RGB-D SLAM

姜翰青
商汤科技研究院
浙大-商汤三维视觉联合实验室

目录

RGB-D 传感器

RGB-D 传感器

- Kinect、Xtion、ZED、Tango等等

Kinect

Xtion

图片来自微软、华硕官网

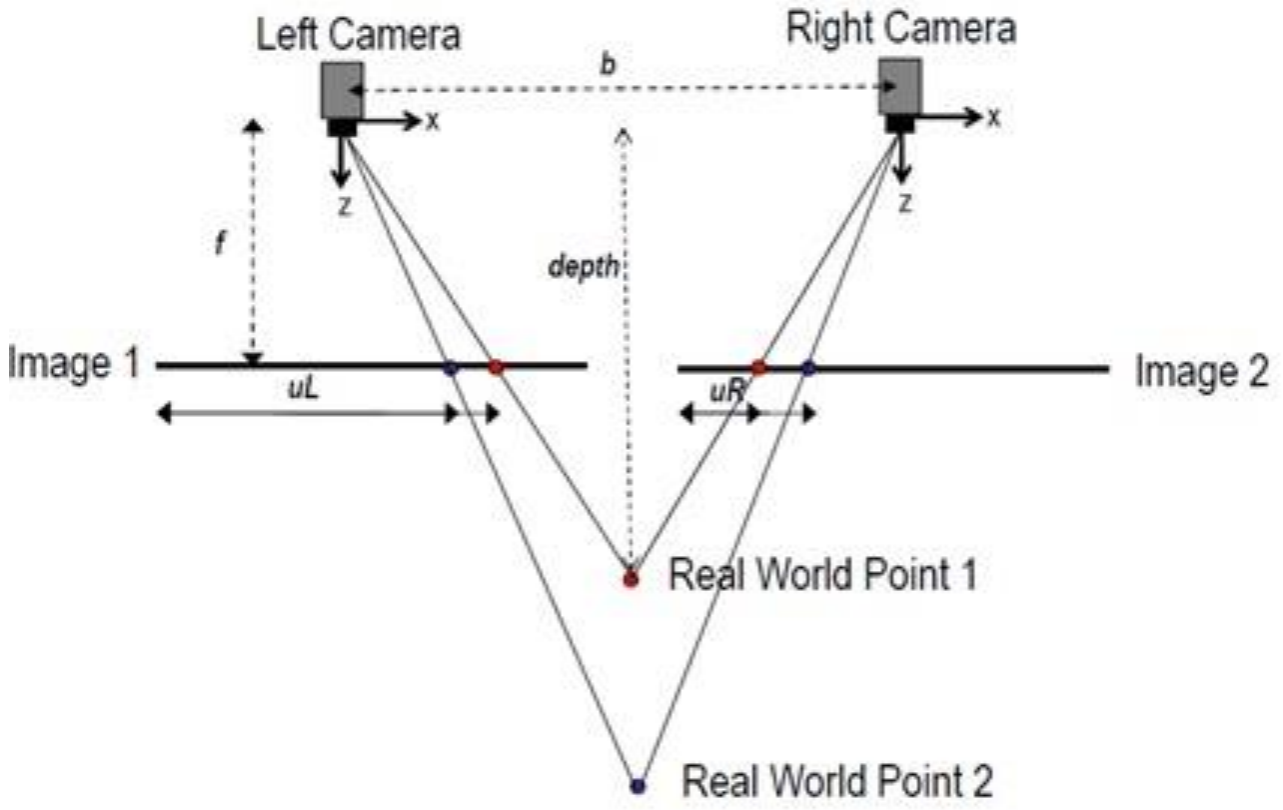
RGB-D 传感器

- 按工作原理分类：
  - 双目方案
    - ZED
    - Tango
  - 结构光方案
    - Kinect v1
    - Xtion
  - TOF方案
    - Kinect v2
    - RealSense

https://www.pcmag.com/news/356997/microsoft-is-ending-kinect-production

双目

- 立体匹配



https://my.oschina.net/abcijkxyz/blog/788647

结构光

- 主动投影已知图案的方法来实现快速鲁棒的匹配特征点

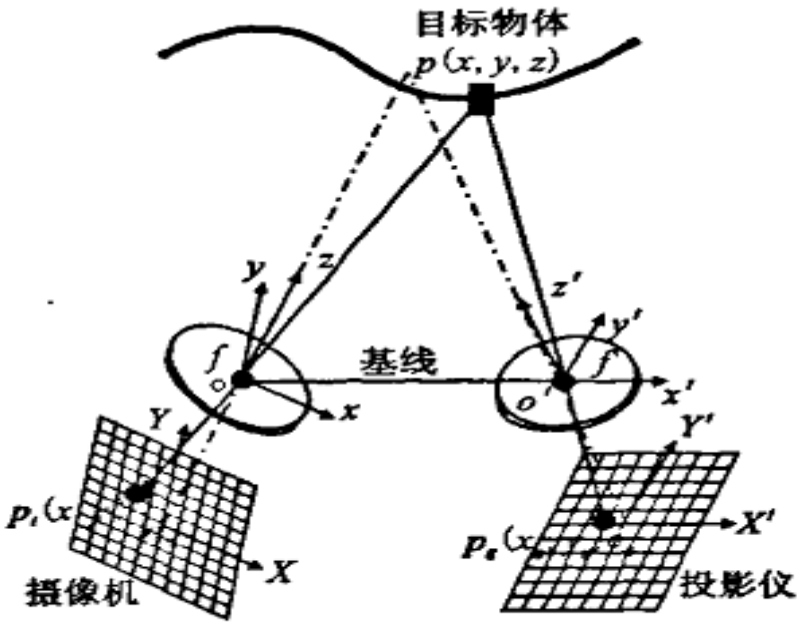
