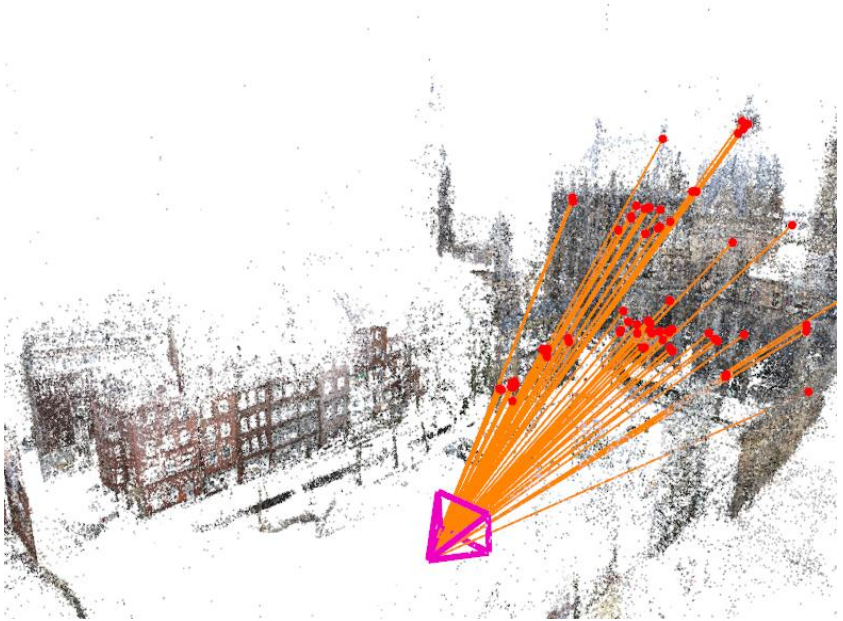
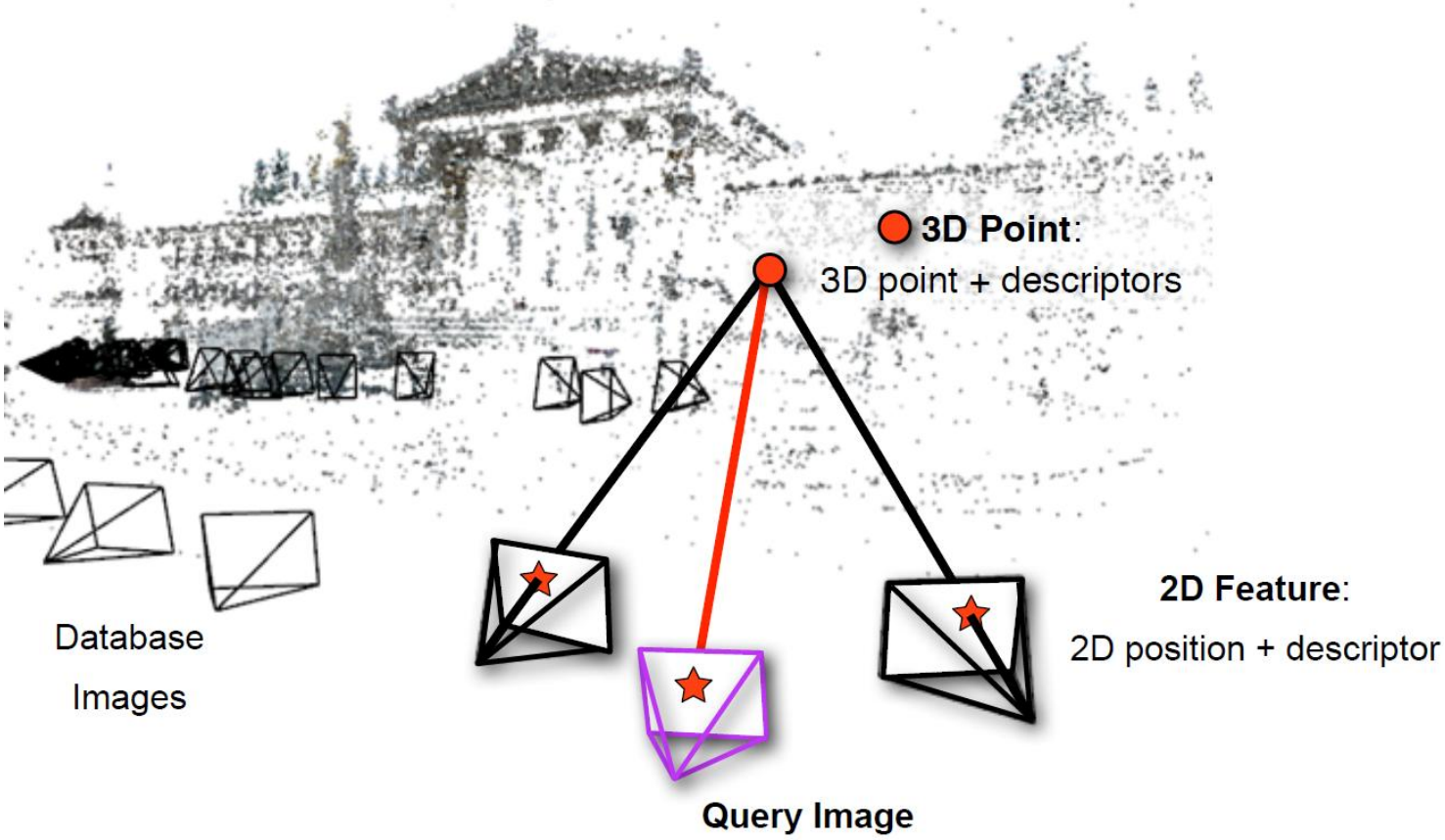


Image-based 3d-to-2d Relocalization



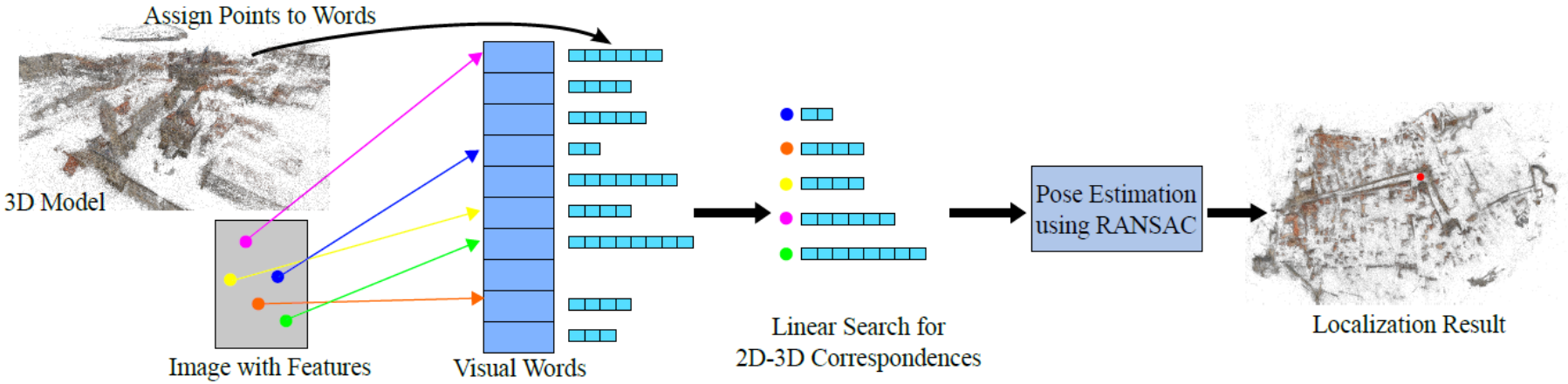
Feature Matching-based Relocalization

- Feature detection and descriptor for key frames or new frames which require relocalization:
  - Harris corner or FAST detection;
  - SURF (64-d vector), SIFT, BRIEF or ORB;
- Feature matching for pose recovery:
  - Brute force or ANN search;
  - Pose difference is small enough.
- Key frame selection:
  - Pose change enough big;

Randomized Tree-based ANN Search for Relocalization

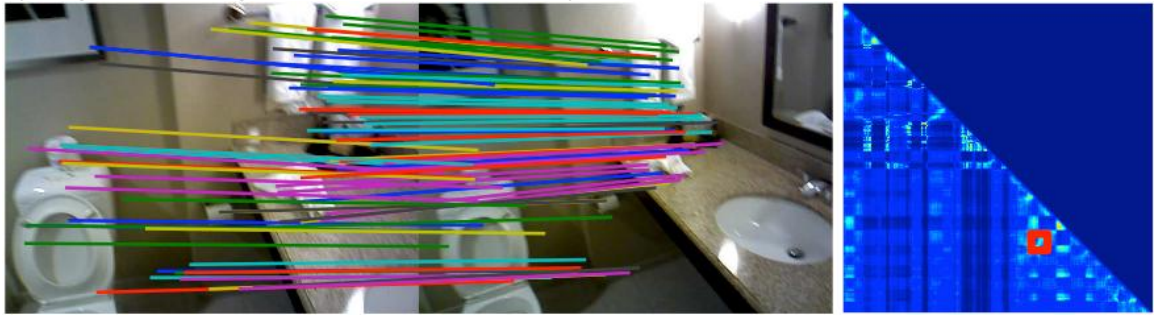
- The kd-tree data structure is based on a recursive subdivision of space into disjoint hyper-rectangular regions called cells;
- ANN (approximate NN) search: Limit number of neighboring k-d tree bins to explore;
- Randomised kd-tree forest: Split by picking from multiple (5) top dimensions with the highest variance in randomized trees, to increase the chances of finding nearby points;
- Best Bin First in K-d tree: a variant of normal K-d tree
  - A branch-and-bound technique that maintains an estimate of the smallest distance from the query point to any of the data points down all of the open paths (priority queue).
- Priority queue-based k-means tree: partition the data at each level into K distinct regions using k-means, then applying it also recursively to the points in each region;
  - The priority queue is sorted in increasing distance from the query point to the boundary of the branch being added to the queue.

Bag of Visual Words for Image-based Relocalization



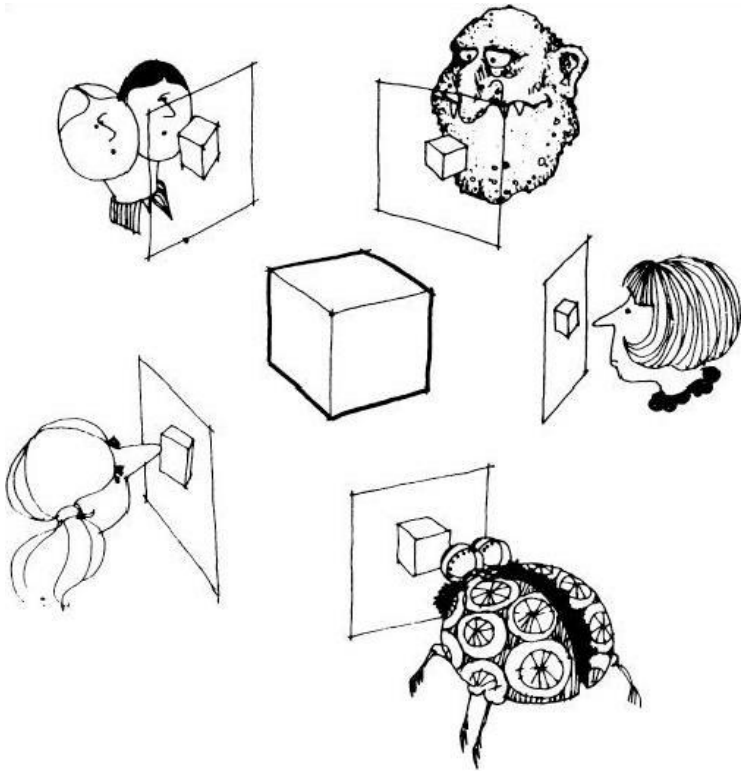
BoVW-based Relocalization in FAB-MAP

- Codebook building from training data:
  - Feature extraction: FAST corner detection + SURF/ORB feature descriptor;
  - Feature clustering for visual vocabulary: incremental K-means;
  - Construct co-occurrence statistics of visual words by Chow Liu tree.
    - A naïve Bayes approximation by a maximum weight spanning tree.
- Visual pattern from key frames or new frames for relocalization:
  - Feature extraction: same as above;
  - Mapping descriptors to bags of words by ANN-based (kd-tree) methods;
- Pattern matching for pose recovery: descriptor mapped to codebook;
  - Sampling in training data or mean field optimization to find the best match.



MVS (Multiple View Stereo)

- Multiple view stereo (MVS): reconstruct a 3D object model from a collection of images taken from known camera viewpoints;
  - 1. Global methods:
    - A volumetric reconstruction to extract a surface;
    - Surface evolving iteratively, like space carving;
  - 2. Local methods:
    - Depth map merging with consistency constraints;
    - Sparse feature matching, region expansion and dense reconstruction for surface fitting.

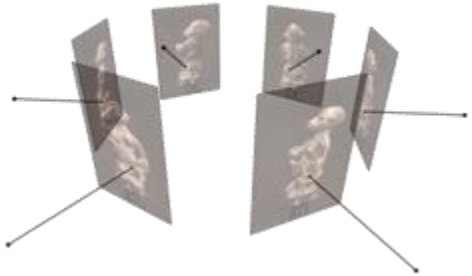


Multiple View Stereo

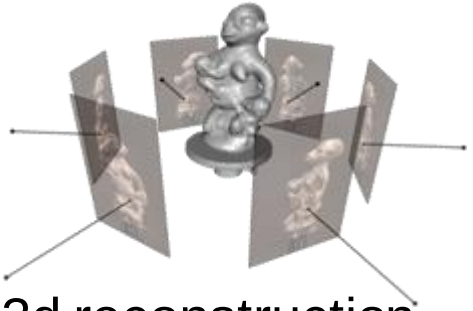
Properties	Global MVS	Local MVS
Robustness to surface curvature	Low	High
Accuracy on ideal dataset	Low	High
Parameter management	Not easy	Comparable easy
Robustness to lack of texture	High	Low
Robustness to unrecorded surface	High	Low
Guarantee a watertight mesh	Easy	Difficult

MVS Pipeline

Image acquisition

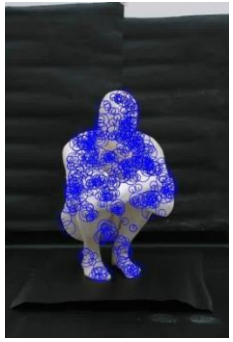


Camera pose

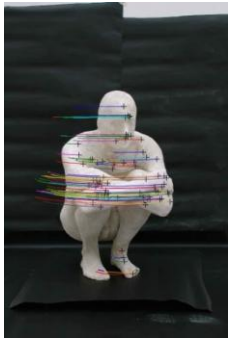


3d reconstruction

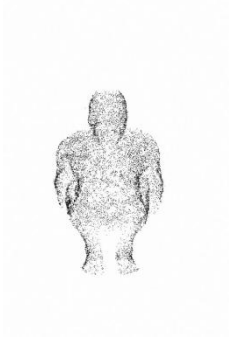
Input sequence



2d features



2d track

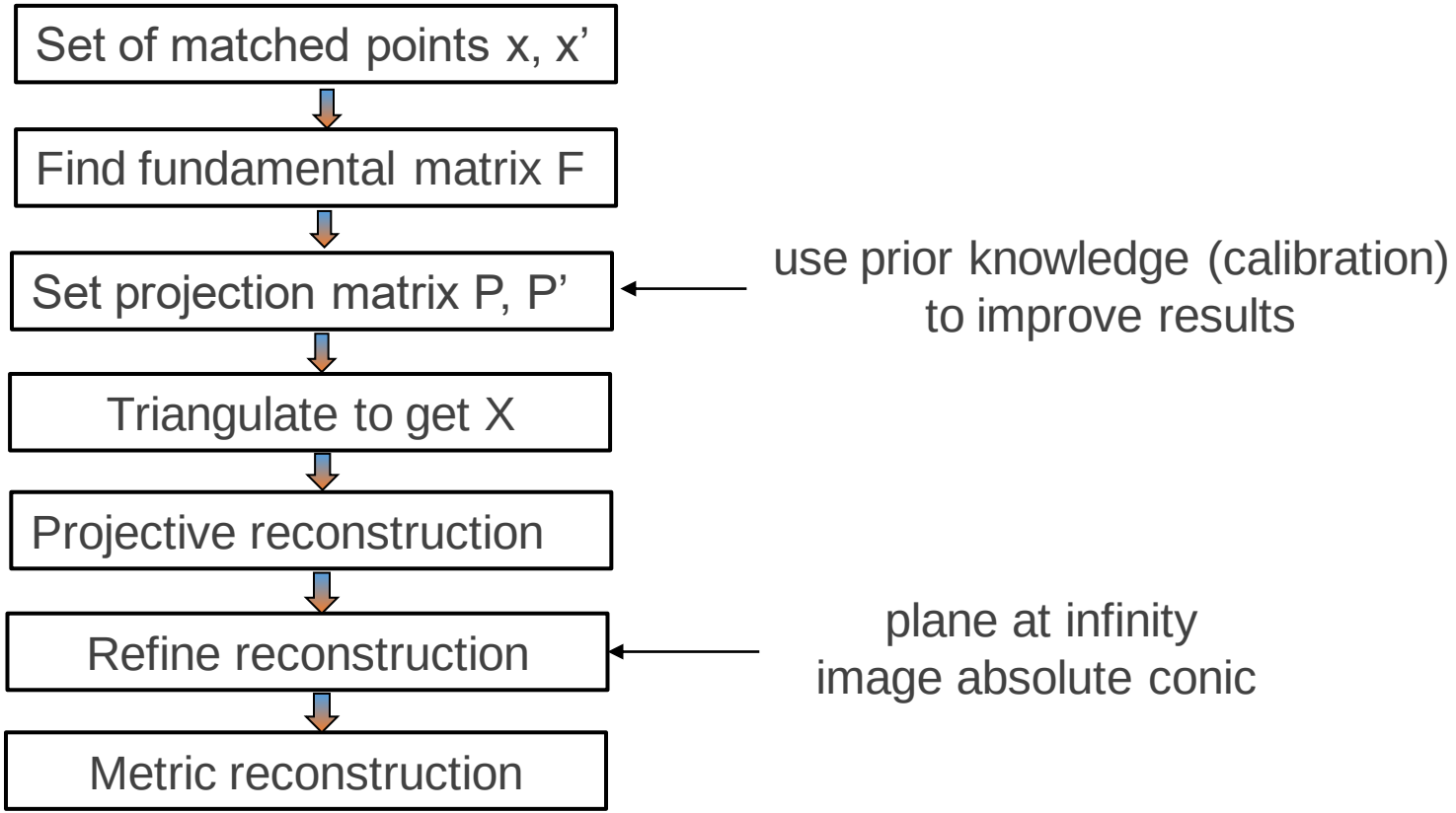


3d points

Stratified Reconstruction

- With no constraints on calibration matrix or the scene, *projective* reconstruction;
  - Preserves intersection and tangency
- Need additional information to *upgrade* the reconstruction: Affine;
  - Preserves parallellism, volume ratios
- Then, similarity;
  - Preserves angles, ratios of length
- Finally Euclidean.
  - Preserves angles, lengths

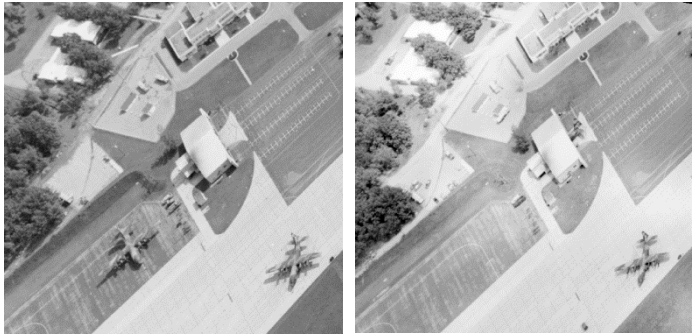
Stratified Reconstruction

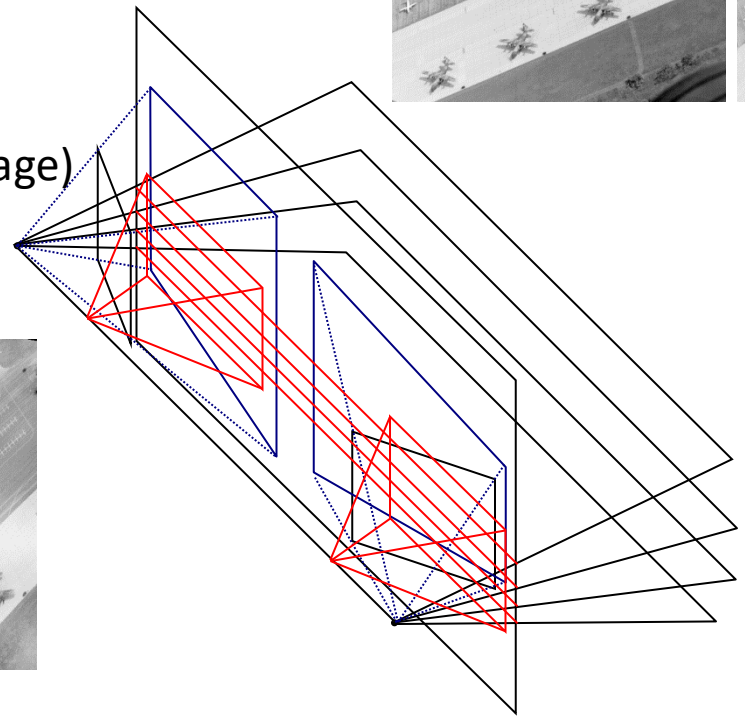


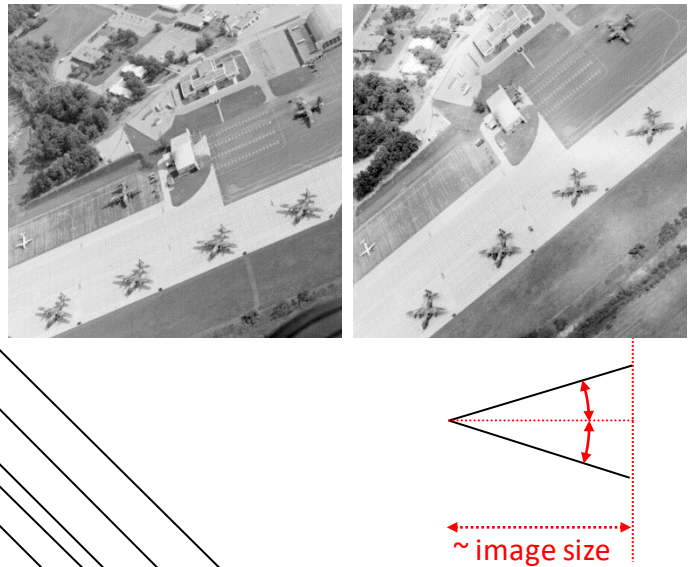
Planar Rectification

Bring two views
to standard stereo setup

(moves epipole to ∞)
(not possible when in/close to image)





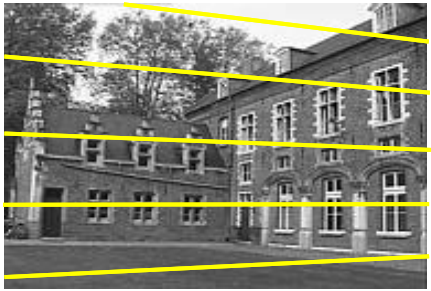


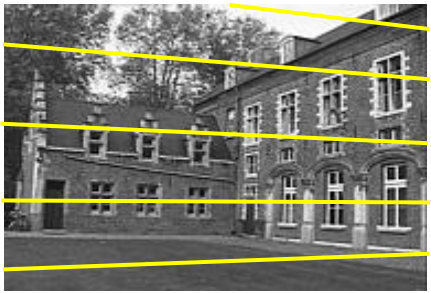
(calibrated)

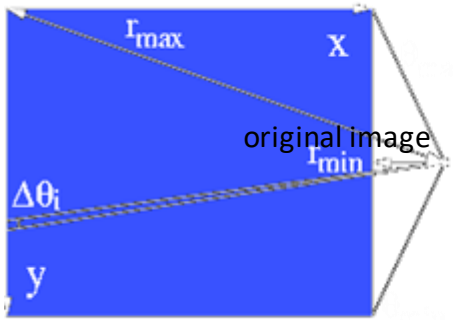
Distortion minimization
(uncalibrated)

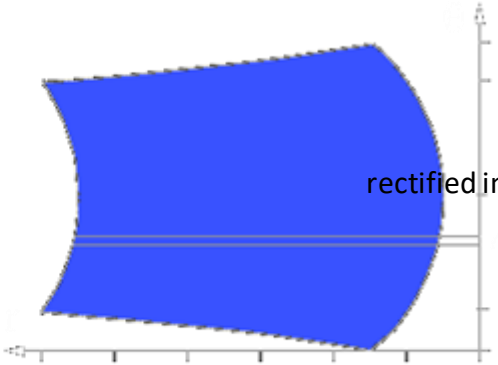
Polar Rectification

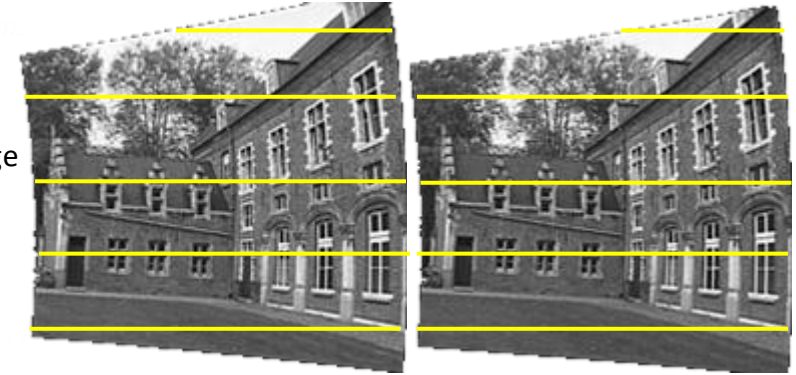
Polar re-parameterization around epipoles
Requires only (oriented) epipolar geometry
Preserve length of epipolar lines
Choose $\Delta\theta$ so that no pixels are compressed











Works for all relative motions
Guarantees minimal image size

Stereo Matching

- Feature-based (sparse) and correlation-based (dense);

image I(x,y)

Disparity map D(x,y)

image I´(x´,y´)

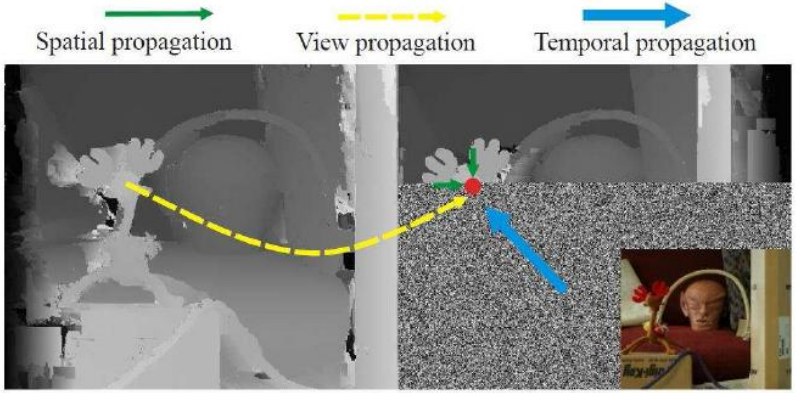
(x´,y´)=(x+D(x,y),y)

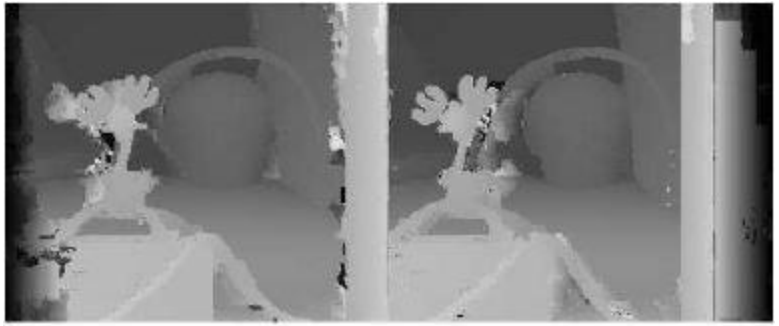
- Constraints
  - epipolar
  - ordering
  - uniqueness
  - disparity limit

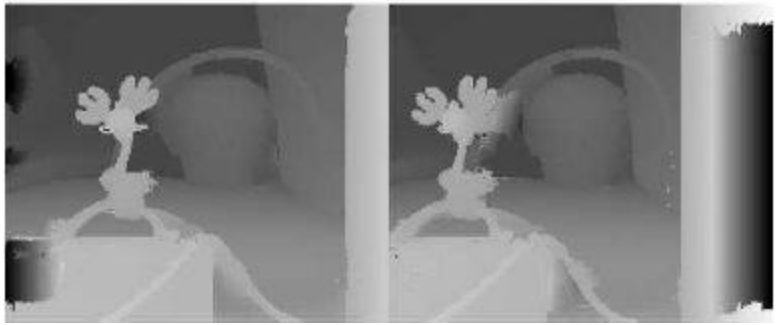
PatchMatch Stereo

- Idea: First a random initialization of disparities and plane para.s for each pix. and update the estimates by propagating info. from the neighboring pix.s;
  - **Spatial propagation**: Check for each pix. the disparities and plane para.s for left and upper neighbors and replace the current estimates if matching costs are smaller;
  - **View propagation**: Warp the point in the other view and check the corresponding etimates in the other image. Replace if the matching costs are lower;
  - **Temporal propagation**: Propagate the information analogously by considering the etimates for the same pixel at the preceding and consecutive video frame;
  - **Plane refinement**: Disparity and plane para.s for each pix. refined by generat. random samples within an interval and updat. estimates if matching costs reduced;
  - **Post-processing**: Remove outliers with left/right consistency checking and weighted median filter; Gaps are filled by propagating information from the neighborhood.

PatchMatch Stereo





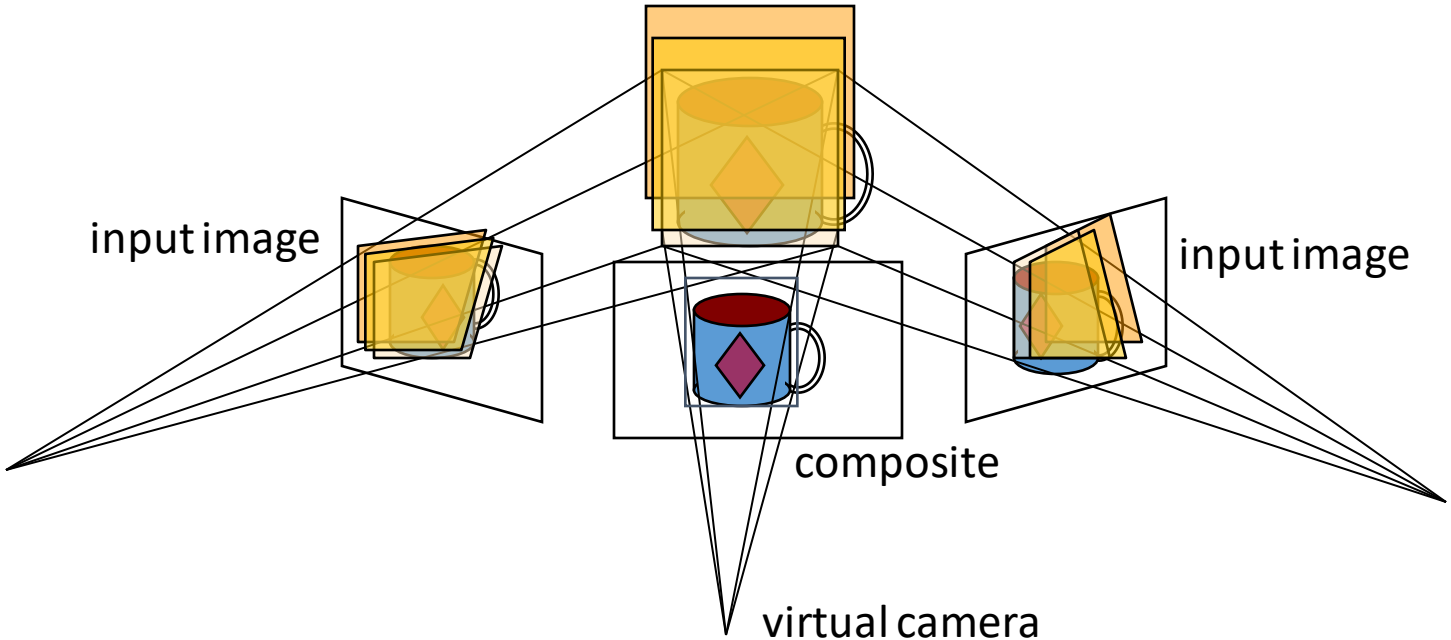


Model

- For each pixel p find a plane $f_p = (a_{f_p}, b_{f_p}, c_{f_p})$ reflecting the local surface orientation
- The disparity at $p = (p_x, p_y)$ can be computed as $d_p = a_{f_p} p_x + b_{f_p} p_y + c_{f_p}$
- The plane f_p is the one of minimum aggregated matching costs among all possible planes $f_p = \arg\min_{f \in \mathcal{F}} m(p, f)$
- The aggregated matching function is defined as $m(p, f) = \sum_{q \in W_p} w(p, q).\rho(q, q - (a_{f_p} p_x + b_{f_p} p_y + c_{f_p}))$, where $\rho(q, q')$ measures the pixel dissimilarity between q and q'

Plane Sweep Stereo

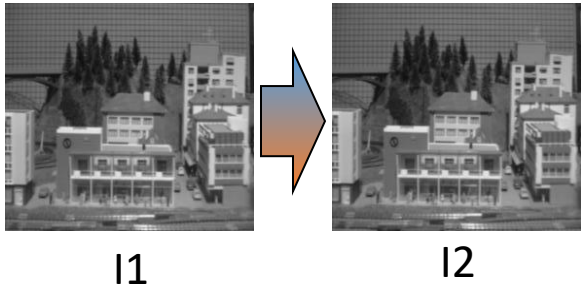
- Choose a virtual view
- Sweep family of planes at different depths w.r.t the virtual camera

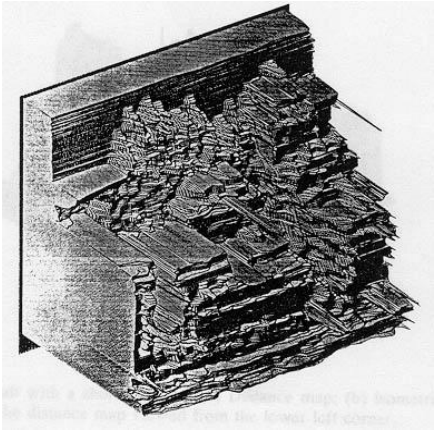


each plane defines an image $\Rightarrow$ composite homography

Multiple-baseline Stereo

- For a reference image, and along the corresponding epipolar lines of all other images, using **inverse depth** relative to the first as the search parameter;
- For larger baselines, search larger area;
- Pros
  - Using multiple images reduces the ambiguity of matching
- Cons
  - Must choose a reference view;
  - Occlusions become an issue for large baseline.
- Possible solution: use a **virtual view**



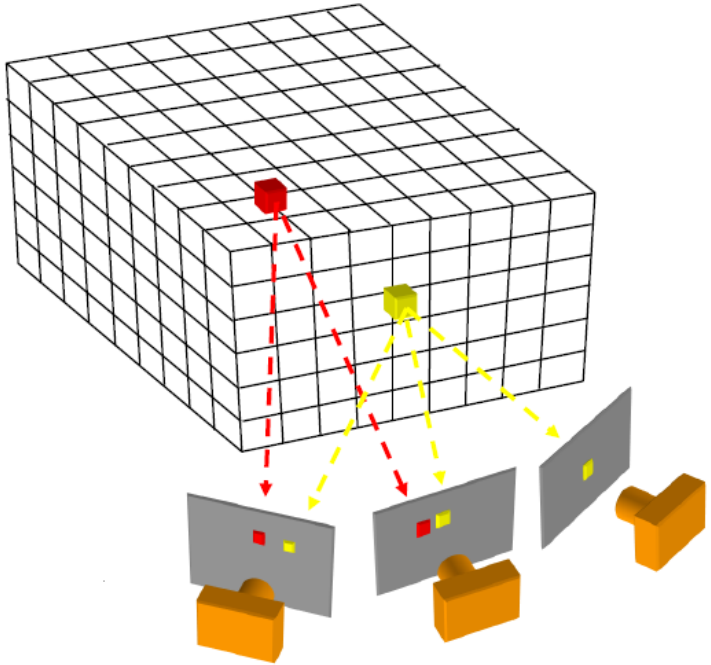


Volumetric Stereo/Voxel Coloring

- Use a voxel volume to get a view-independent representation;
- Goal: Assign RGB values to voxels in V *photo-consistent* with images

Discretized Scene Volume

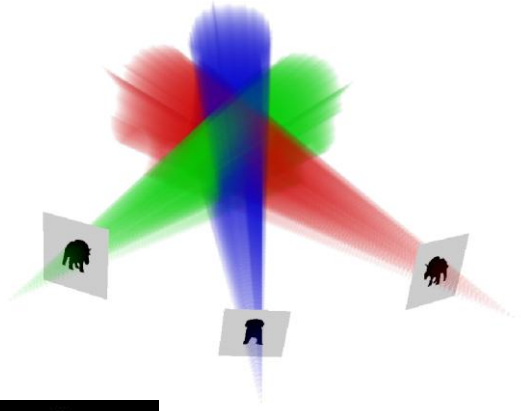
Input Images(Calibrated)



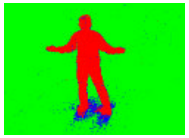
Space Carving

- Initialize to a volume V containing the true scene;
  - Bounding box volume subdivided into voxels
  - Voxel color hypotheses collected from all visible images
  - Elimination of inconsistent hypotheses
- Repeat until convergence (slow);
  - Choose a voxel on the current surface
  - Project to visible input images
  - Carve if not photo-consistent

Visual Hull



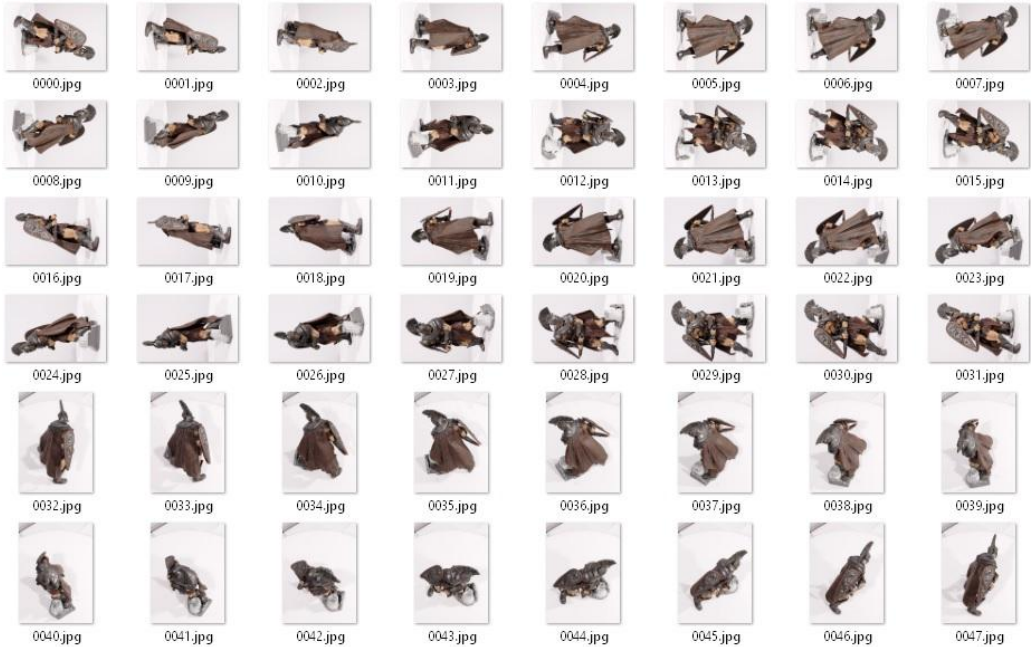
- Shape-from-silhouette:
  - Silhouette defines generalized cone;
- Pre-step: silhouette segmentation
  - Background subtraction
  - Tracking
  - Active illumination



- Visual hull reconstruction
  - Voxel-based
  - Polyhedral
  - Image-based visual hull rendering

Find intersection of silhouettes
– Divide volume into voxels
– Cut voxels outside any silhouette

Visual Hull - An Example

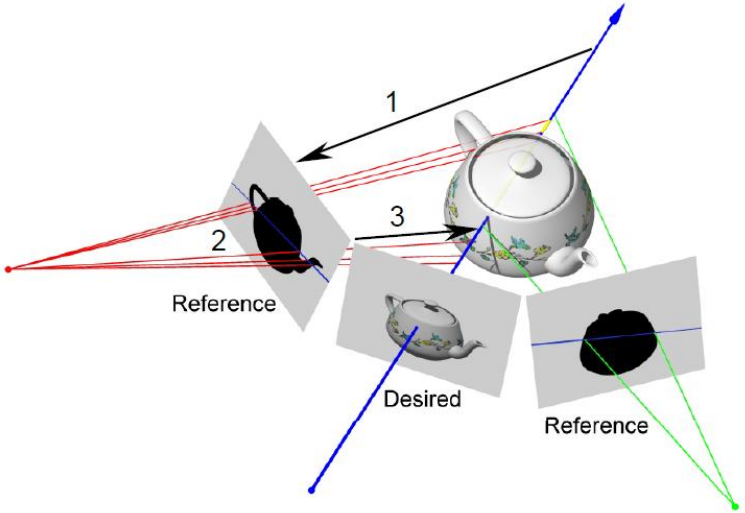


Visual Hull

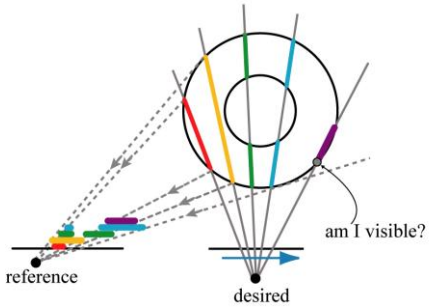
- Polygon-based visual hull
  - Silhouette contours approximated as 2D polygons
    - Precision adjustment: variable # polygon vertices
  - Contours define polyhedral 3D cones
  - Intersection of cones: Polygonal *Visual Hull*

Visual Hull

- Image-based visual hull
  - From all silhouettes
    - Find tightest depth range for each pixel
  - From closest image
    - Determine pixel color from near end of depth range
  - Visibility check:
    - Project all pixels' depth ranges into reference image
    - Built z-buffer in reference image plane
    - Desired pixel location on top.
  - Implicit depth.
  - Issues:
    - Background manually got.



1) the desired ray is projected onto a reference image;
2) the intervals where the ray crosses the silhouette are determined;
3) these intervals are lifted back onto the desired ray where they can be intersected with intervals from other reference images.

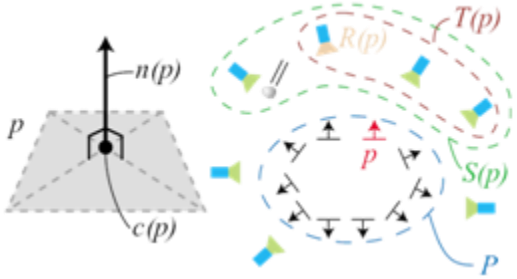


Carved Visual Hulls

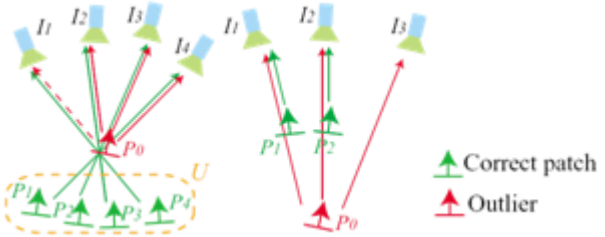
- Compute visual hull
- Use dynamic programming to find rims and constrain them to be fixed
- Carve the visual hull to optimize photo-consistency

Region Growing for MVS

1. Fitting step
   A local surface patch is fitted, iterating visibility



2. Filter step
   Visibility is explicitly enforced



3. Expand step
   Successful patches are used to initialise active boundary

Depth-Map Fusion for MVS

- Compute depth hypotheses from each view using neighboring views
  - Compute completely un-regularised depth-maps independently for each viewpoint;
- Merge depth-maps into single volume with use of redundancy
  - Compute one correlation curve per image, then find its local maximum
  - Utilize the pixel neighborhood in a 2D discrete MRF: peaks as hypothesis and “unknown depth” label against spatially inconsistent peaks;
  - Model sensor probabilistically as a Gaussian + Uniform mixture, evolution of estimates in sequential inference.
- Extract 3d surface from this volume
  - Regularise at final stage;
  - Graph cut-based segmentation with photo consistency;
    - Surface extracted by marching cubes from binary map.

Comparison of Region Growing and Depth Map Fusion

	Region growing	Depth-map fusion
summary	Starts from a cloud of 3d points, and grows small flat patches maximizing photo-consistency	Fuses a set of depth-maps computed using occlusion-robust photo-consistency
pros	Provides best overall results due to a plane-based photo-consistency	Elegant pipeline; Plug-n-play blocks; Easily parallelizable.
cons	Many tunable parameters, i.e., difficult to tune to get the optimal results	Photo-consistency metric is simple, but not optimal; The metric suffers when images are not well textured or low resolution

Patch-based MVS

- Feature extraction, matching, patch expansion/propagation, filtering with visibility consistency;
- Works well for various objects and scenes
- Surfaces must be Lambertian and well-textured
- Problematic for architectural scenes
- Limitation of MVS: make use of planarity and orthogonality, geometric priors for interpolation
  - Manhattan world stereo (piecewise axis-aligned planar scenes)?

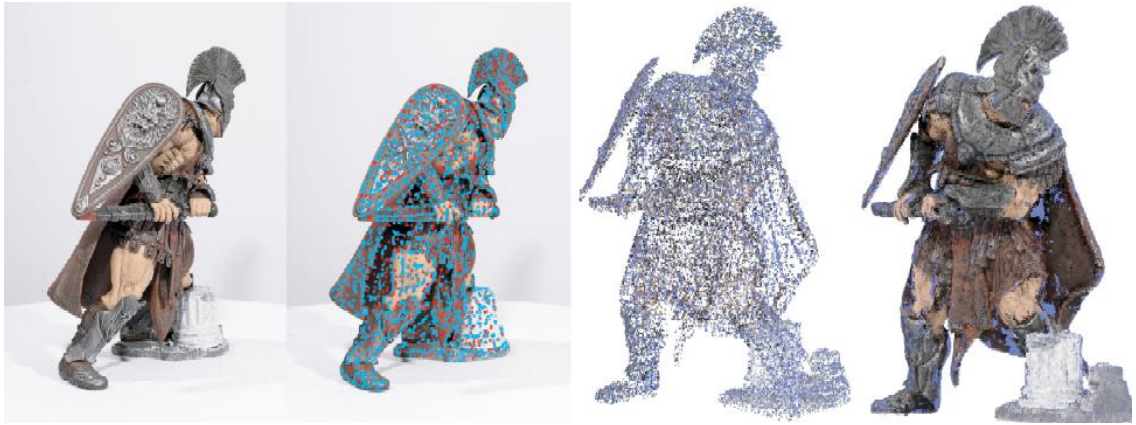
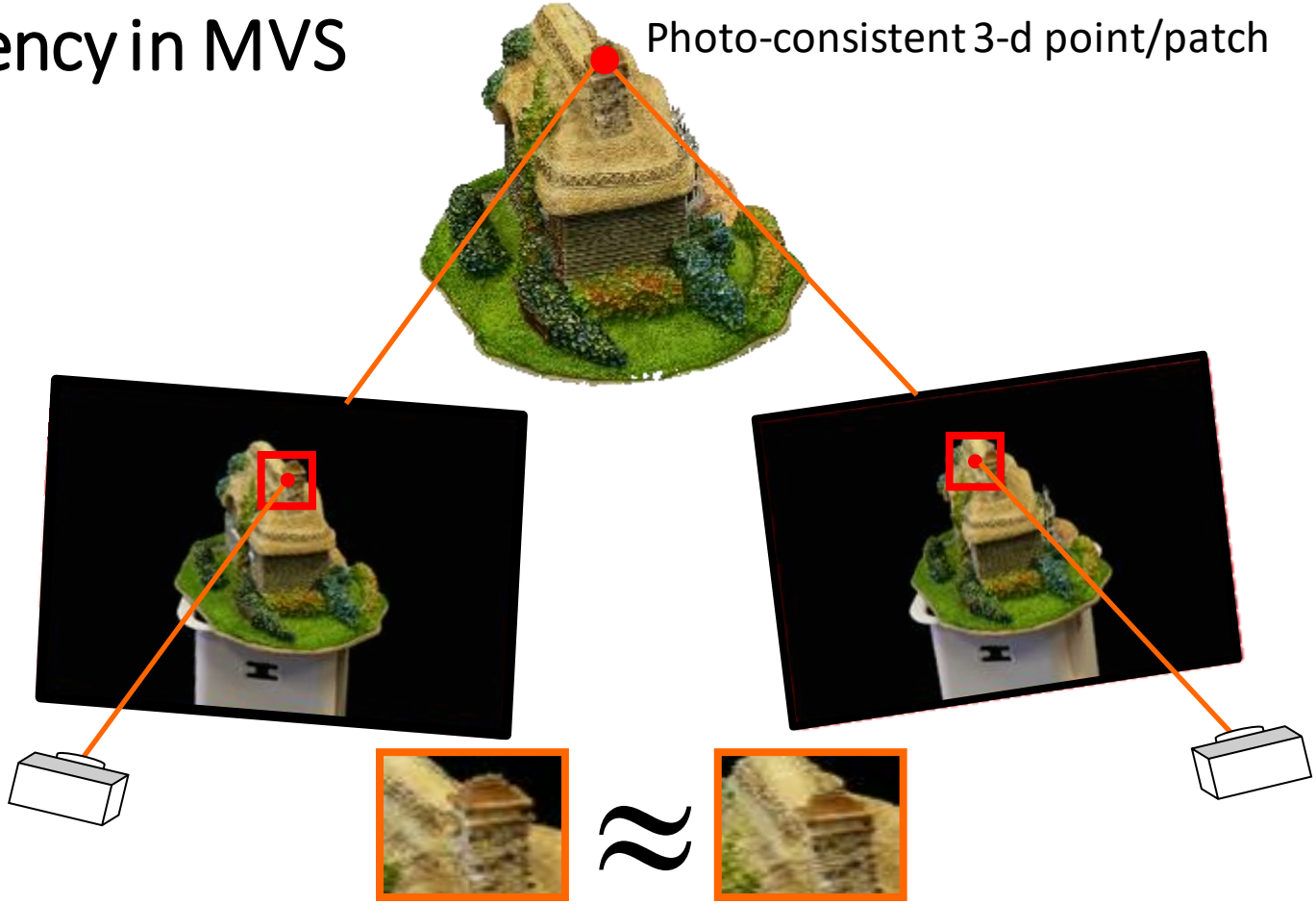


Photo-Consistency in MVS

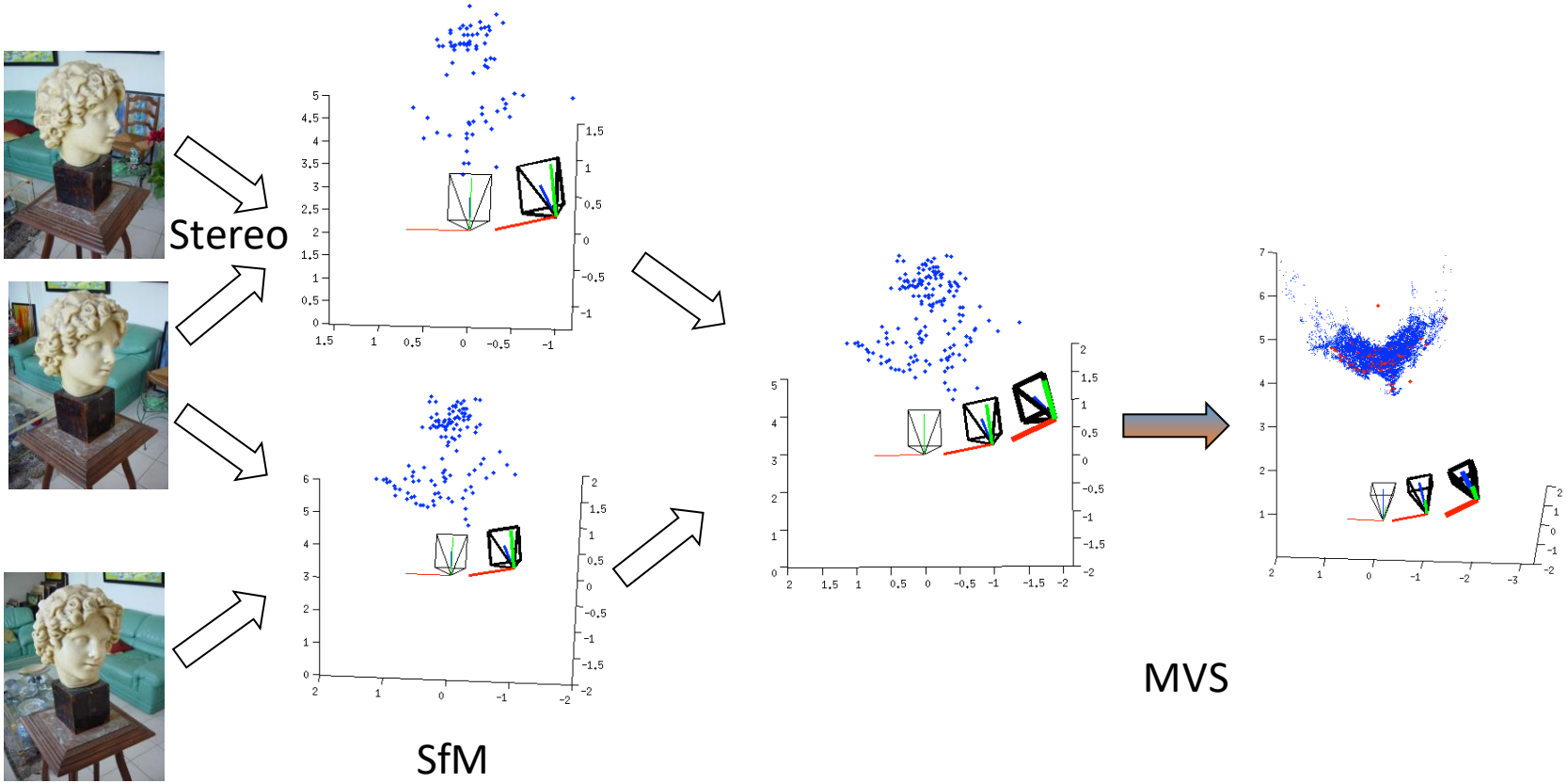
- A *photo-consistent scene* is a scene that exactly reproduces input images from the same camera viewpoints;
- Can't use input cameras and images to tell the difference between a photo-consistent scene and the true scene.



Regularization for Textureless Regions in MVS

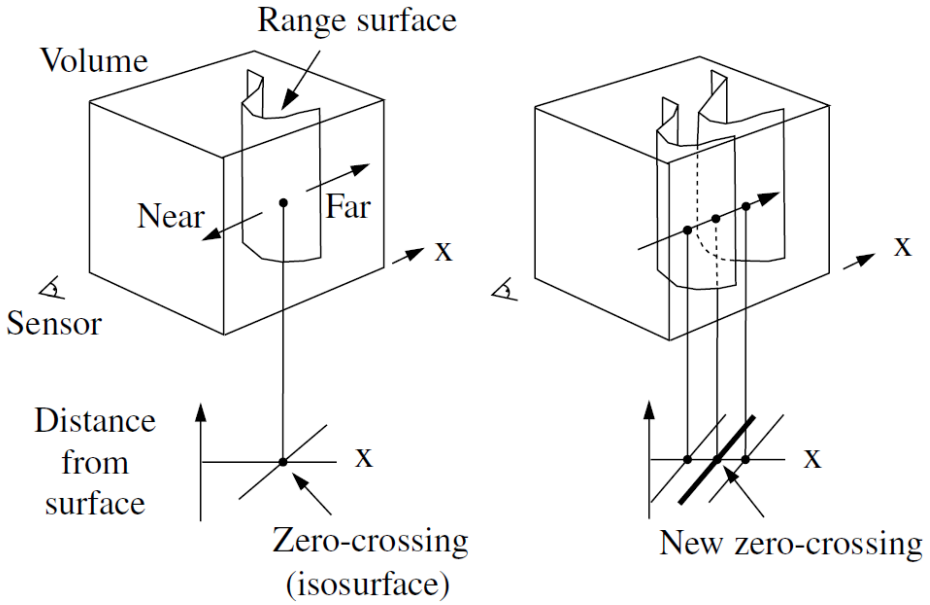
- More information is needed, besides of photo-consistency.
  - User provides a priori knowledge.
  - Classical assumption: Objects are “smooth.”
  - Also know as *regularizing the problem*.
- Optimization problem:
  - Find the “best” smooth consistent object.
- Popular used methods:
  - Level set;
  - Snakes;
  - Graph cut;
  - Exploiting silhouettes.

Initial Stereo, Camera Pose by SfM and Depth by MVS



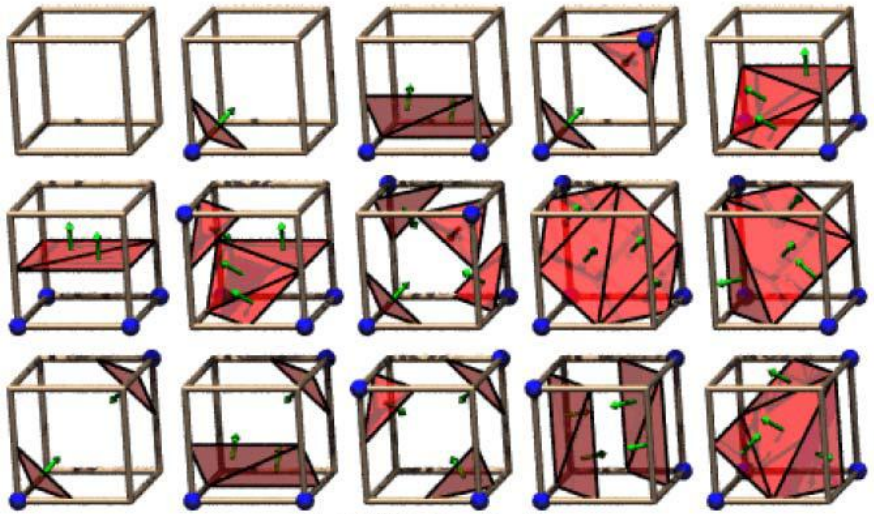
Signed Distance Function for Surface Representation

- Volumetric integration;
- Curless &Levoy (1996)'s method for fusing depth maps into a global surface using the SDF representation;
- Importantly the minimiser of L2 cost is average of the SDF measurements which can be computed recursively, enabling massive data fusion;
- Samples also weighted to take into account uncertainty of each data.



Marching Cubes

- Transform volume data sets to a surface model
- Use polygon rendering methods
- Use polygon manipulation methods
  - Deformation
  - Morphing
- Approximate iso-surface with a triangle mesh gen. by comparing iso-value and voxel densities;
- Look at each cell of volume 1-by-1, generate triangles inside cell by configuration of voxel values;



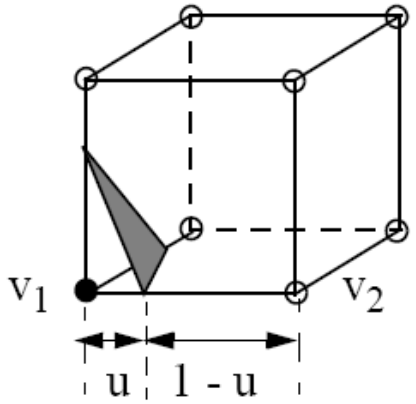
The 15 Cube Combinations

Marching Cubes

- Imagine all voxels above the iso-value are 1 and others are 0;
- Find triangular surface includes all 1s and excludes all 0s;
- Each of the 8 cornels of one cell can be 0 or 1, totally we have 28 =256 cases;
- Can be reduced to 15 cases use symmetry.
- Exact positions of triangle vertices determined by linear interpolation of the isovalue;

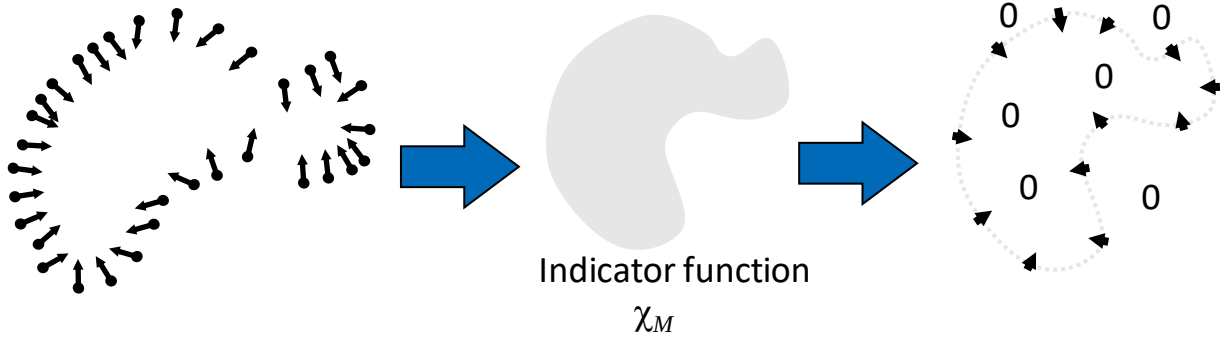
$$iso = v_1 \cdot (1-u) + v_2 \cdot u \longrightarrow u = \frac{v_1 - iso}{v_1 - v_2}$$

- Generally can be fast computed by a Lookup table
- Have some ambiguous cases



Poisson Surface Reconstruction

- Reconstruct surface of model by solving for indicator function of the shape;



Oriented points

Indicator function χ_M

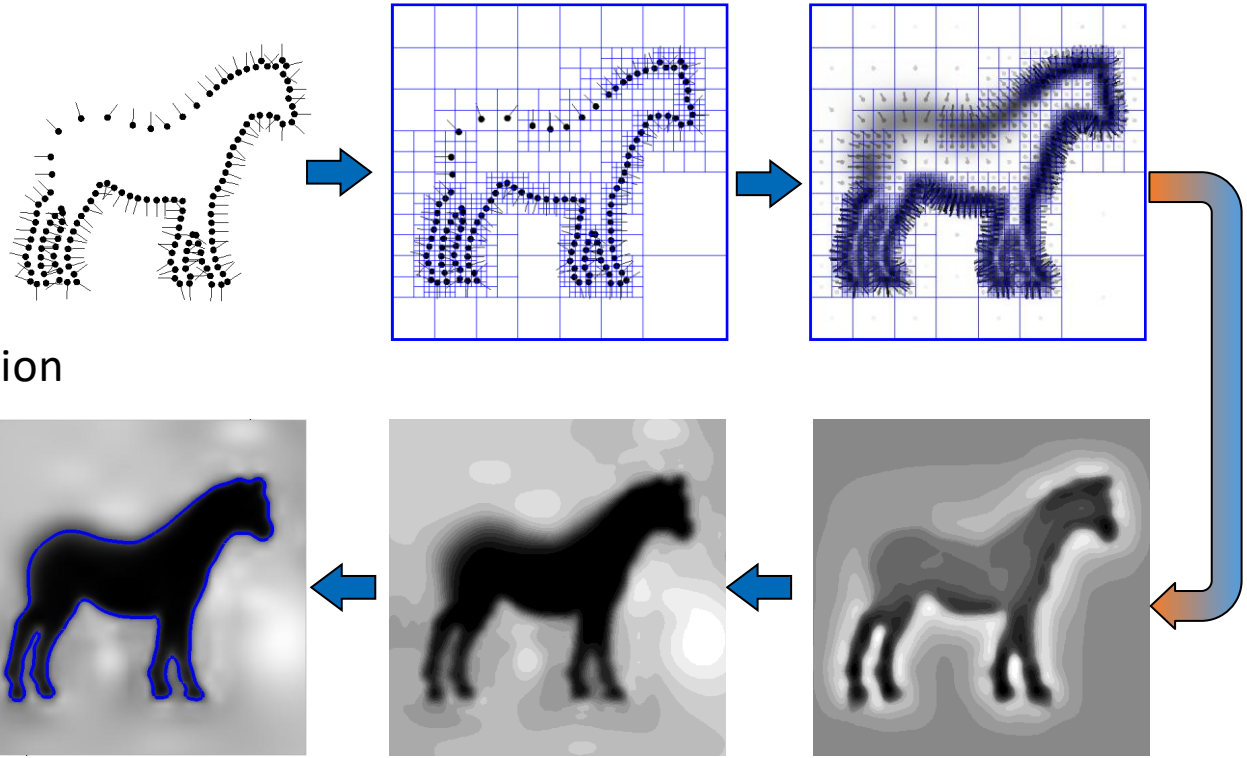
$$\chi_M(p) = \begin{cases} 1 & \text{if } p \in M \\ 0 & \text{if } p \notin M \end{cases}$$

- There is a relationship btw the normal field and gradient of indicator function;
- Integration: Represent points by a vector field $\vec{V}$, and find function χ whose gradient best approximates $\vec{V}$: $\min_{\chi} \left\| \nabla\chi - \vec{V} \right\|$
- Applying divergence operator, we transform this into a Poisson problem:

$$\nabla \cdot (\nabla\chi) = \nabla \cdot \vec{V} \quad \Leftrightarrow \quad \Delta\chi = \nabla \cdot \vec{V}$$

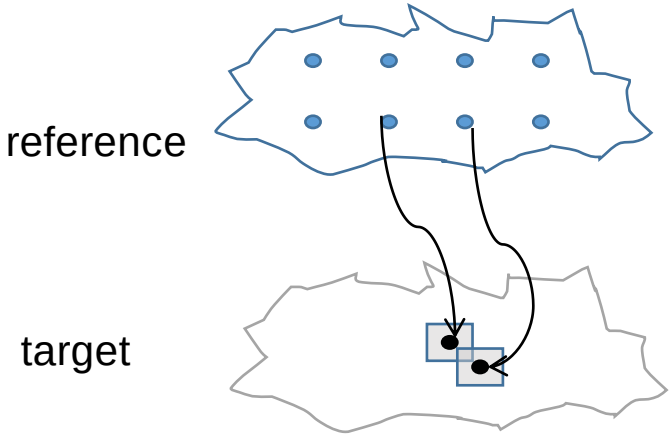
Poisson Surface Reconstruction

- Given the Points:
  - Set octree
  - Compute vector field
    - Define a function space
    - Splat the samples
  - Compute indicator function
    - Compute divergence
    - Solve Poisson equation
  - Extract iso-surface



Surface Rendering

- Splatting (pixel-based) [Mark & Bishop 98]
  - Splat: the pixel shape projected to reconstruct the new view in the image space.
  - Footprint: reconstruction kernel.
- A depth map represents a cloud of 3d points to be splatted.
- Splatting broaden the individual 3d points.



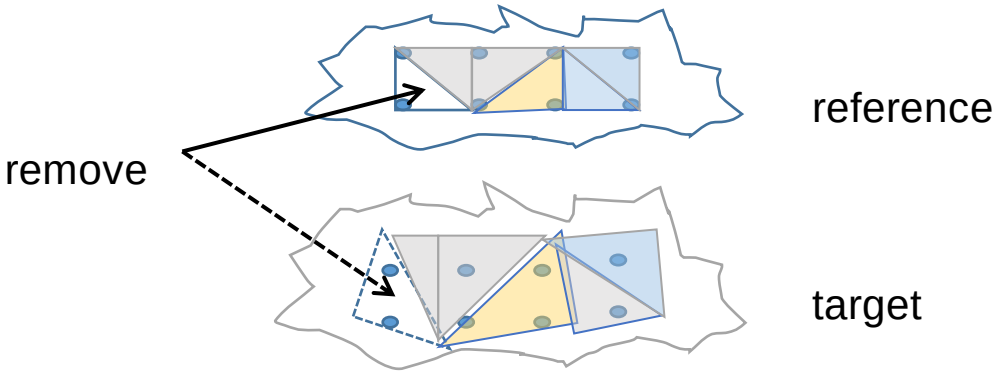
Triangulation (mesh-based) [Kanade et al. 97]

- Pixel plus depth as vertex of a triangular mesh
- Interpolation within meshes.

Feature-based reconstruction

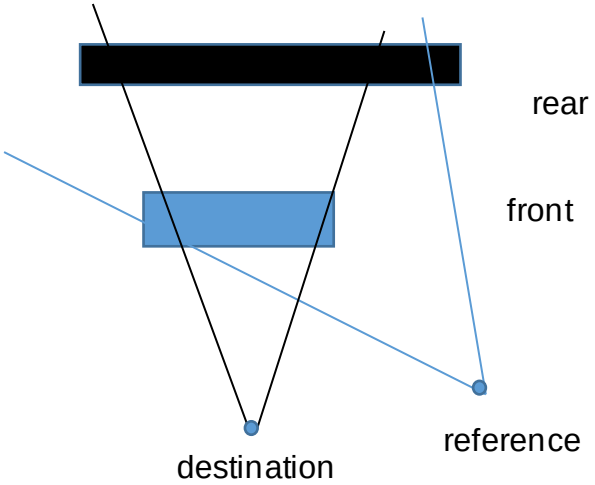
Depth discontinuity (phantom/rubber sheet)

- Breaking all edges with large difference
- Removing those corresponding triangles.



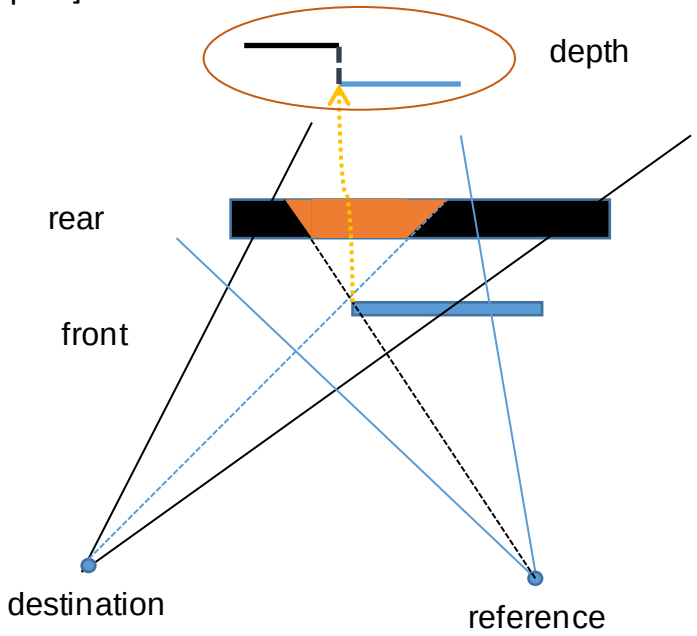
Folding/Visibility Problem

- An occlusion issue: overlapped from multiple projections.
- An alternative: mapping the pixels in a back-to-front specific order [McMillan & Bishop 95].
  - Unavailability for rendering with multiple reference images.
- Z-buffering [Chen & Williams 93]
  - Winner-take-all;
  - Soft Z-buffering: top k values (k=3~4) are blended.
  - Multiple views: the views close to the new view get more emphasized than those views away from the new view (angular distance).

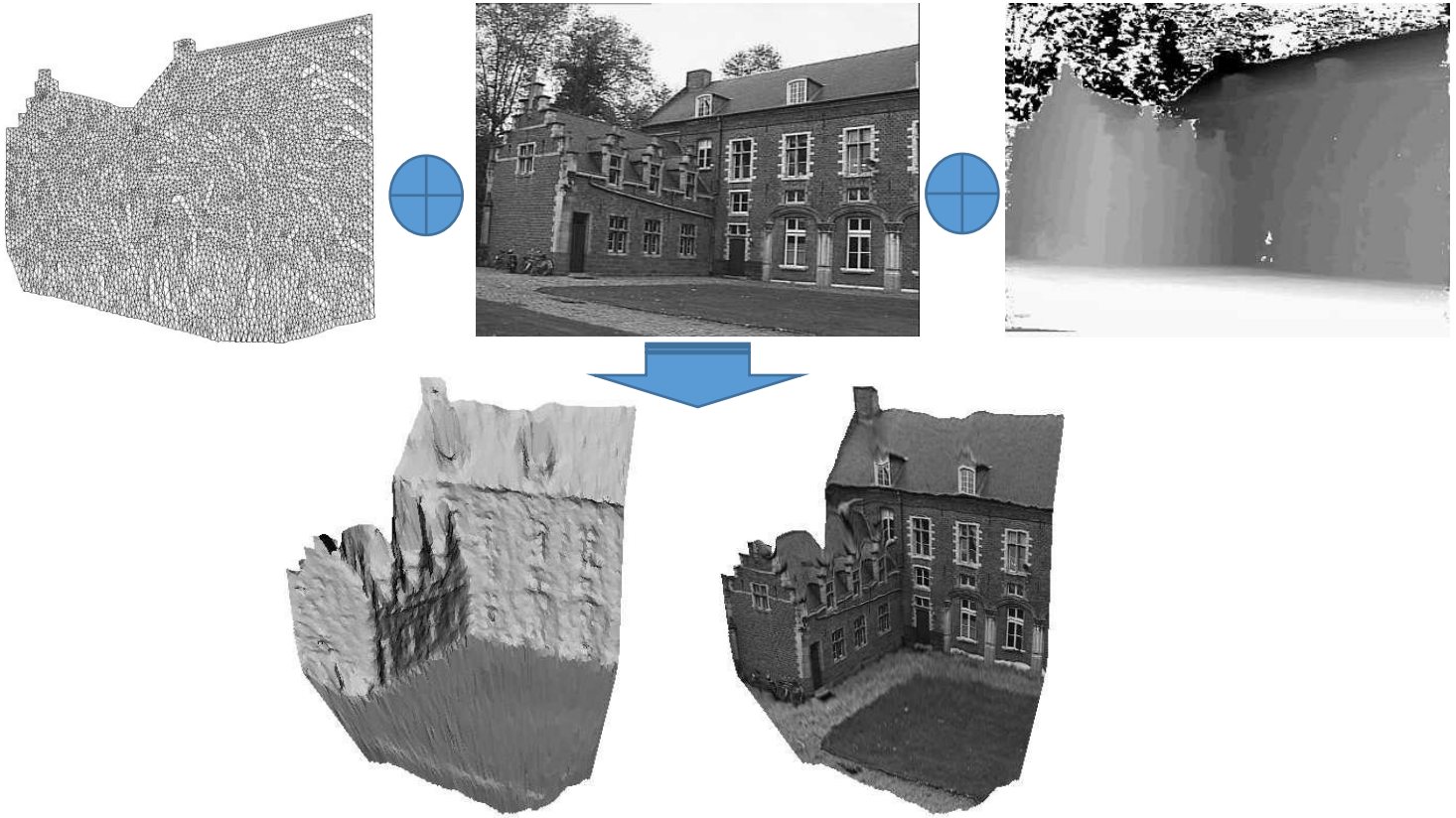


Hole-Filling Problem

- A disocclusion issue: holes by depth discontinuities at the boundary between FG and BG.
- Popular methods:
  - Searching backwards along the epipolar lines [McMillan & Bishop 95]
    - Complicated for non-convex foreground objects ;
  - Inpainting or texture synthesis [Tauber et al 07];
    - Risk of filling the holes with foreground textures -> depth-aware;
  - Layered depth image (LDI) [Shade, Gotler & Szeliski 98]:
    - How to generate a compact LDI is a segmentation task;
  - Low-pass filtering of the depth image:
    - Prone to geometric distortions[Fehn04];
- Layered representation:
  - Boundary matting [Zitnick et al. 04]
    - Alpha matting for boundary layer ;
  - Background layer [Criminisi et al. 07]:
    - Bi-modal disparity histogram-based segmentation.

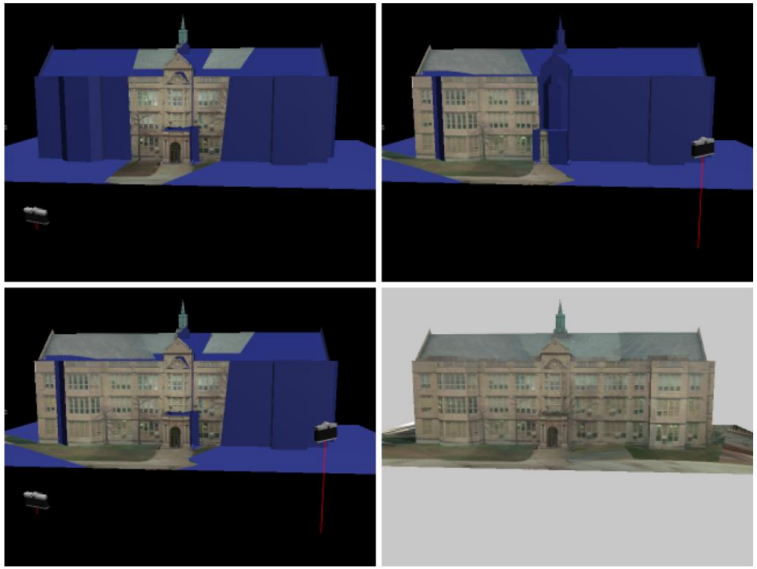


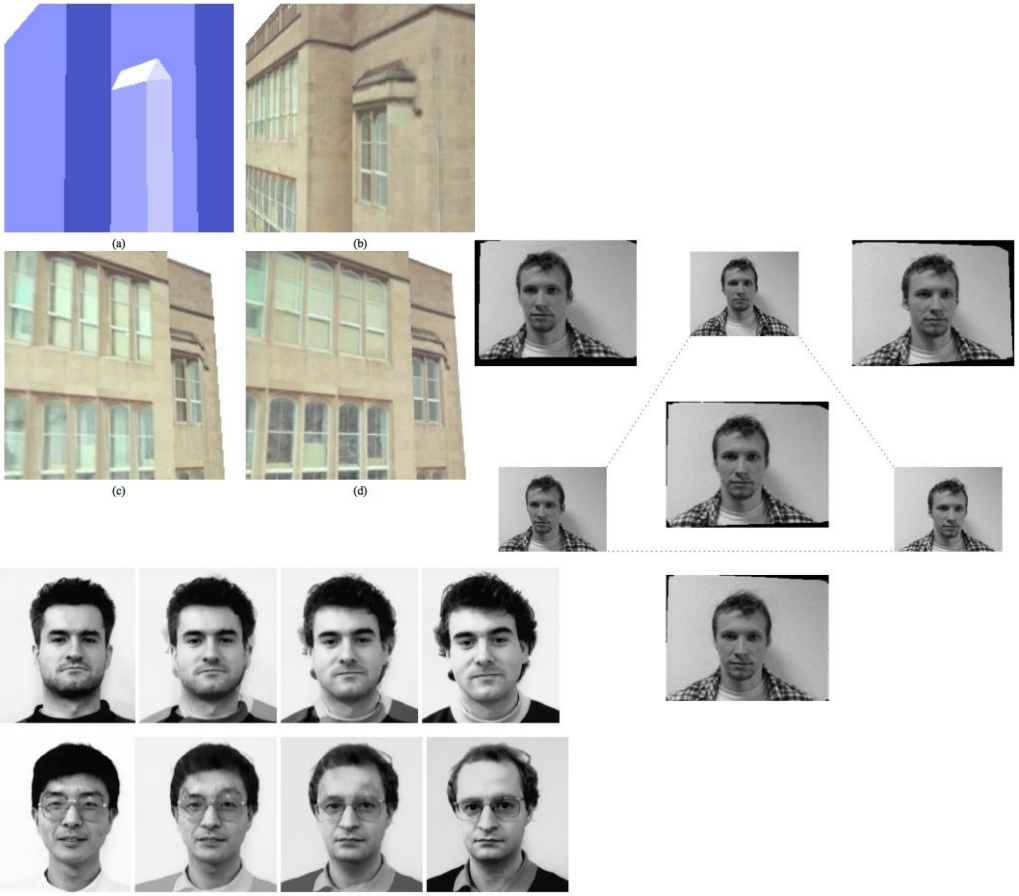
Texture Mapping



Texture Mapping

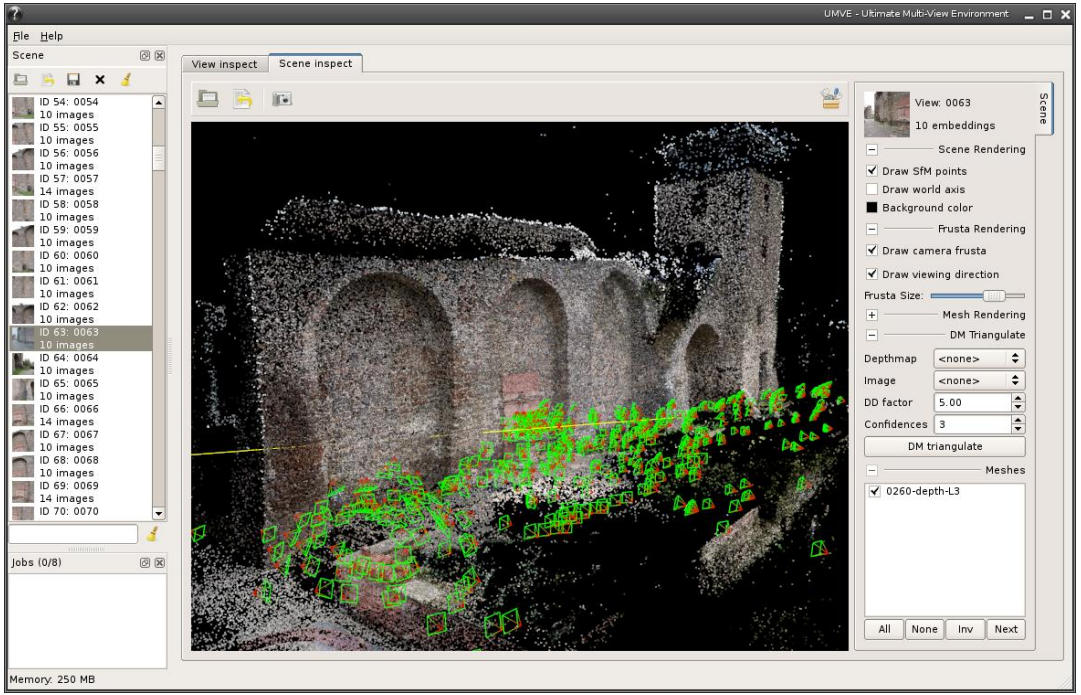
- Texture stitching
- View-based texture mapping
- View interpolation and warping





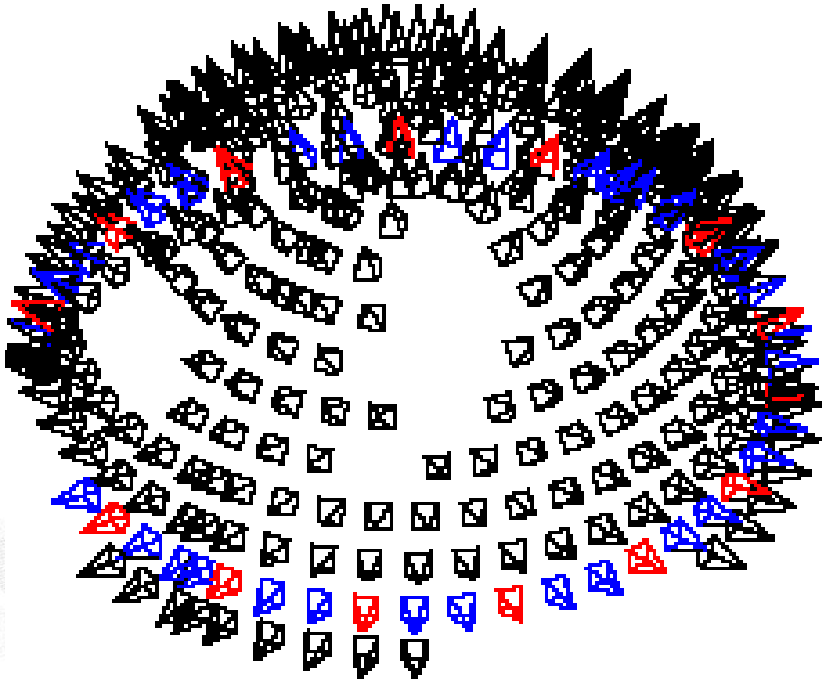
MVE: Multi-View Reconstruction Environment

- A complete end-to-end pipeline for image-based geometry reconstruction:
  - Structure-from-Motion;
  - Multi-View Stereo;
  - Surface Reconstruction.
- Open MVG library;
- Open Bundler: SfM;
- Dense depth map estimation:
  - Get from uncontrolled images;
- MVS methods:
  - Region growing, depth merging.



Evaluation of MVS' Performance

- Evaluation @vision.middlebury.edu;
- DTU Robot Image Data Sets;



Appendix:

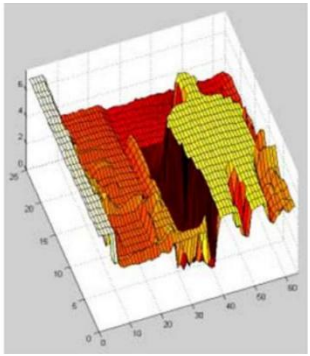
Depth Sensor-based Reconstruction

Outline

- Time of Flight
- Structured Light
- KinectFusion
- Combination of Feature + ICP
- RGB-D Visual Odometry (VO)
- Kintinous: Extended KinectFusion
- Combination of FOVIS & RGB-D VO
- OminiKinect
- Shake'n' Sense
- KinFuSeg
- SLAM++: Object level
- Inertial KinectFusion
- Scalable Volumetric Surface Reconstruction
- Patch Volume: Segmentation-based Consistent Mapping
- Dense Scene Reconstruction with Points of Interest
- Points-based fusion
- 3D Reconstruction at Scale using Voxel Hashing
- DynamicFusion
- RGB-D Camera Relocalization

Time-of-Flight (ToF)

Perform the depth quantifying the changes that an emitted light signal encounters when it bounces back from objects in a scene;



Advantages:

- Only one camera required
- Acquisition of 3D scene geometry in real-time
- Reduced dependence on scene illumination
- Almost no dependence on surface texturing

Pulsed Modulation: Measure distance to a 3D object by measuring the absolute time a light pulse needs to travel from a source into the 3D scene and back, after reflection;

- Speed of light is constant and known;

Continuous Wave Modulation: Continuous light waves instead of short light pulses; Modulation in terms of frequency of sinusoidal waves; Detected wave after reflection has shifted phase; Phase shift proportional to distance from reflecting surface.

- Retrieve phase shift by demodulation of received signal
- Demodulation by cross-correlation of received signal with emitted signal

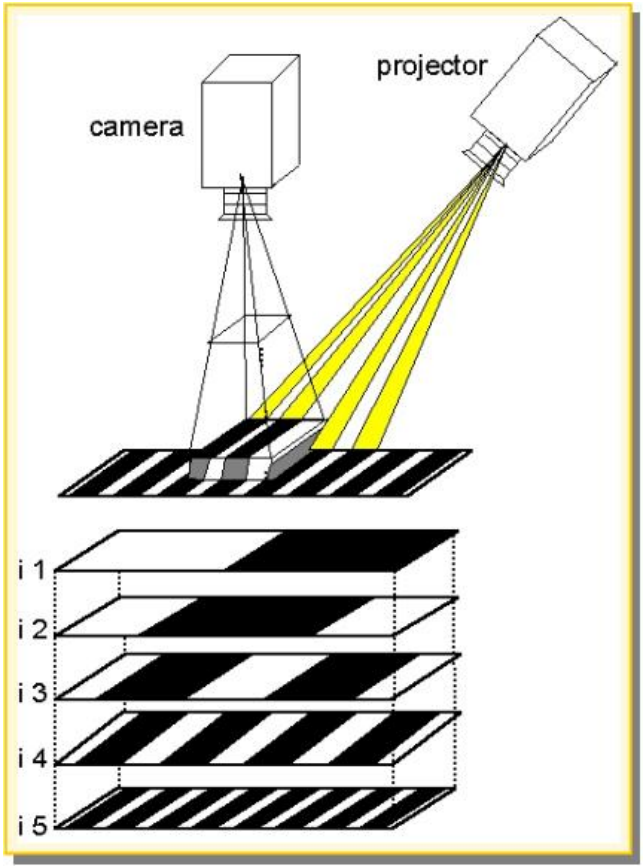
Structured Light

Patterns of light projected onto an object:

- Grids, Stripes, Elliptical patterns
- Pseudo random pattern in Kinect: the codeword that labels each pixel, is obtained from a neighborhood of pixels around it.

Surface shapes deduced from the distortions of the patterns, produced on object surface;

- With knowledge of relevant camera and projector geometry, depth can be calculated by triangulation;
- The challenge in optical triangulation is obtaining correspondence between points in the projected pattern and pixels in the image.



KinectFusion: Depth Surface Mapping and Tracking

It uses the data from the Kinect's depth sensor to construct real-time 3D models of medium sized scenes;

It employs a Simultaneous Localization and Mapping (SLAM) technique, automatic tracking of its pose plus synchronous real-time construction of the finished 3D model;

- 1. register the incoming depth maps from the Kinect by **Iterative Closest Point (ICP)**, which tracks the pose of the sensor from one frame to the next and allows each incoming frame to be placed in a unified global coordinate space;
- 2. employ Volumetric Integration which computes a **truncated signed distance function (SDF)** from each input depth map frame and accumulates the data into a discretized volumetric representation of the global coordinate space;

Pre-ICP registration: bilateral filter and subsampling, build depth map pyramid -> vertex/normal pyramids;

ICP registration: estimate rigid transform, with a **point-to-plane error metric** for faster convergence;

- A coarse to fine iterations through 3-level pyramids and check the stability/validity;

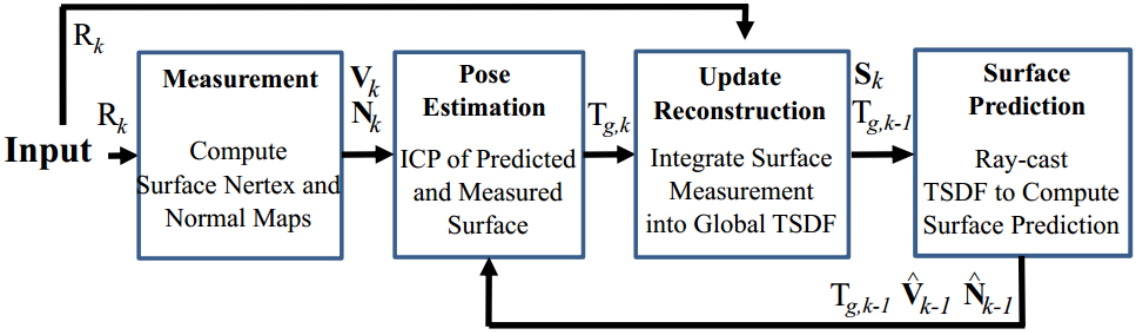
Volumetric integration: 3-d grids of voxels swept through by SDF with depth maps (**weighted averaging**);

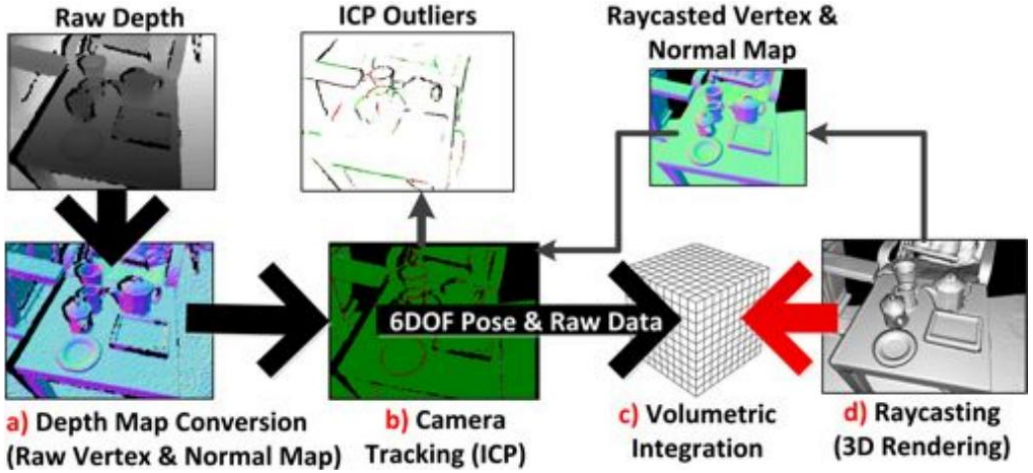
- SDF gets the relative distance from a given 3D point to the surface (+mu/0/-mu) for surface reconstruction;

Post-volumetric integration: to visualize the surface by **ray-casting** and also used the synthetic depth map for ICP at next time, i.e. aligning the live depth map with the raycasted view of the model;

- Find zero-crossings of the truncated SDF in ray-traversing and get its normal by interpolating over neighbor voxels;
- Apply **Marching Cubes** for the extraction of **mesh-based surface** representation from volumetric data.

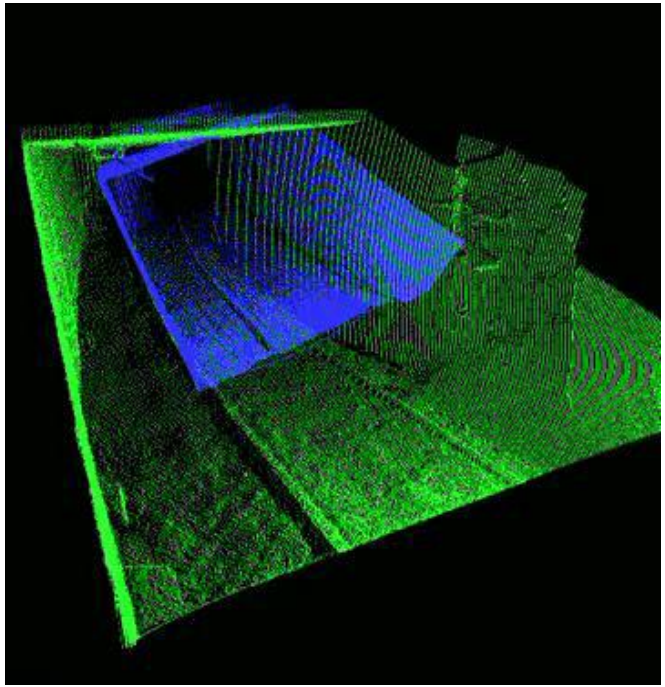
KinectFusion: Depth Surface Mapping and Tracking





ICP (Iterative Closest Point) for 3-D Points Matching

- ICP is a powerful algorithm for calculating the displacement between point clouds;
- The overall speed depends most on the choice of matching algorithm;
- ICP is (in general) only locally optimal;
- Can get stuck in local minima;
- Variants on all stages of ICP:
  - Selecting and weighting source points;
  - Finding corresponding points;
  - Rejecting certain (outlier) correspondences;
  - Choosing an error metric;
  - Minimization.



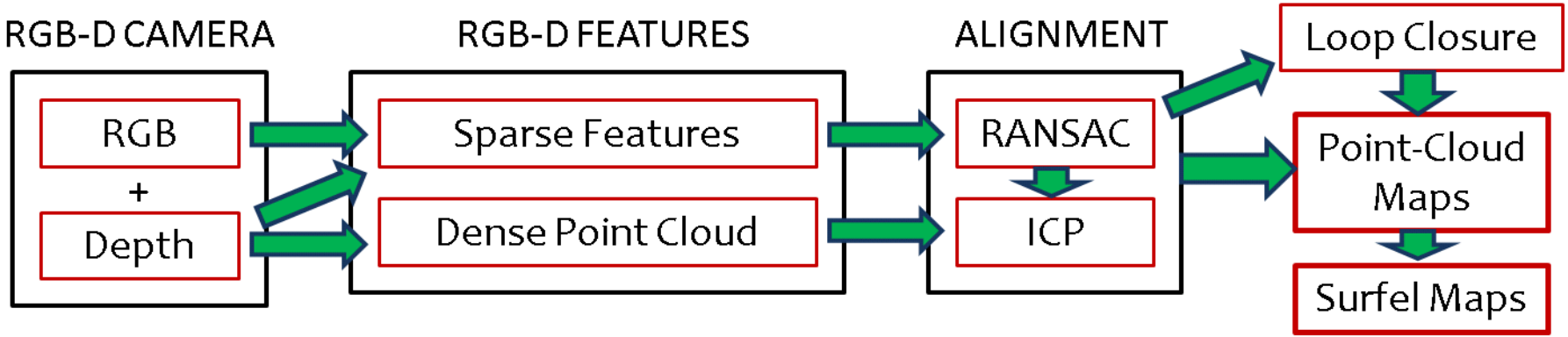
- Note: SVD-based rigid transform estimation

$$W = \sum_i (\mathrm{p}_i - \bar{\mathrm{p}})(\mathrm{q}_i - \bar{\mathrm{q}})^\top = USV^\top$$

$$R = UV^\top$$
$$\mathrm{t} = \bar{\mathrm{p}} - R\bar{\mathrm{q}}$$

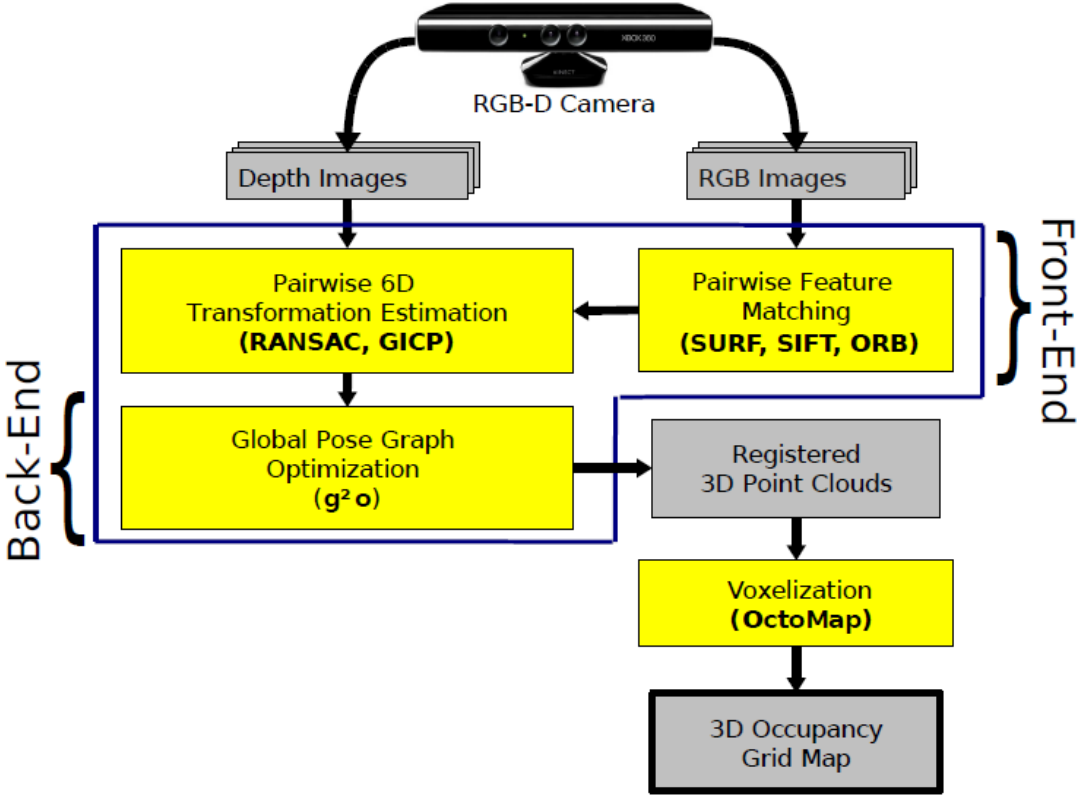
Combination of RGB Feature + Depth ICP

- RGBD-ICP: use both sparse visual features and dense point clouds for frame-to-frame alignment and loop closure detection;
- Visual and depth information are also combined for view-based loop closure detection, followed by pose optimization to achieve globally consistent maps.
- The surfel representation is updated incrementally.



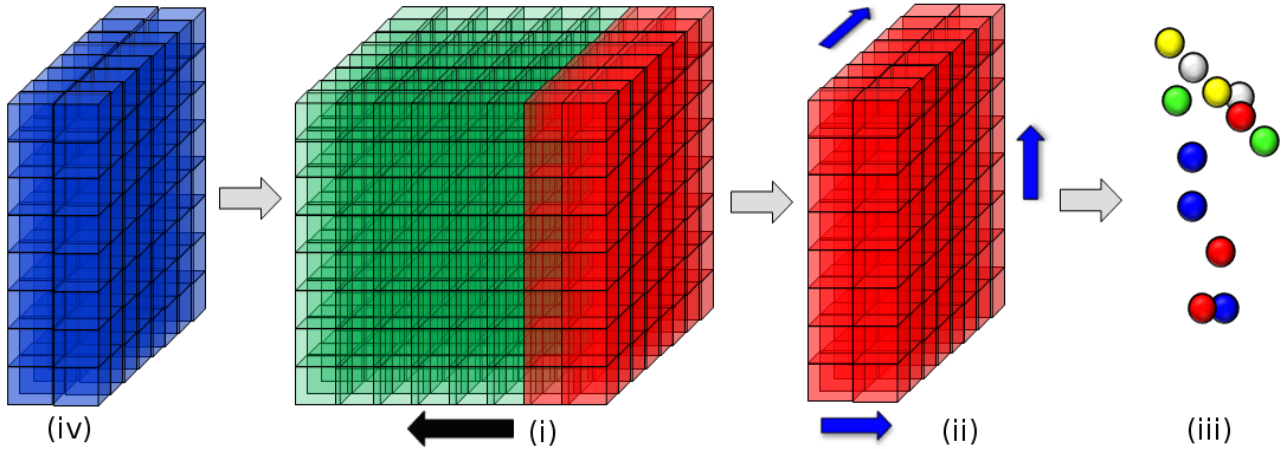
Combination of RGB Feature + Depth ICP

- Extract visual features associated to 3D points;
- Mutually register pairs of image frames and build a pose graph, optimized using g2o;
- Generate a textured voxel occupancy map using the OctoMapping approach.



Kintinous: Spatially Extended KinectFusion

- Dense mesh-based mapping of extended scale environments in real time;
  - Region of space being mapped by KinectFusion vary;
  - Extract a dense point cloud from the regions leaving the KinectFusion volume;
  - Incrementally add points into a triangular mesh structure.
- TSDF in KinectFusion: a volumetric representation of the scene, which zero crossings used for extracting a predicted surface by ray casting, and be incrementally integrated with aligned new ones via a weighted running average;
- Pose graph: pose wrt surface slice and new elements added on boundary crossings;
- Mesh triangulation: TSDF volume slices fed into greedy **mesh triangulation**;
- Visual Odometry FOVIS: **a RGB feature-based method to substitute ICP estimate**;
- Loop closure detection: **place recognition by DBoW of SURF features (drift reduction);**
- Map reintegration: synthesize a depth map by mesh rendering to merge or interleave into the current Kinect depth (replace ray casting?).



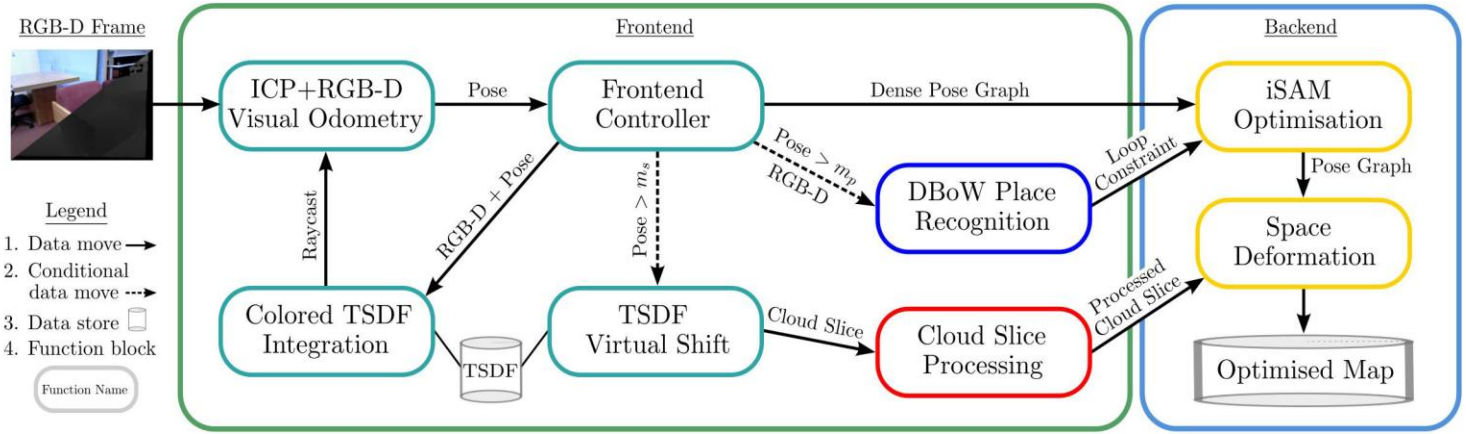
The four main steps of the Kintinuous algorithm:

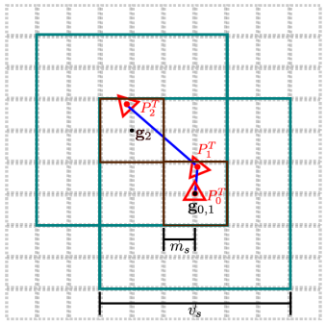
(i) Camera motion exceeds movement threshold (**black** arrow);

(ii) Volume slice (red) is raycast (orthogonal directions shown in blue arrows) for point extraction and reset;

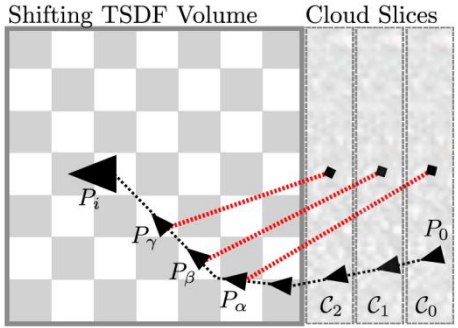
(iii) Point cloud extracted and fed into greedy mesh triangulation algorithm;

(iv) New spatial region enters volume (blue).

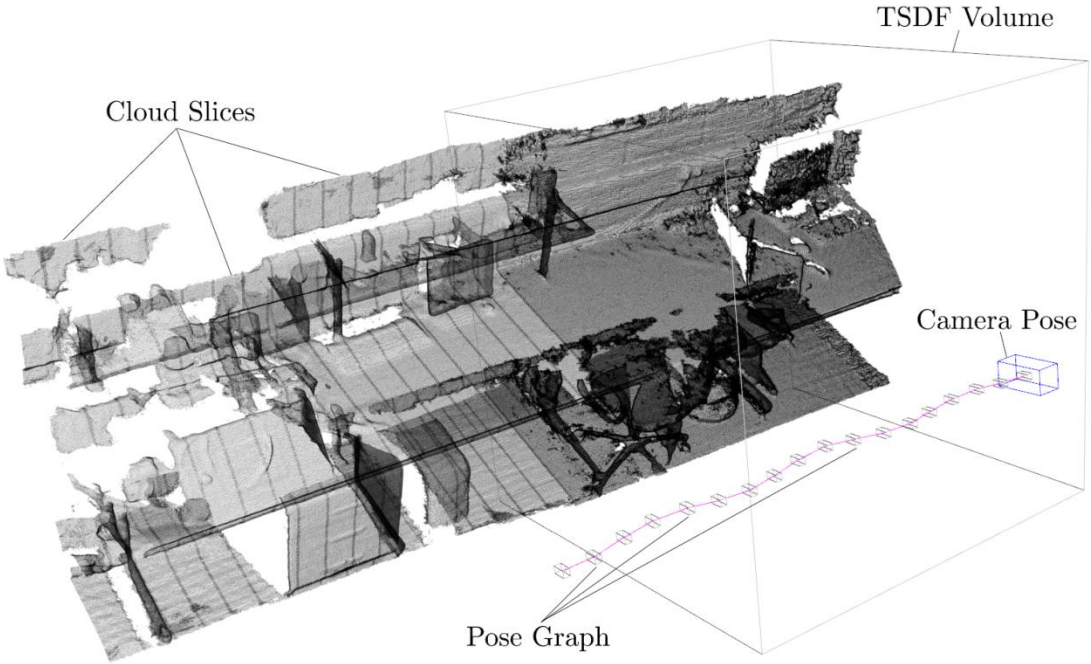




Visualization of the interaction between the movement threshold m_s and the shifting process.



2D visualization of the association between extracted cloud slices, the camera poses and the TSDF volume.



Visualization of a shifted TSDF volume with extracted cloud slices and pose graph highlighted, using dynamic cube positioning. The pose graph is drawn in pink, while small cuboids are drawn for camera poses that have cloud slices associated with them.

RGB-D Visual Odometry

- Estimate the camera motion by aligning two consecutive intensity images with corresponding depth maps obtained from an RGB-D camera;
- Object function based on the photo-consistency assumption;
- Warping: 2-d unprojected to 3-d space and then reprojected to 2-d image;
  - Twist coords: 3-d linear velocity, 3-d angular velocit $\boldsymbol{\xi} = (\nu_1, \nu_2, \nu_3, \omega_1, \omega_2, \omega_3)^\top$

$$\mathbf{p} = \pi^{-1}(\mathbf{x}, Z_1(\mathbf{x})) = Z_1(\mathbf{x}) \left(\frac{u + c_x}{f_x}, \frac{v + c_y}{f_y}, 1 \right)^\top$$

$$\tau(\boldsymbol{\xi}, \mathbf{x}) = \pi(T(g(\boldsymbol{\xi}), \mathbf{p})) = \pi(T(g(\boldsymbol{\xi}), \pi^{-1}(\mathbf{x}, Z_1(\mathbf{x}))))$$

$$T(g, \mathbf{p}) = R\mathbf{p} + \mathbf{t}.$$

$$\pi(T(g, \mathbf{p})) = \left(\frac{f_x x}{z} - c_x, \frac{f_y y}{z} - c_y \right)^\top$$

- MAP estimation: IRLS.

Combination of FOVIS and RGB-D Visual Odometry

- FOVIS: visual odometry based on RGB features;
- Generalized ICP: Both Photo-consistency constraints and ICP constraints.

$$\mathbf{E}_{icp} = \sum_k \left\| \left(\mathbf{v}^k - \exp(\hat{\xi}) \mathbf{T} \mathbf{v}_n^k \right) \cdot \mathbf{n}^k \right\|^2 \qquad \mathbf{E}_{rgbd} = \int_\Omega \left[I(w_\xi(x, t_1), t_1) - I(w_\xi(x, t_0), t_0) \right]^2 \mathrm{d}x$$

$$I_{RGB} : \Omega \times \mathbb{R}_+ \to [0,1]^3, \quad (x,t) \mapsto I_{RGB}(x,t)$$

$$h : \Omega \times \mathbb{R}_+ \to \mathbb{R}_+, \quad (x,t) \mapsto h(x,t)$$

$$S : \Omega \to \mathbb{R}^3, \quad x \mapsto S(x)$$

$$S(x) = \left(\frac{(x+o_x) \cdot h(x)}{f_x} \quad , \quad \frac{(y+o_y) \cdot h(x)}{f_y} \quad , \quad h(x) \right)^\top$$

$$G(g, P) = RP + T$$

$$\pi(G) = \left(\frac{G_1 f_x}{G_3} - o_x \quad , \quad \frac{G_2 f_y}{G_3} - o_y \right)^\top$$

$$w_\xi : \Omega \times \mathbb{R}_+ \to \Omega, \quad (x,t) \mapsto w_\xi(x,t)$$

$$w_\xi(x,t) = \pi\Big(G\big(\exp((t-t_0)\widehat{\xi}) g(t_0), S(x) \big) \Big)$$

$$\mathbf{E} = \mathbf{E}_{icp} + w \cdot \mathbf{E}_{rgbd}$$

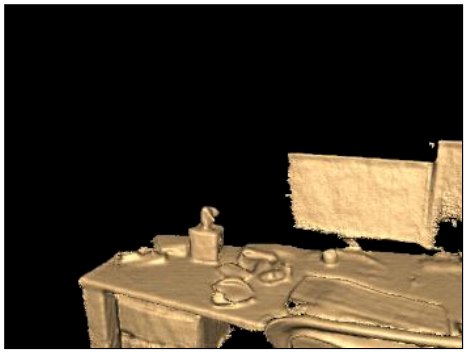
Scalable Real-time Volumetric Surface Reconstruction

Large active region

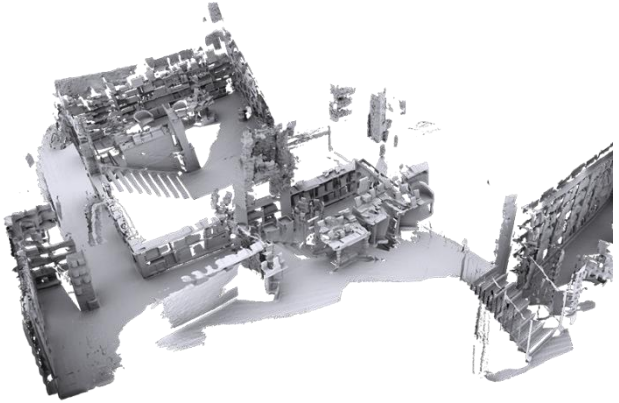
cover full sensor range

improved tracking quality

better user experience

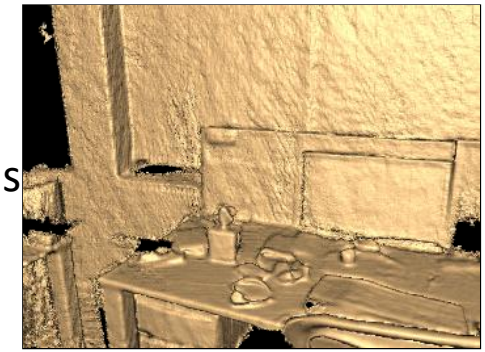


clipped volume (512^3)

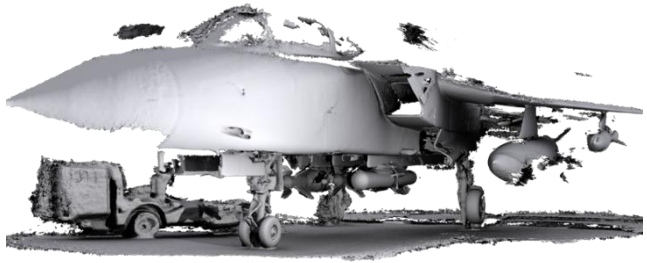


Bidirectional lossless streaming

revisit previously scanned areas



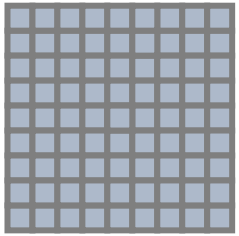
extended volume (1024^3)



Scalable Real-time Volumetric Surface Reconstruction

Background

volumetric surface reconstruction with regular grids



3D regular grid: $O(n^3)$ space and time

512^3 4-byte voxels: 512 MB (4mm per voxel: 2m volume)

640^3 voxels: 1 GB (4mm per voxel: 2.5m volume)

Kinect: 40cm to 6.5m range

Dense free-space storage wastes memory

Scalable approach

streaming hierarchical grids

Hierarchical

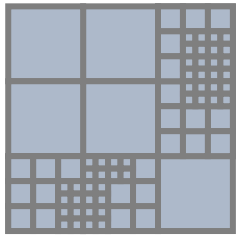
compress free space

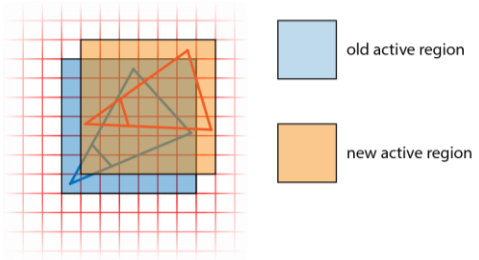
maintain large active region

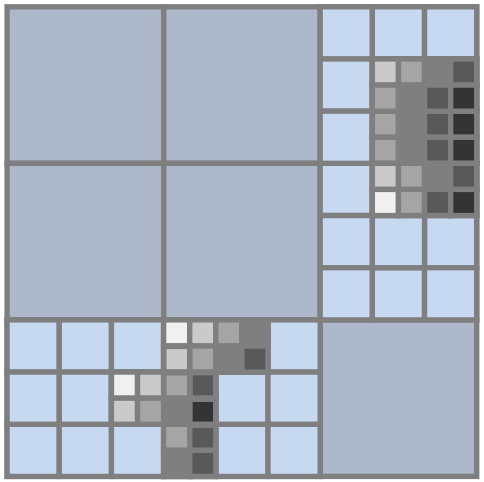
Streaming

volume moves with camera

no limits on range and revisiting

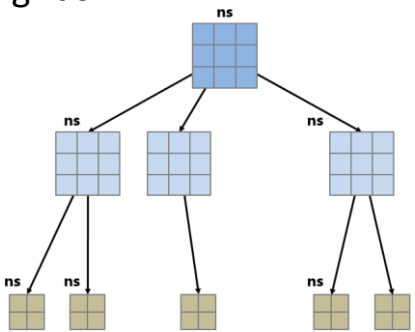




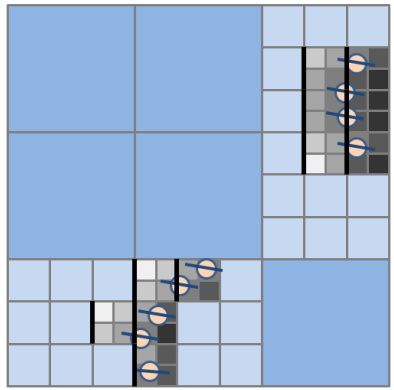


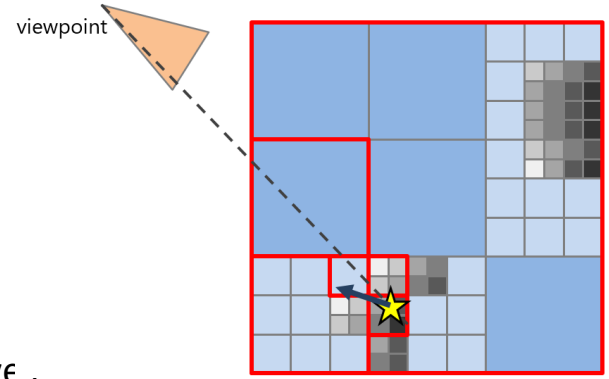
Scalable Real-time Volumetric Surface Reconstruction

1. **Align** point clouds
   - leaves are regular grids
   - update SDF (as before)
2. **Integrate** new information
   - find and allocate child grids
   - update SDF in leaves
   - scan children
   - if any are near surface:
     - mark self



Hierarchical integration
- walk over **coarse** root
- find voxels near surface
- allocate child grids
- recurse



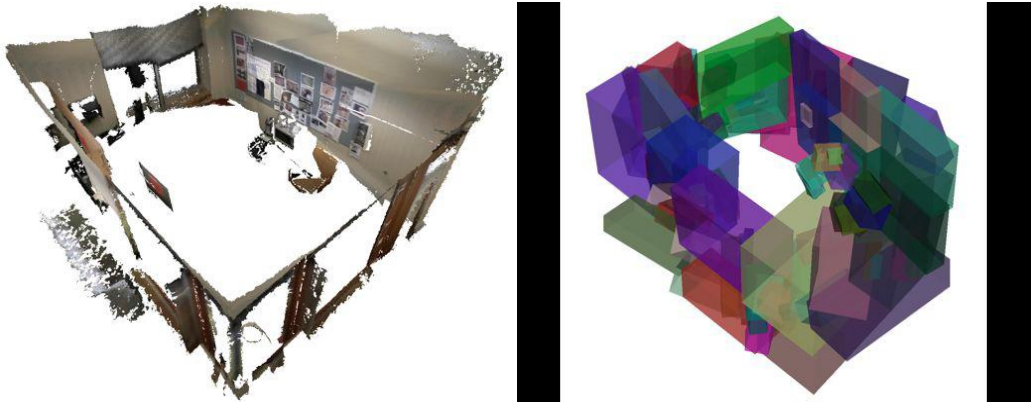


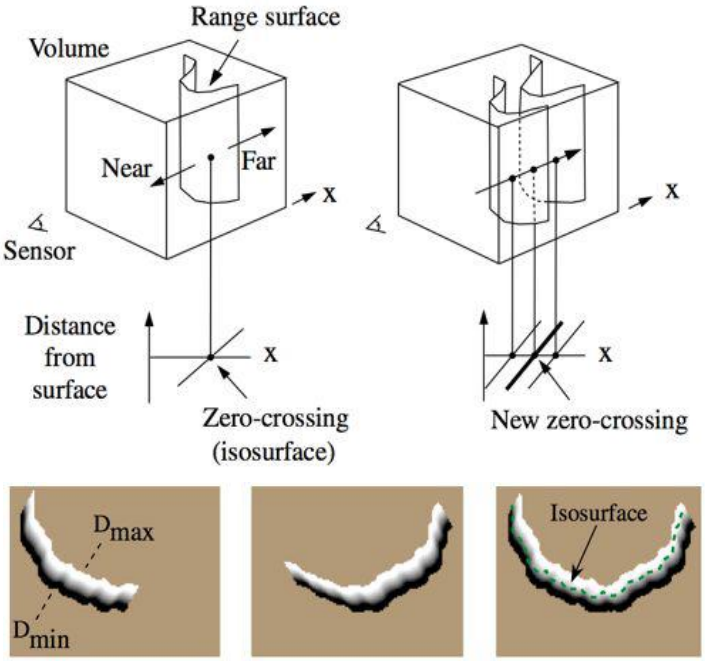
3. **Extract** 3D surface
   - Recursively march ray
   - **Skip** large empty regions
   - Find zero crossing in leaves

Patch Volume: Segmentation-based Consistent Mapping

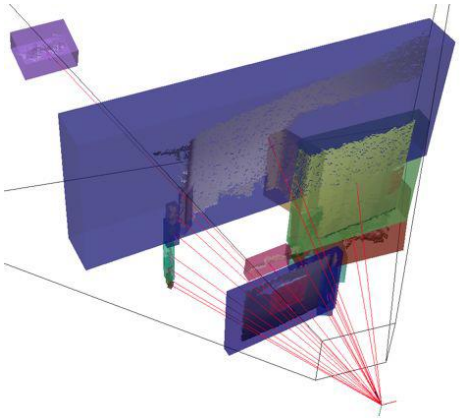
Model: A collection of fusion volumes

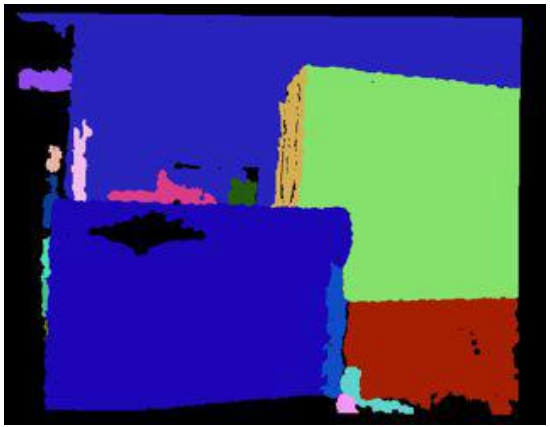
- Allocate volumes based on planar patches
- Save memory: model only occpupied space
- Enable arbitrary scale
- Shift volumes for global consistency

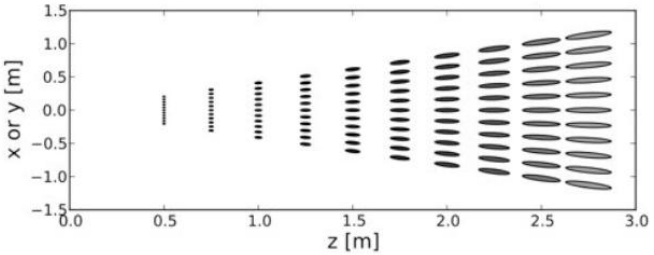




Patch Volume: Segmentation-based Consistent Mapping







Nguyen et al., "Modeling Kinect Sensor Noise for Improved 3D Reconstruction and Tracking", 3DIM/3PVT, 2012

Geometry Volume

$$w^d_{new} = \frac{\sigma_z(z_{min})}{\sigma_z(D_f(u,v))}\frac{z^2_{min}}{D_f(u,v)^2}$$

$$v_F \leftarrow \frac{w^d_{old}d_{old} + w^d_{new}d_{new}}{w^d_{old} + w^d_{new}}$$

$$v_{W_F} \leftarrow \min(w^d_{old} + w^d_{new}, w_{max})$$

Color Volume

$$w^c_{new} = w^d_{new}\left(1 - \frac{|d|}{d_{max}}\right)$$

$$v_C \leftarrow \frac{w^c_{old}c_{old} + w^c_{new}c_{new}}{w^c_{old} + w^c_{new}}$$

$$v_{W_C} \leftarrow \min(w^c_{old} + w^c_{new}, w_{max})$$

Dense Alignment

Geometric error

$$\epsilon_g = w_g(\mathbf{p}_f - \mathbf{p}_r)\cdot\mathbf{n}_r$$

$$w_g = \frac{\sigma_z(z_{min})}{\sigma_z(\mathbf{p}_r.z)}$$

Color error

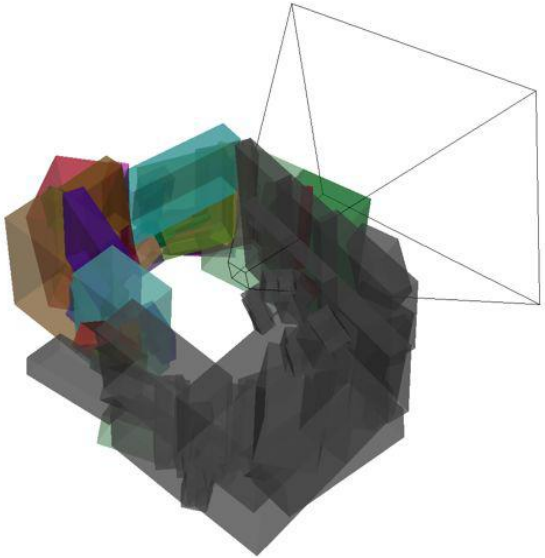
$$\epsilon_c = (Y_f(u_f, v_f) - Y_r(u,v))$$

Combined error

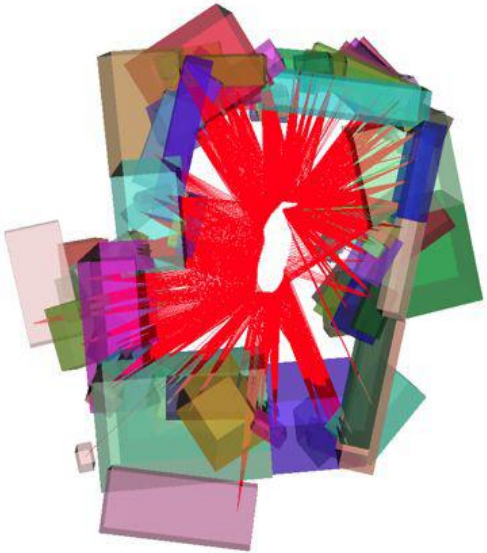
$$\epsilon = \lambda\epsilon_g + \epsilon_c$$

Patch Volume: Segmentation-based Consistent Mapping

loop closure alignment



graph optimization



result

Dense Scene Reconstruction with Points of Interest

Frame-to-model registration + global optimization

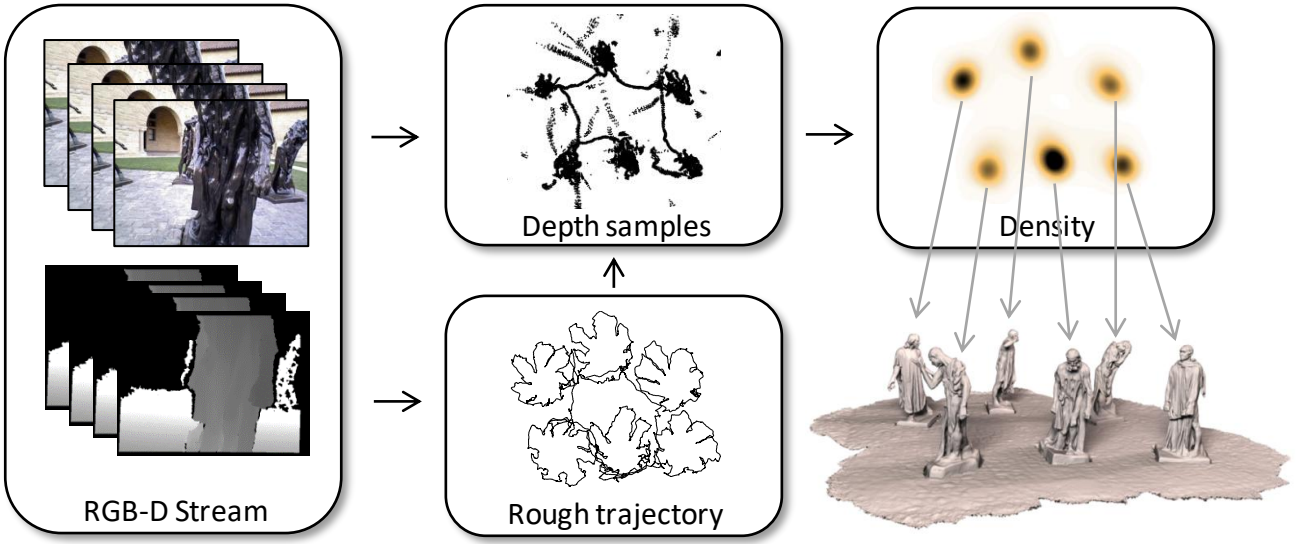
Estimate an optimal camera trajectory that is **globally consistent** and **locally accurate**

Frame-to-model registration: a chicken-egg problem

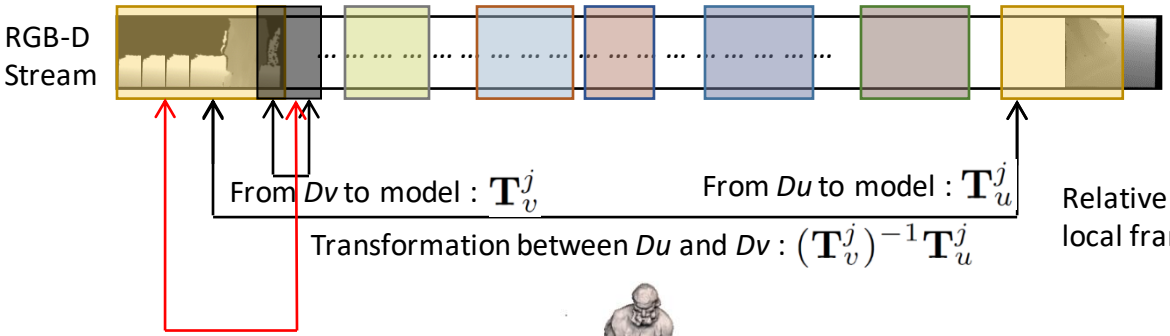
Build local models (both spatially and temporally) for frame-to-model registration

Points of interest: densely scanned regions with details

Offline global optimization to distribute residual error

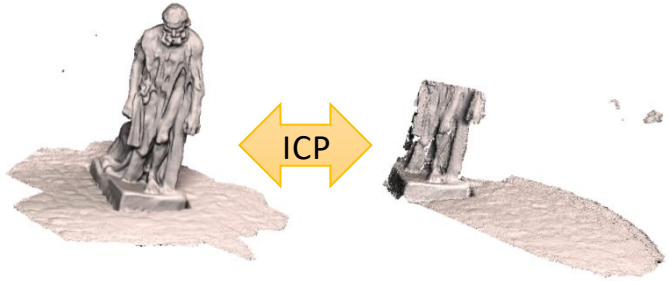


Dense Scene Reconstruction with Points of Interest



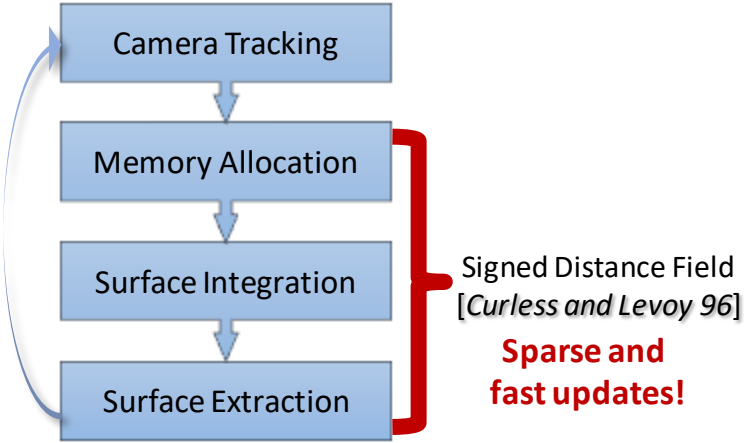
Relative transformations from local frame-to-model registration

Cross-fragment edges are established based on ICP result



Global optimization over a fully connected pose graph

Real-time 3D Reconstruction at Scale using Voxel Hashing



Signed Distance Field
[*Curless and Levoy 96*]

Sparse and fast updates!

Idea: unstructured 3D scene rep

High-resolution (including color) $\in O(n^2)$

Efficient allocation and updates

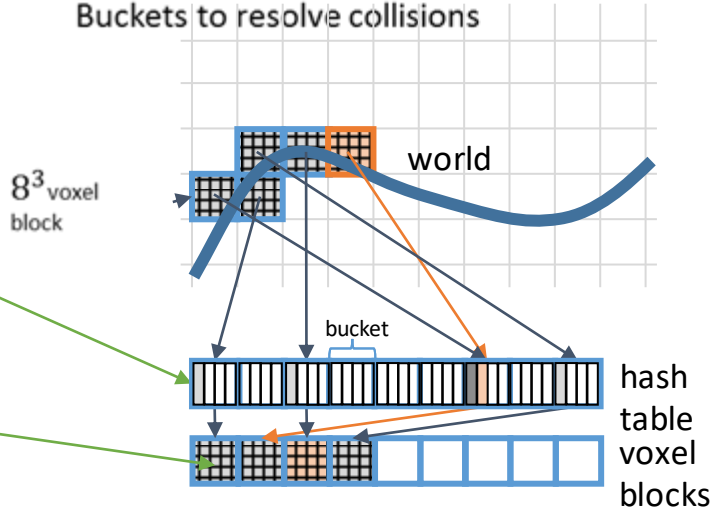
Spatial hashing [*Teschner et al. 03*]

$$H(x,y,z) = (x \cdot p_1 \oplus y \cdot p_2 \oplus z \cdot p_3) \bmod n$$

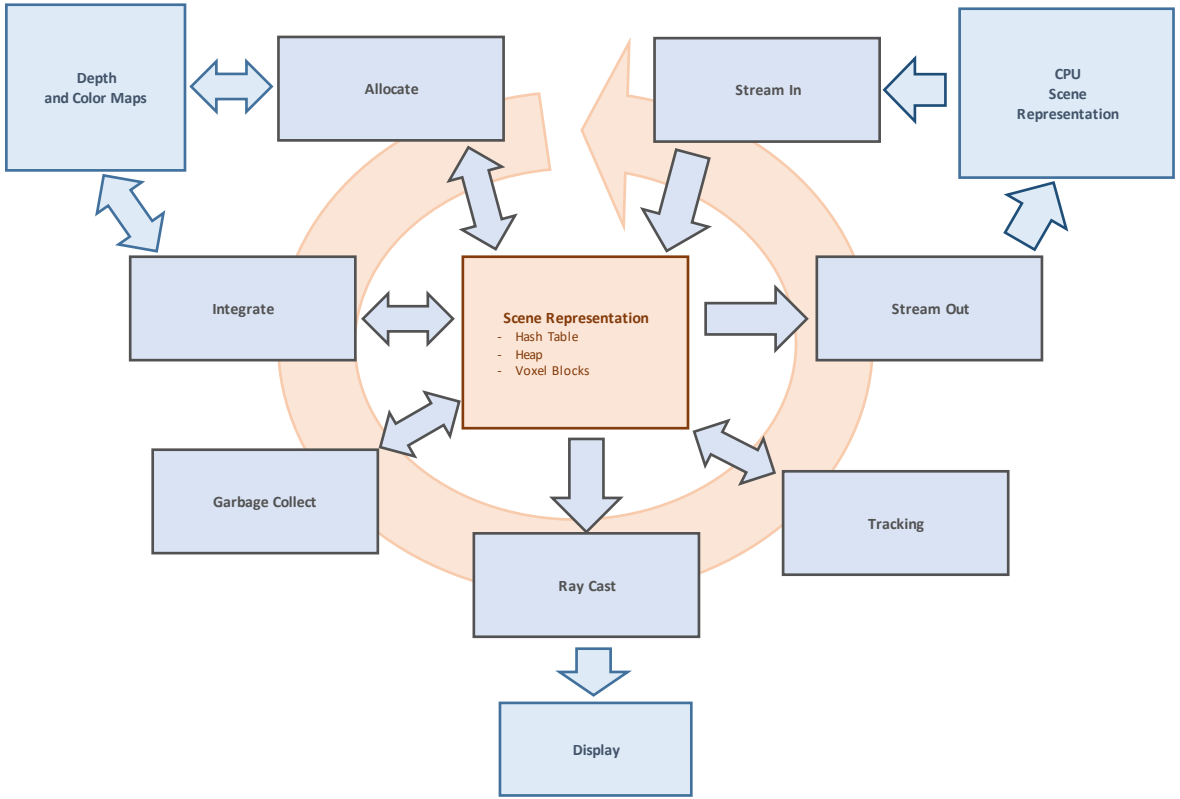
Buckets to resolve collisions

```
struct HashEntry {
   short position[3];
   short offset;
   int   pointer;
};
```

```
struct Voxel {
   float sdf;
   uchar colorRGB[3];
   uchar weight;
};
```



Real-time 3D Reconstruction at Scale using Voxel Hashing

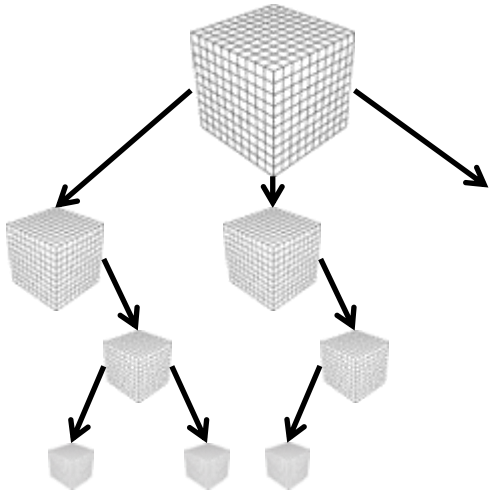


Real-time 3D Reconstruction at Scale using Voxel Hashing

Hash maintenance is easy
- Insert, remove, reorganize, …
- $O(1)$ operations

Hierarchy maintenance costly
- Insertion, removal, tree traversal, …

Streaming in/out complete sub-trees



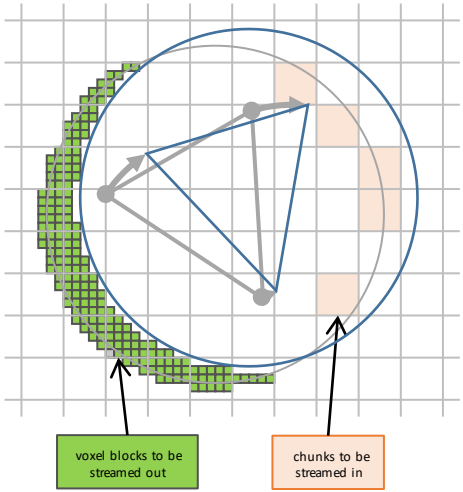
Iterated Closest Point (ICP)
- [Besl and McKay 92], [Low 04]
- Projective correspondences
- Three level hierarchy

$$T_{g,k} = \begin{bmatrix} R_{g,k} & t_{g,k} \\ 0 & 1 \end{bmatrix}$$

Stream voxel blocks outside ROI to CPU
- Gather in chunks

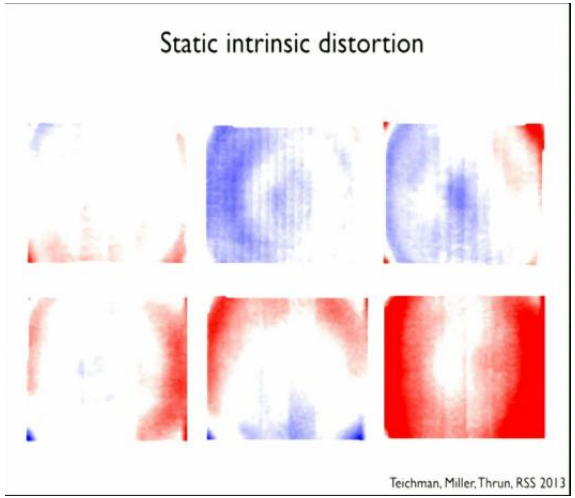
Stream chunks completely to the GPU
- Conservative bound for view frustum

Insertion/deletion in the hash is simple

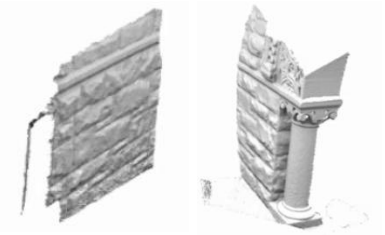


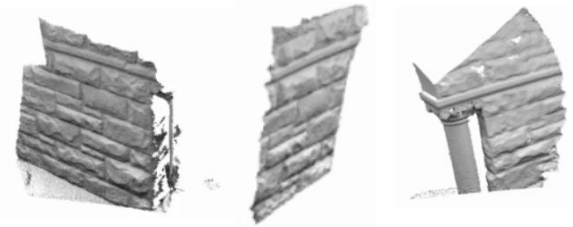
Elastic Fragments for 3-D Reconstruction

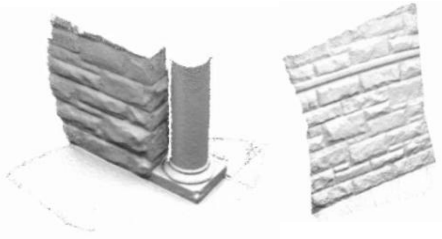
- High-frequency noise
- Quantization
- Missing data
- Static intrinsic distortion



- Partition trajectory into short segments.
- Reconstruct a scene **fragment** from each segment.
- Establish correspondences between pairs of fragments.
- **Nonrigidly** align all fragments to each other.







$$E(\mathbf{T}) = E_{\text{align}}(\mathbf{T}) + E_{\text{reg}}(\mathbf{T})$$

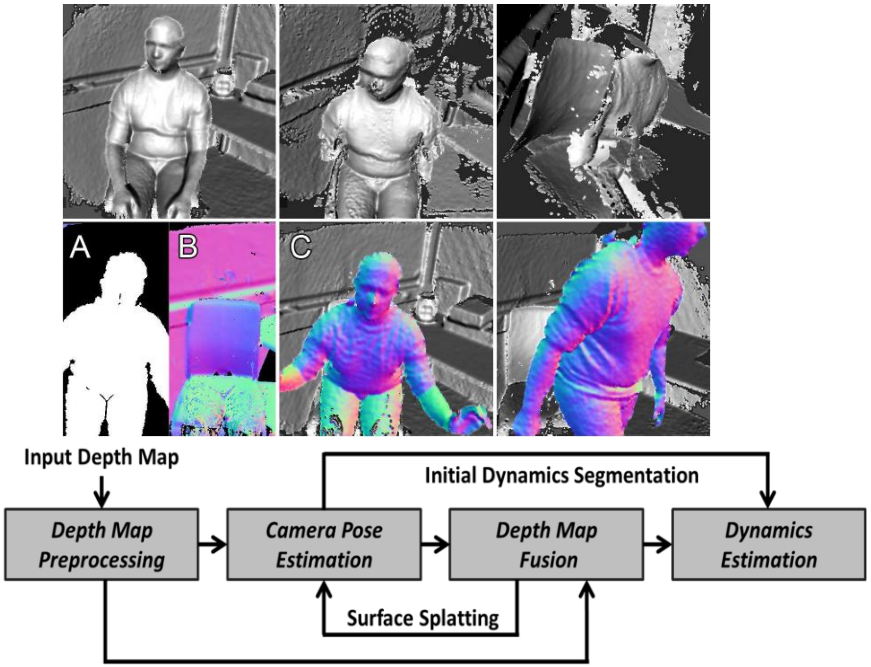
$$E_{\text{align}}(\mathbf{T}) = \sum_{i,j} \sum_{(\mathbf{p},\mathbf{q}) \in \mathcal{K}_{i,j}} \|(\mathbf{p}' - \mathbf{q}') \cdot \mathbf{N}'_{\mathbf{p}}\|^2$$

$$E_{\text{reg}}(\mathbf{T}) = \sum_{i} \sum_{\mathbf{p} \in \mathbf{P}_i} \sum_{\mathbf{q} \in \mathcal{N}(\mathbf{p})} \|(\mathbf{q}' - \mathbf{p}') - \mathbf{R}'_{\mathbf{p}}(\mathbf{q} - \mathbf{p})\|^2$$

Points-based Fusion

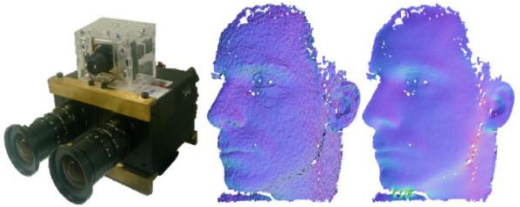
- Trade scale to achieve higher quality reconstructions of small objects/scenes;
- Handle larger scenes:
  - Trade real-time performance and/or quality;
  - Limit the bounds of the active reconstruction.

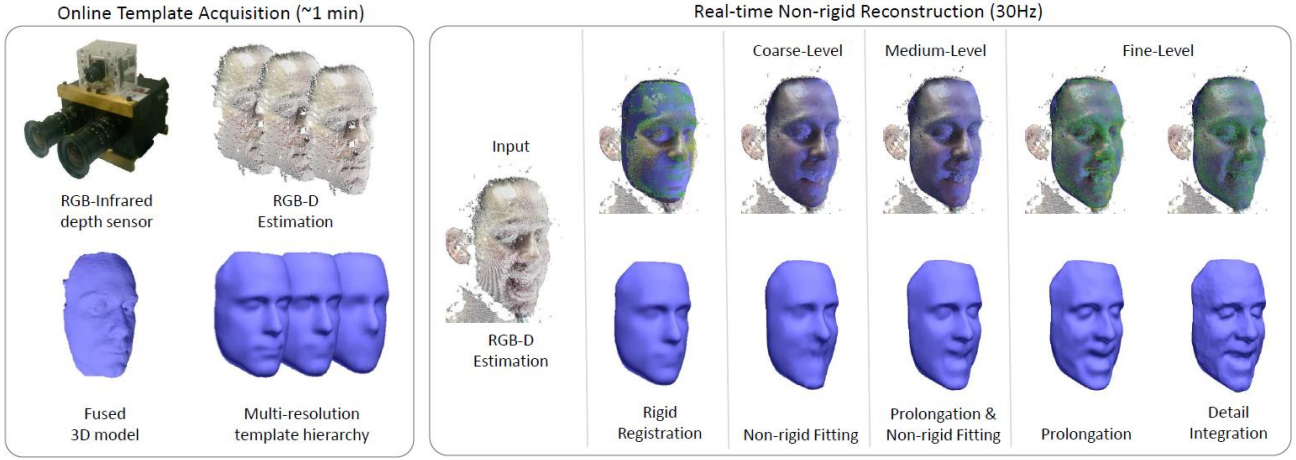
- Real-time dense reconstruction for additional spatial scale and robustness in dynamic scenes;
  - A simple and flat point-based representation;
  - Leverage the graphics pipeline (camera pose estimation by ICP and surface splatting, data association, outlier removal, fusion of depth maps, and detection/update of dynamic object by point-based region growing).



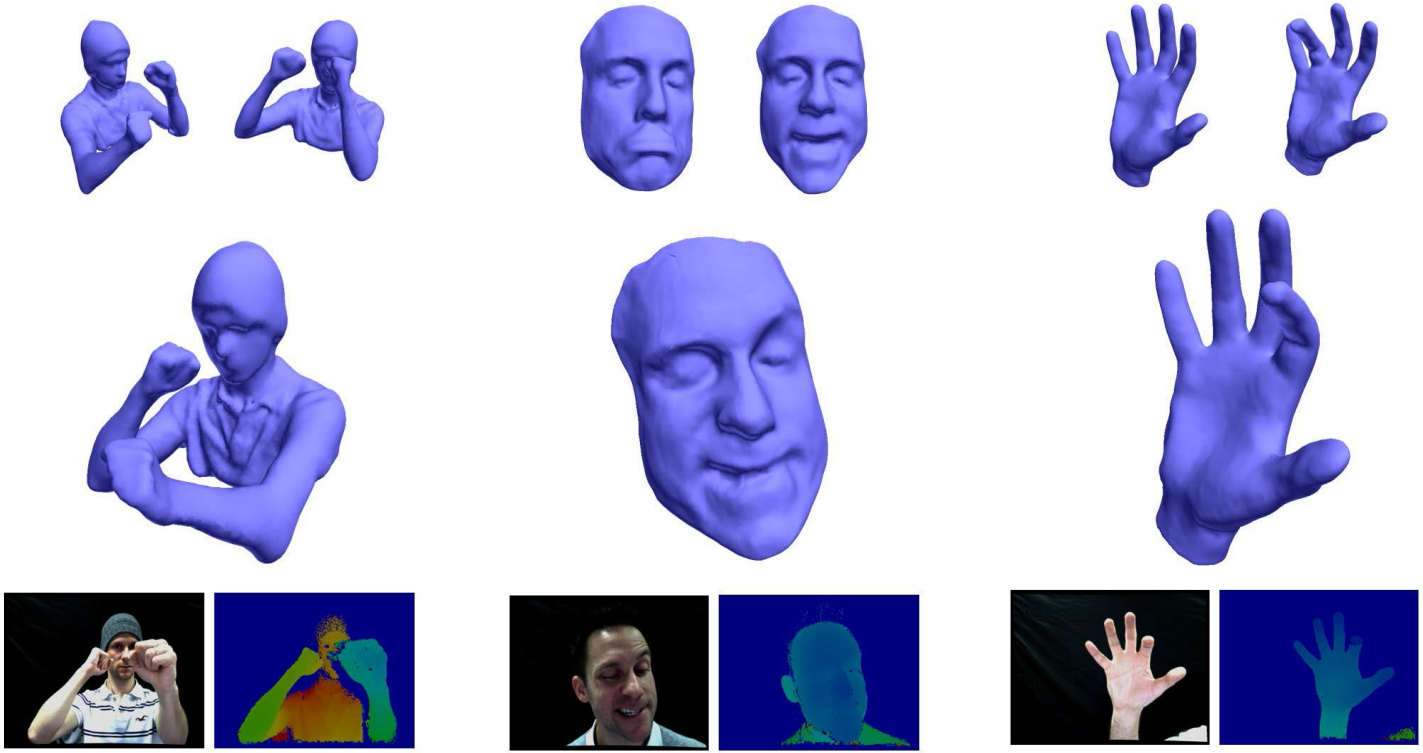
Non-rigid Reconstruction by Registration

- Scan a smooth template model of the subject as they move rigidly;
- Non-rigid registration to the smooth template using an extended non-linear as-rigid-as-possible (ARAP) framework: Gauss-Newton solver
  - High-frequency details are fused onto the final mesh;
  - A thin shell deformation model.



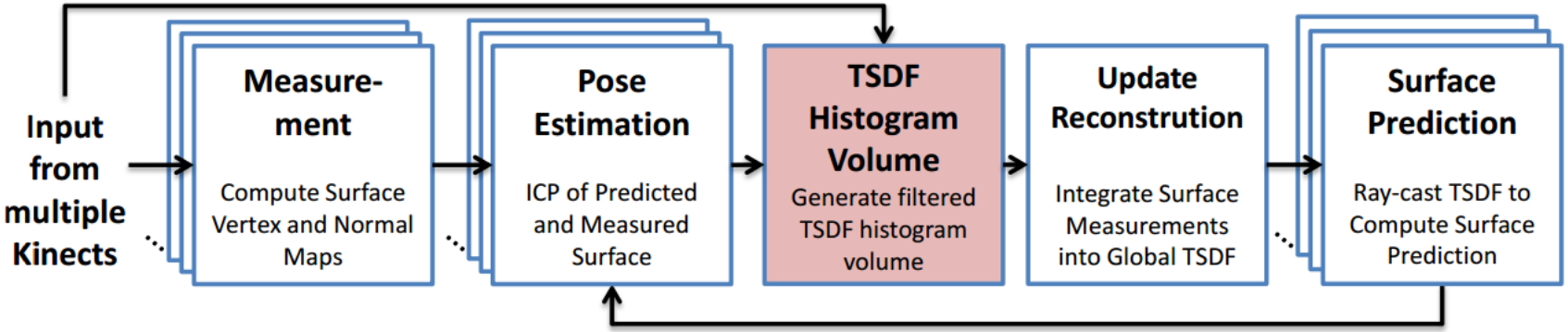


Non-rigid Reconstruction by Registration



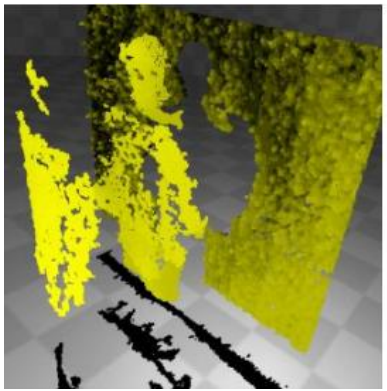
OmniKinect: A Multiple Kinect Sensor System

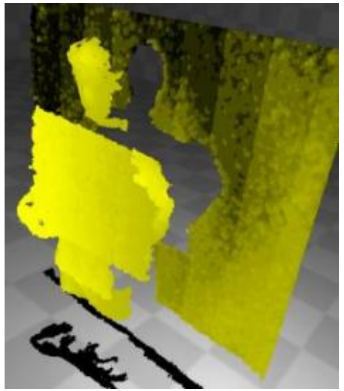
- A larger number of Kinects used simultaneously to produce high quality volumetric reconstructions whilst overcoming systematic errors in the depth measurements;
- The image-based visual hull (IBVH) algorithm reconstructs and renders 3D objects from silhouette images, therefore provide pure dense image synthesis faster than with KinectFusion;
- Use background subtraction based on color- and depth values to segment foreground objects, then carve the point clouds on the image planes by only considering depth values that are inside the silhouettes, which are calculated by using binary foreground masks;
- Perform IBVH rendering followed by point splatting to get both surface estimates. Then, all point splats are culled against the visual hull surface at that pixel.



Shake’n’Sense: Vibrating A Kinect Camera

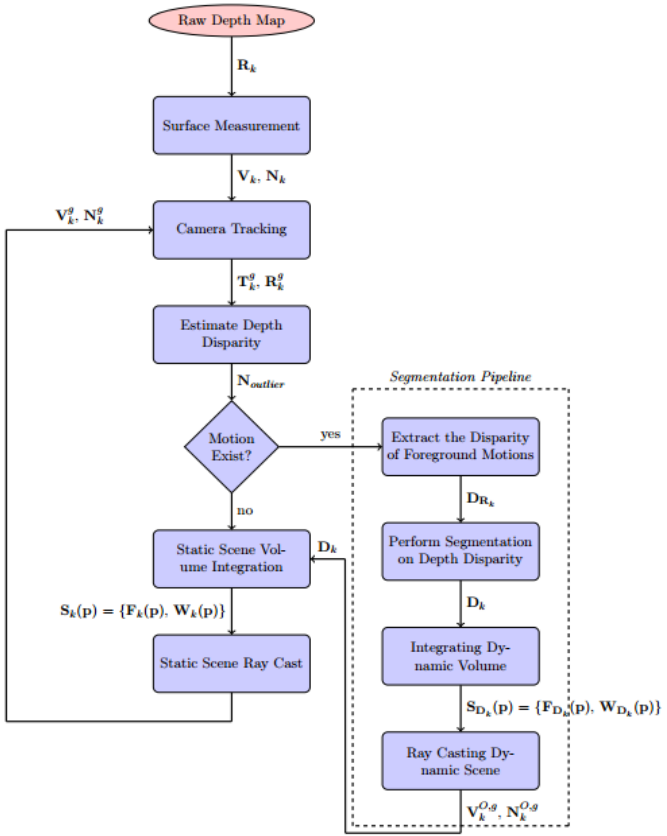
- For structured light sensors such as Kinect, the depth signal severely degrades when multiple cameras are pointing at the same scene, due to the sensor projecting a structured light dot pattern onto the scene continuously, without modulation (crosstalk);
- Minimally vibrate a Kinect camera unit using an offset-weight vibration motor and thereby artificially introduce motion blur, where both the structured light DOE illuminator and the IR camera of the Kinect will move in harmony;
  - The frequency of vibration can be controlled within the range of 15Hz to 120Hz.
  - To accurately assess the optimal vibration frequency, attached an 3-axis accelerometer to the camera’s casing.





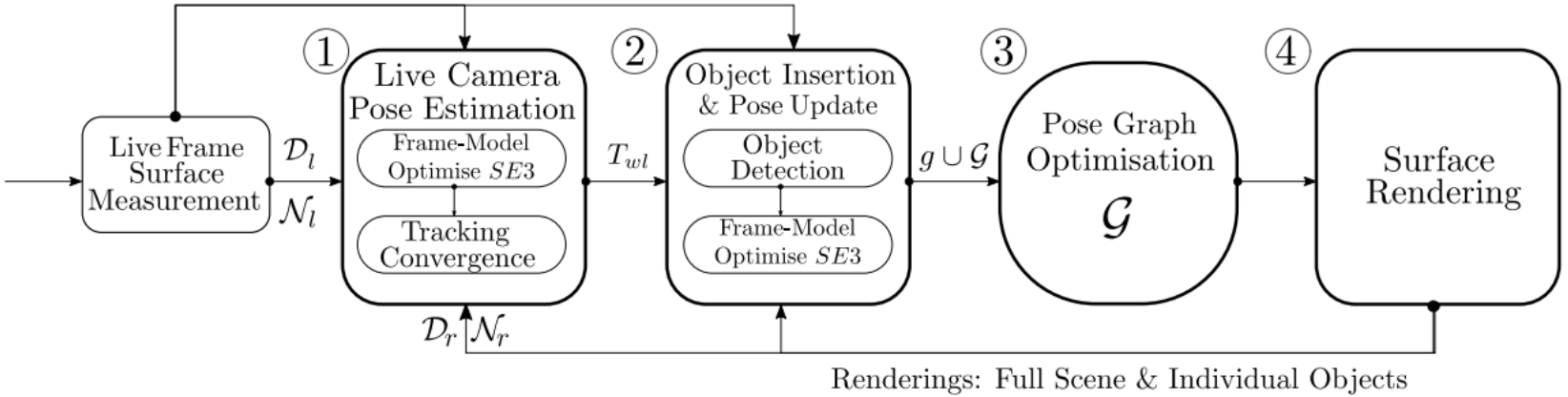
KinfuSeg: Dynamic SLAM Based on Kinect Fusion

- Perform in a dynamic environment, which owns much less restriction on its static surrounding.
- Segment moving objects in a separate 3D volume.
  - Depth based segmentation first to extract motion from the surroundings;
    - Extract all the ICP outliers as depth disparity map, then a mean filter and graph-cut method to achieve segmentation.
  - Build 3D volumes for static scenes and motions respectively;
  - Separate volume integration and ray casting on the static scene and dynamic scene respectively;
- Run in real-time in almost all the situations.



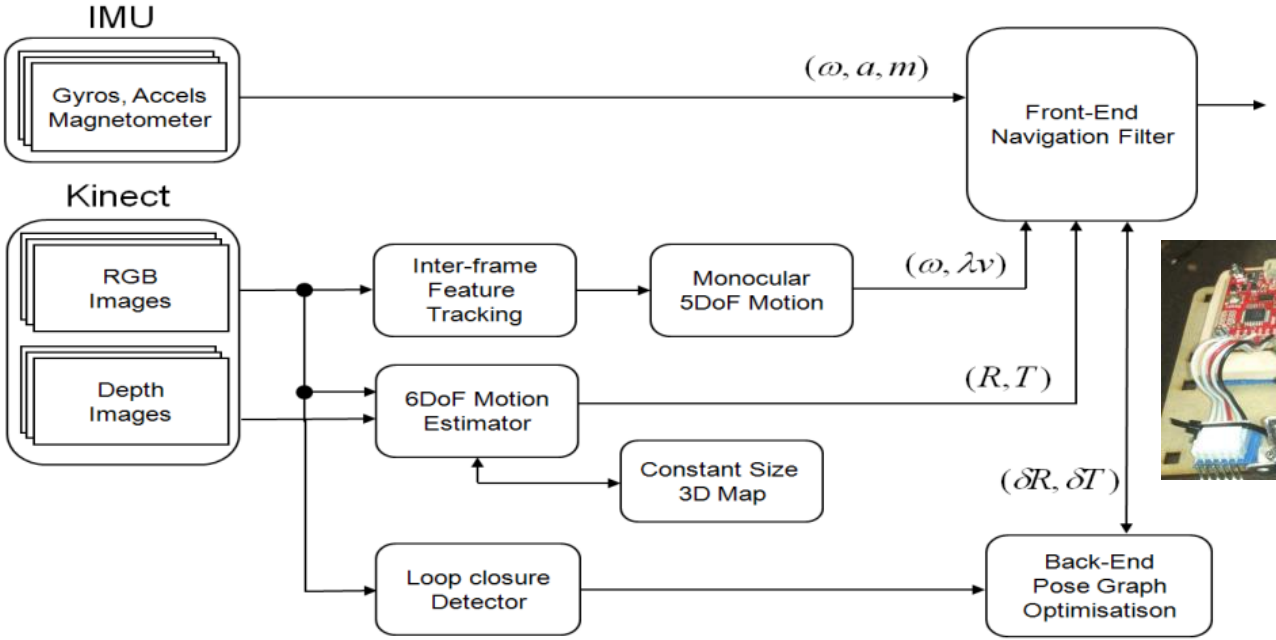
SLAM++: SLAM at the Level of Objects

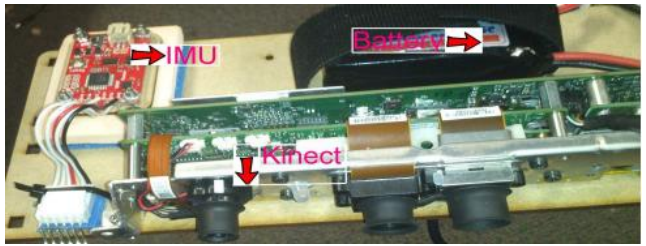
- By recognizing objects in the loop, semantic understanding instantly;
- Currently a small number of precisely known objects by KinectFusion scanning and segmenting;
- Noisy raw detected poses refined with ICP against object model;
- Camera is tracked using ICP against a depth map raycasted from the current whole world model;
- After tracking, only areas failing ICP searched for new objects;
- Pose graph optimization with structural priors (objects located on a common ground plane).



Inertial Kinect Fusion

- The vision node is designed in a modular way that can provide either full 6DOF or partial 5DOF;
- The visual translation in 5DOF mode is treated as a directional constraint of the motion.

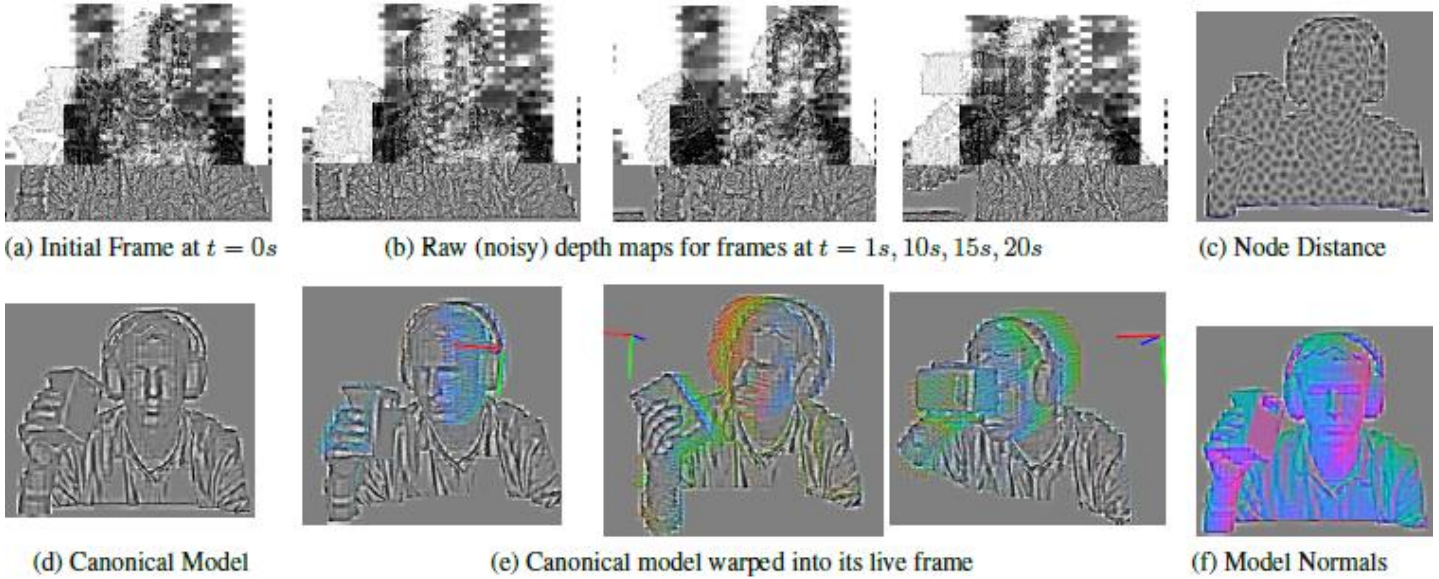




DynamicFusion

- Dense SLAM system capable of reconstructing non-rigidly deforming scenes in real-time by fusing together RGBD scans: solving for a volumetric flow field that transforms the state of the scene at each time instant into a fixed, canonical frame;
  - Dense nonrigid surface fusion with dense nonrigid ICP term;
- Simultaneously estimating a dense volumetric 6D motion field that warps the estimated geometry (dense volumetric warp-field);
  - Following these warps, the scene is effectively rigid, and standard KinectFusion updates can be used to obtain a high quality, denoised reconstruction.

DynamicFusion

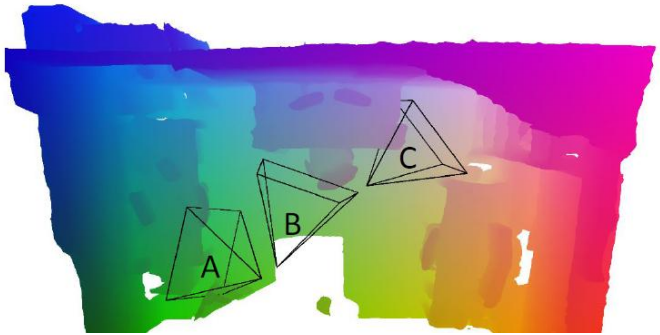


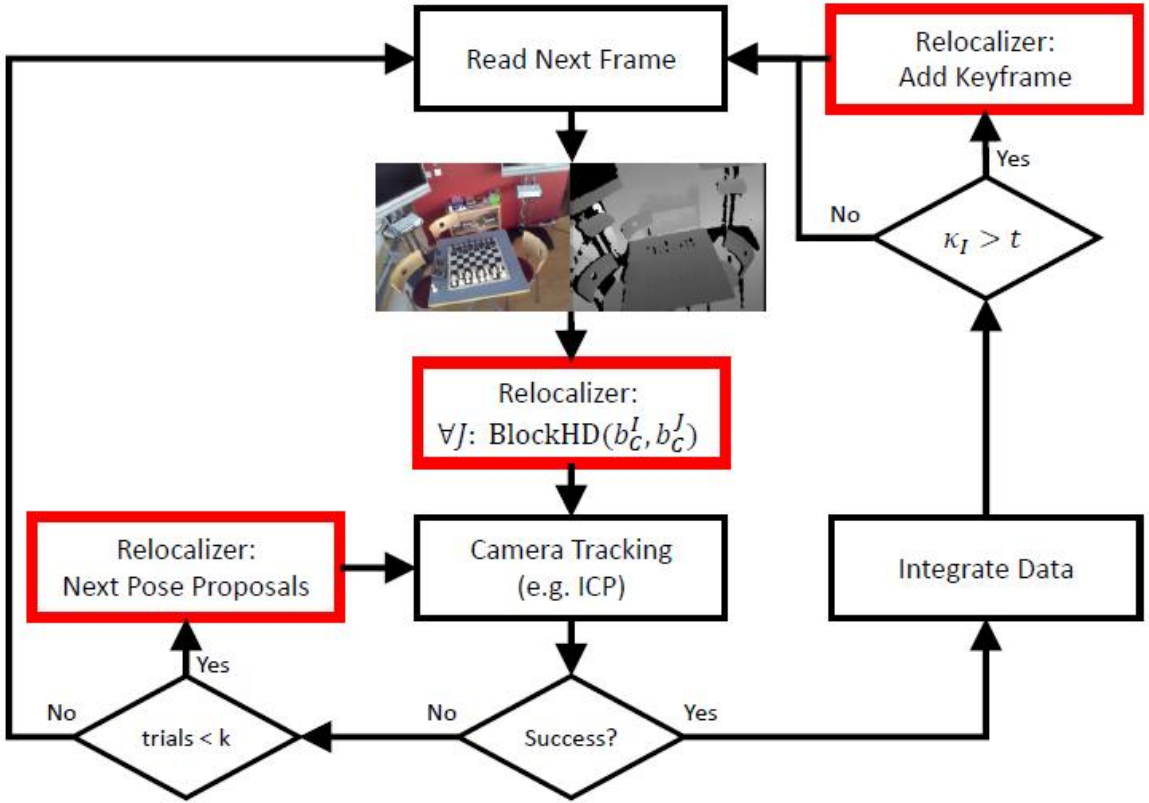
(a) Initial Frame at $t = 0s$ (b) Raw (noisy) depth maps for frames at $t = 1s, 10s, 15s, 20s$ (c) Node Distance

(d) Canonical Model (e) Canonical model warped into its live frame (f) Model Normals

Online stream of noisy depth maps (a,b), the warp field as a set of sparse 6D transform nodes and the nearest node to model surface distance in the canonical frame (c), output with a real-time dense reconstruction of the moving scene (d,e). All depth maps densely fused into a single rigid TSDF reconstruction (d, f).

RGB-D Camera-based Re-localization



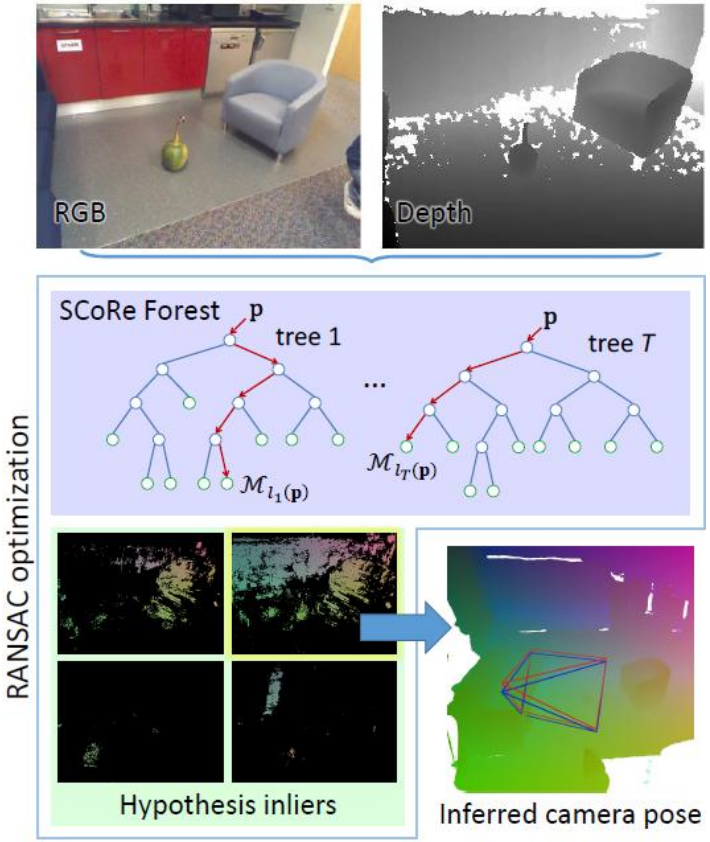


RGB-D Camera Pose Estimation by SCoRe Forest as Classifiers

- A optimization with RANSAC, uses a scene coordinate regression forest (SCoRe Forest) as classifiers to obtain image to scene correspondences;
- The algorithm maintains a set of inlier pixels for each of several camera pose hypotheses;
- The hypothesis with the lowest energy (highest number of inliers) is chosen as the final inferred pose.

Algorithm 1 Pseudocode for RANSAC optimization

$K \leftarrow K_{\text{init}}$
sample initial hypotheses $\{H_k\}_{k=1}^{K}$
initialize energies $E_k \leftarrow 0$ for $k = 1, \ldots, K$
while $K > 1$ **do**
  sample random set of B test pixels $\mathcal{I}$
  for all $i \in \mathcal{I}$ **do**
    evaluate forest to obtain $\mathcal{M}_i$
    for all $k \in \{1, \ldots, K\}$ **do**
      $E_k \leftarrow E_k + e_i(H_k)$
  sort hypotheses (H_k, E_k) by E_k
  $K \leftarrow \lfloor \frac{K}{2} \rfloor$
  refine hypotheses H_k for $k = 1, \ldots, K$
return best pose H_1 and energy E_1

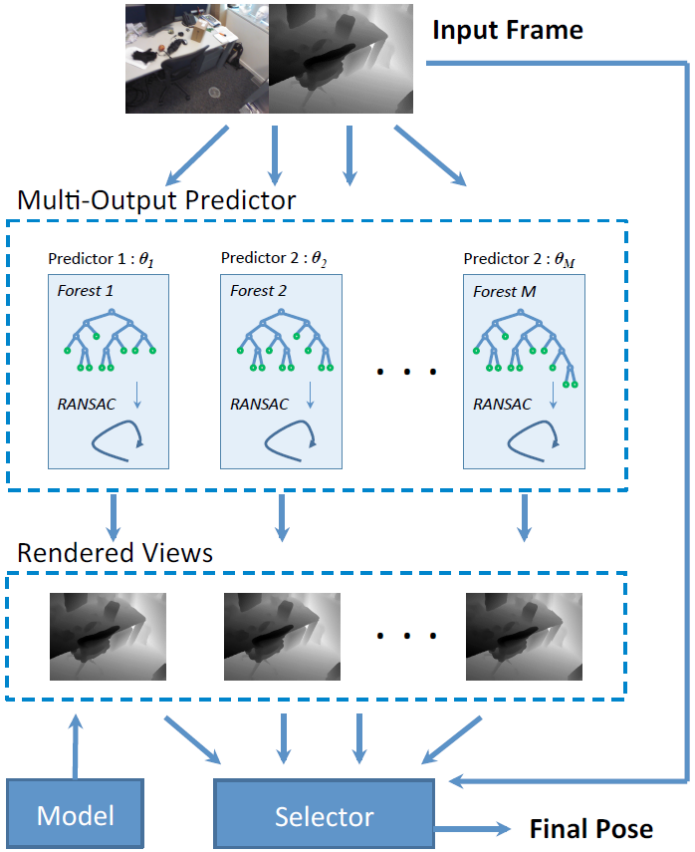


RGB-D Camera Relocalization by Multi-Output Learning

- Formulated as inversion of the generative rendering procedure, i.e., to find the camera pose most similar with the observed input, solved by nonlinear regression;
- A hybrid discriminative-generative learning architecture
  - A set of M predictors which generate M camera pose hypotheses;
  - A 'selector' or 'aggregator' that infers the best pose from the multiple pose hypotheses.
- The output of a predictor can be refined using the 3D model, represented by a truncated signed distance function;
  - Searching for a nearby pose that best aligns the camera space observations with the predicted surface.

Algorithm 1 Learn Multi-Output Predictor

Inputs: $S = \{(\mathcal{I}_j, H_j) : j \in [n]\}, M, \xi, \ell, \sigma$
Initialize with uniform weights: $\mathrm{w}_0 \leftarrow \frac{1}{n} \cdot 1$
$\Theta_0 \leftarrow \{\}$
for $t = 1, \ldots, M$ **do**
  $\Theta_t \leftarrow \Theta_{t-1} \cup \mathrm{TrainForest}_{[19]}(S, \mathrm{w}_{t-1})$
  $\mathrm{w}_t \leftarrow \mathrm{UpdateWeights}(\Theta_t, \mathrm{w}_{t-1}, \ell, \xi, \sigma)$
end for
return Θ_M

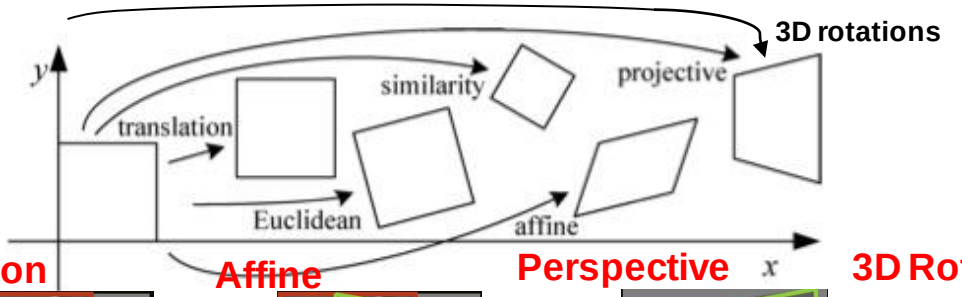


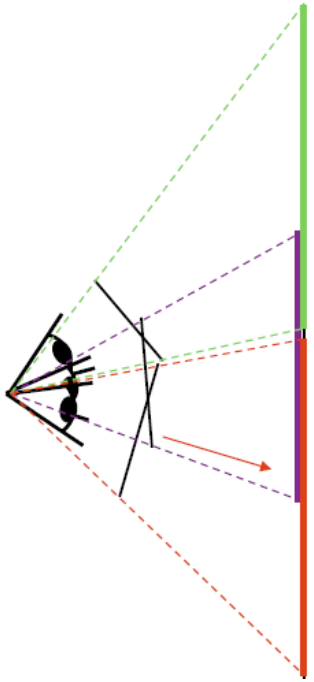
Appendix:

Immersive Panorama View

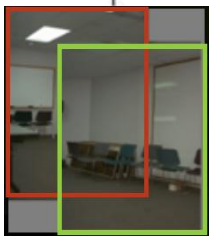
Image Mosaic

- The mosaic has a natural interpretation in 3D
  - the images are reprojected onto a common plane
  - the mosaic is formed on this plane
  - mosaic is a synthetic wide-angle camera

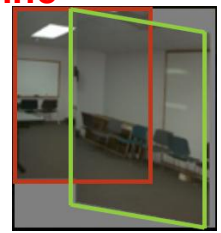




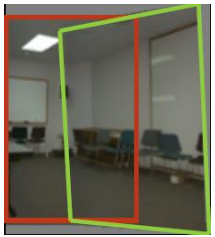
Translation



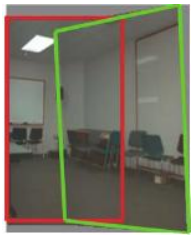
Affine



Perspective

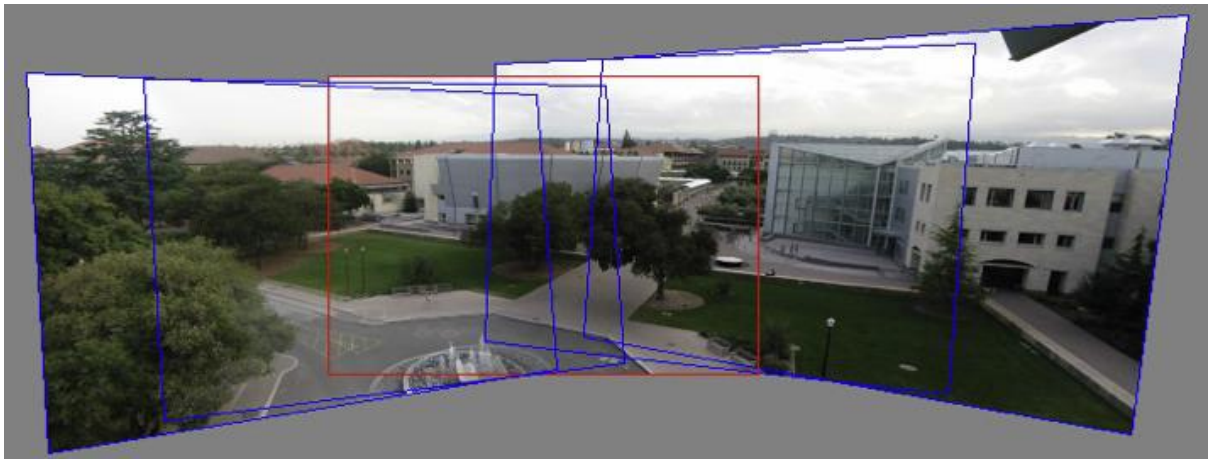


3D Rotations

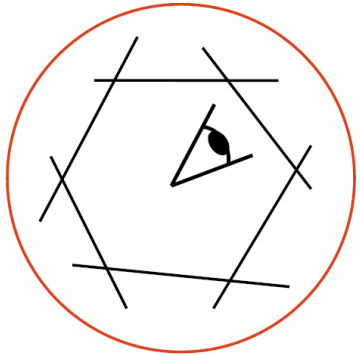


Perspective Stitching

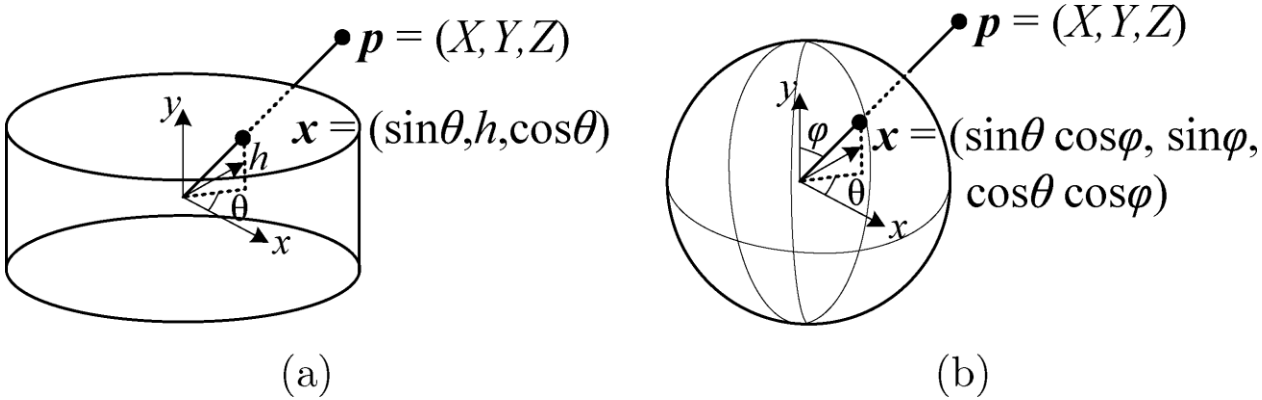
- Pick one image, typically the central view (red outline);
- Warp the others to its plane:
  - Pixel-based or feature-based registration;
  - Global alignment: bundle adjustment.
- Blend (global and local): alpha blending/Poisson blending/multi-band blending.



What if a 360 Field of View?



- Mapping to non-planar (e.g., cylindrical or spherical) surfaces;
- Lens distortion: radial or tangential;
  - Fisheye lens: a different model than polynomial models (radial);
  - Local registration for de-ghosting:
    - optic flow or spline-based,
  - Model of planar plus parallax;



Cylindrical Panoramas

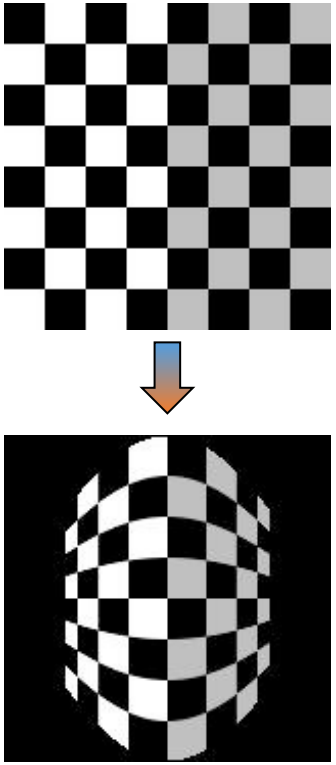
- Map 3D point (X,Y,Z) onto cylinder; $(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X, Y, Z)$
- Convert to coordinates on unit cylinder; $(sin\theta, h, cos\theta) = (\hat{x}, \hat{y}, \hat{z})$
- Convert to image coordinates on unwrapped cylinder.
  - A cylindrical image is a rectangular array. $(\tilde{x}, \tilde{y}) = (f\theta, fh) + (\tilde{x}_c, \tilde{y}_c)$

$$x' = s\theta = s\tan^{-1}\frac{x}{f}, \qquad x = f\tan\theta = f\tan\frac{x'}{s},$$

$$y' = sh = s\frac{y}{\sqrt{x^2+f^2}}, \qquad y = h\sqrt{x^2+f^2} = \frac{y'}{s}f\sqrt{1+\tan^2 x'/s} = f\frac{y'}{s}\sec\frac{x'}{s}$$

(a)

(b)



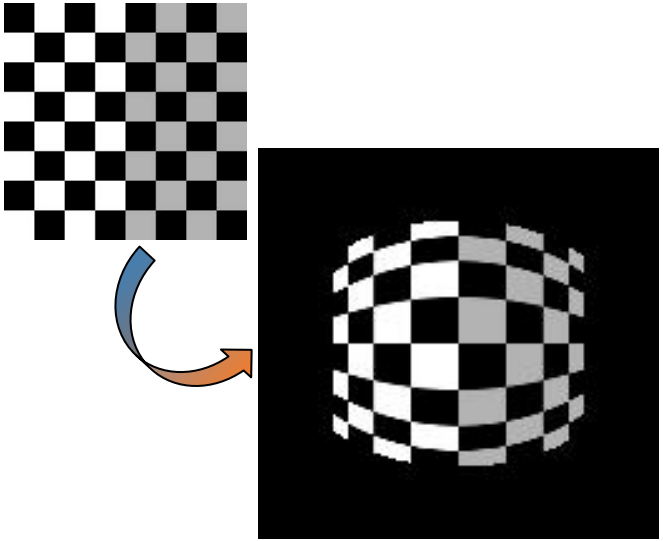
Spherical Projection

- Map 3D point (X,Y,Z) onto sphere;
- Convert to spherical coordinates;
- Convert to spherical image coordinates;
  - Map onto the unwrapped sphere.

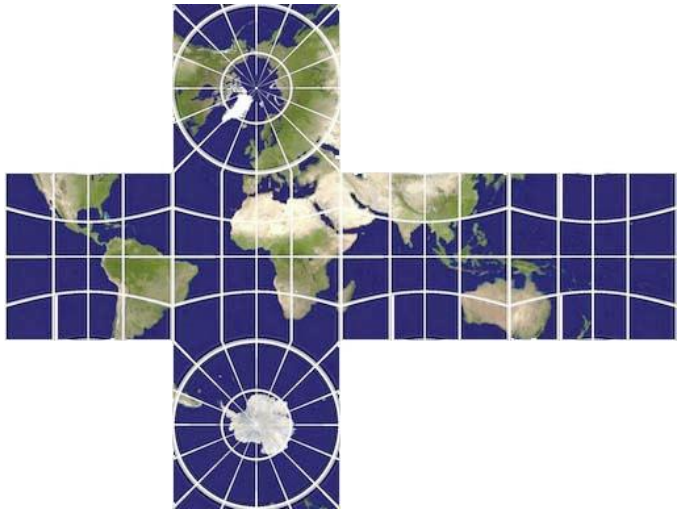
$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2 + Y^2 + Z^2}}(X, Y, Z)$$

$$(\sin\theta \cos\phi, \sin\phi, \cos\theta \cos\phi) = (\hat{x}, \hat{y}, \hat{z})$$

$$(\tilde{x}, \tilde{y}) = (f\theta, f\phi) + (\tilde{x}_c, \tilde{y}_c)$$



Spherical Projection



$$x' = s\theta = s\tan^{-1}\frac{x}{f},$$
$$y' = s\phi = s\tan^{-1}\frac{y}{\sqrt{x^2+f^2}},$$

while the inverse is given by

$$x = f\tan\theta = f\tan\frac{x'}{s},$$
$$y = \sqrt{x^2+f^2}\tan\phi = \tan\frac{y'}{s}f\sqrt{1+\tan^2 x'/s} = f\tan\frac{y'}{s}\sec\frac{x'}{s}$$

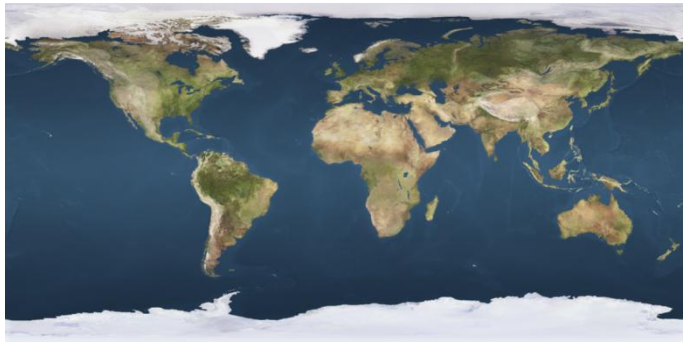
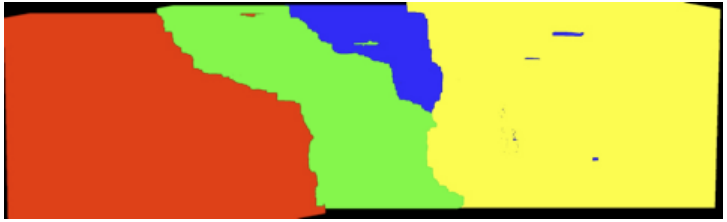


Image Blending

- Directly averaging the overlapped pixels results in ghosting artifacts
  - Moving objects, errors in registration, parallax, etc.
- Assign one input image each output pixel
  - One solution: optimal assignment can be found by graph cut
- Inconsistency btw pixels from different input images
  - Different exposure/white balance settings
  - Photometric distortions (e.g., vignetting)
- Poisson blending
  - Copy the gradient field from the input image;
  - Reconstruct the final image by solving a Poisson equation;
- Laplacian pyramid blending;
- Color correction and HDR for seam finding.

Image Blending



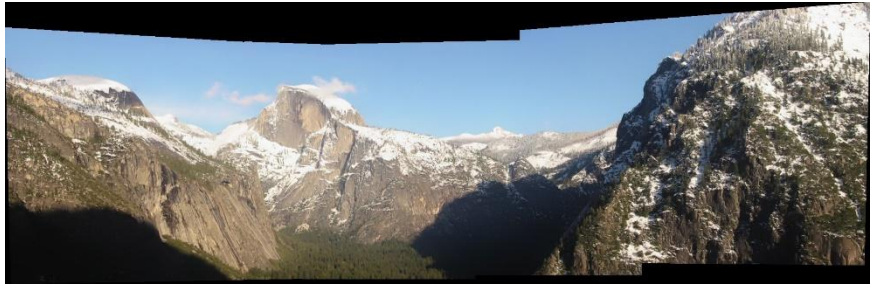
Graph cut

Poisson blending

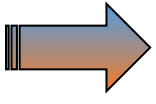
Color correction

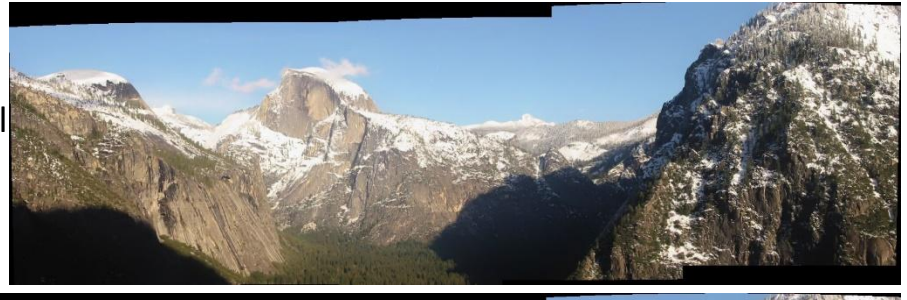
Examples of Panoramic View

Planar



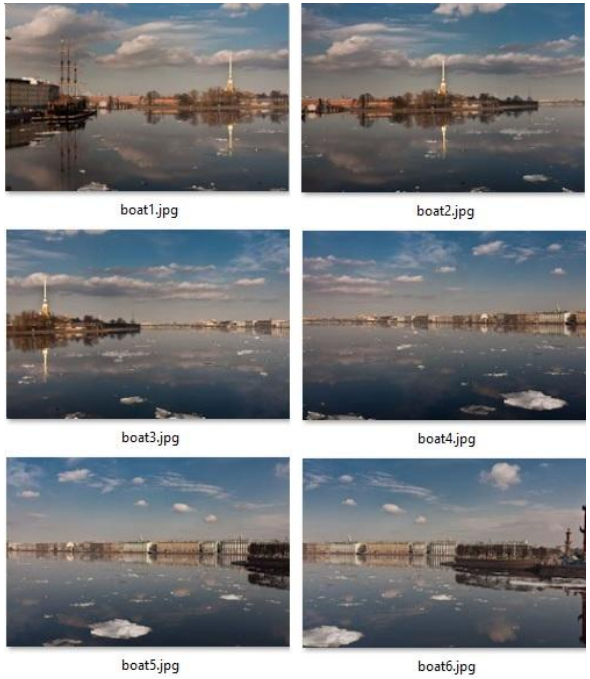
Cylindrical





Spherical

Examples of Panoramic View

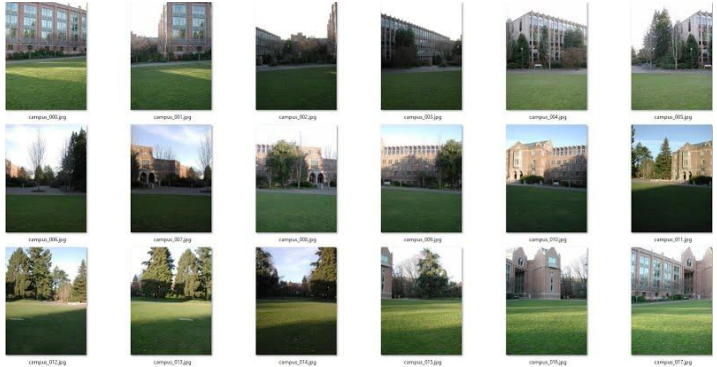


Planar

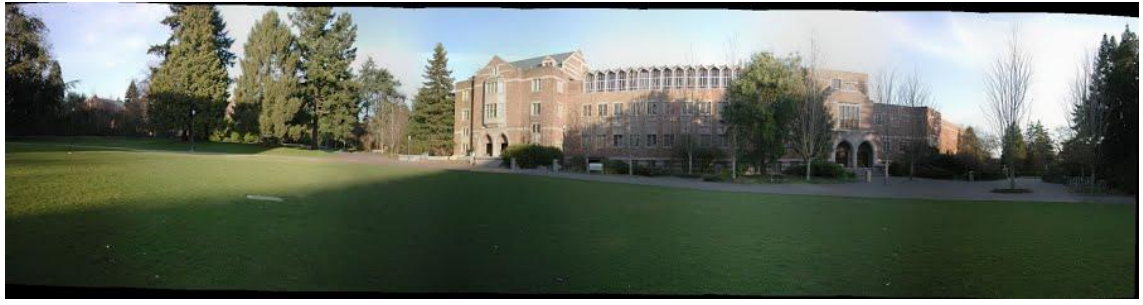
Cylindrical

Spherical

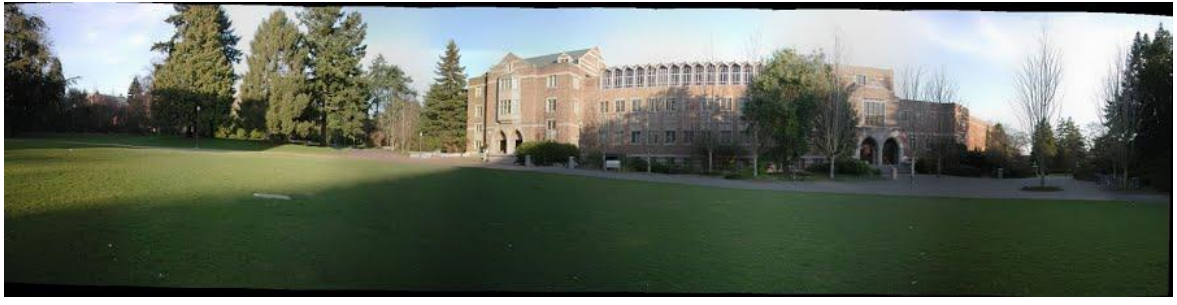
Examples of Panoramic View



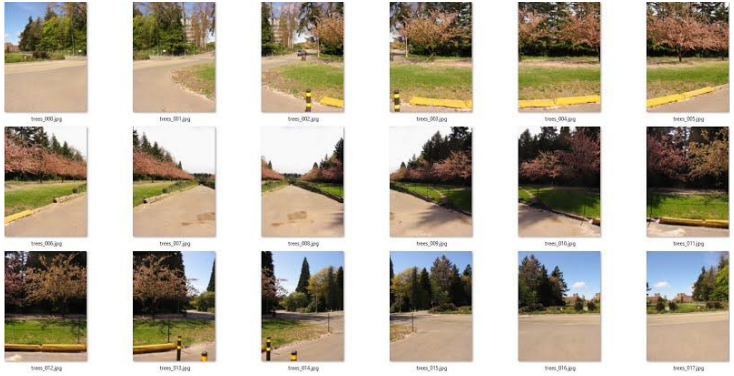
Cylindrical



Spherical



Examples of Panoramic View



Cylindrical

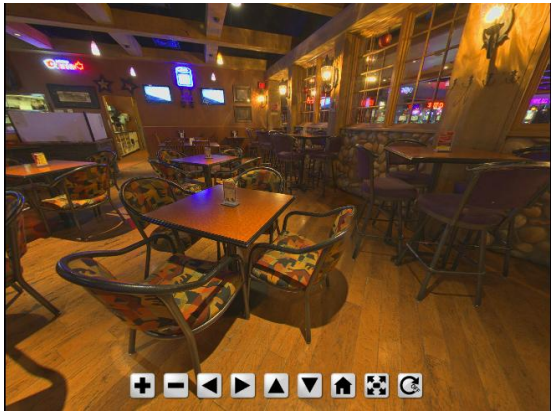
Spherical

View Navigation/Browsing

- Moviemaps, omnidirectional camera: registered to a street map;
- Object Movies: Quicktime VR;
- View interpolation: image-based modeling and rendering
  - View morphing in Photo Tourism system (used for 'PhotoSynth');
- Navigation with 6 DOF: path planning and fitting
  - Camera parameters: SfM;
  - View scoring: angular deviation, field of view, resolution;
  - Navigation control: free viewpoint, scene specific or optimized transition btw controls;
    - Discover orbit detection, panorama, and canonical images;
  - Scene viewer: choosing and warping the appropriate images in interaction with users;
    - Warping with proxy planes or 3-D model;
    - Render first by drawing the background, then the selected image.
  - Path planning: smooth transition optimized;
  - Appearance stabilization: geometric/photometric alignment.

360 Spherical View

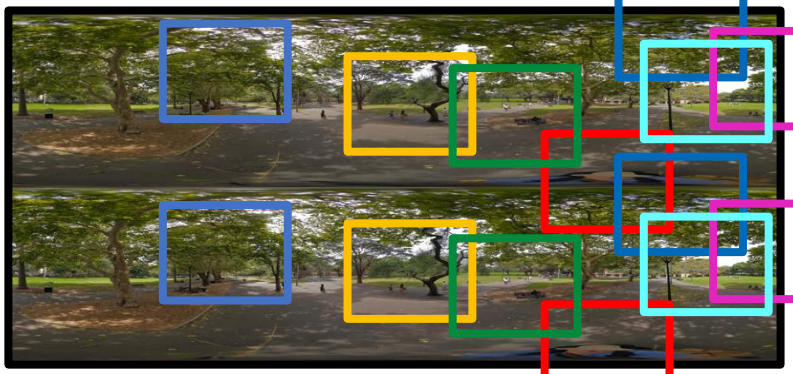
- Saved as equi-rectangular (or spherical) image: a 360x180 flat projection of a sphere;
- Html5/WebGL support viewing a spherical image by vertex + fragment shaders;
- 360 viewer/player can map the equi-rectangular image as texture with a spherical surface mesh for interactive viewing;
- YouTube supports 360 spherical video playback with 2 DoF (no zooming).



Spherical Video Player or Rendering

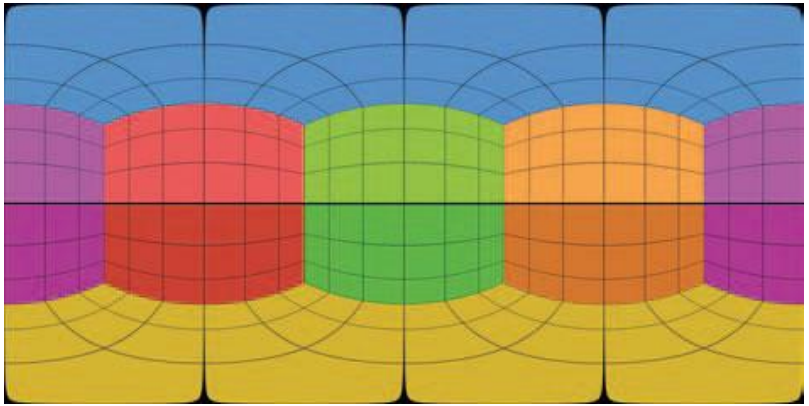
Spherical Stereo Video Playback

Format  OR 



Little Planet

- Wrap the equi-rectangular image (360x180 spherical panorama) into a sphere: a miniature model of a planet;
- Output: hyperbolic, stereographic, or squoculus.



Equi-rectangular panorama: Each major compass point is indicated by a separate colored square.

Stereographic, fov 130
Vertical distances in the side walls are distributed evenly. All details below the horizon are displayed as a relatively large planet in the center.

Hyperbolic, fov 130
The side walls appear to stretch toward the center and vertical structures are overly tall. The floor appears to be disproportionately small.

Squoculus, fov 120
Uneven distortion with emphasis on the foreground. Generally smaller in spite of an almost identical FOV.

Thanks!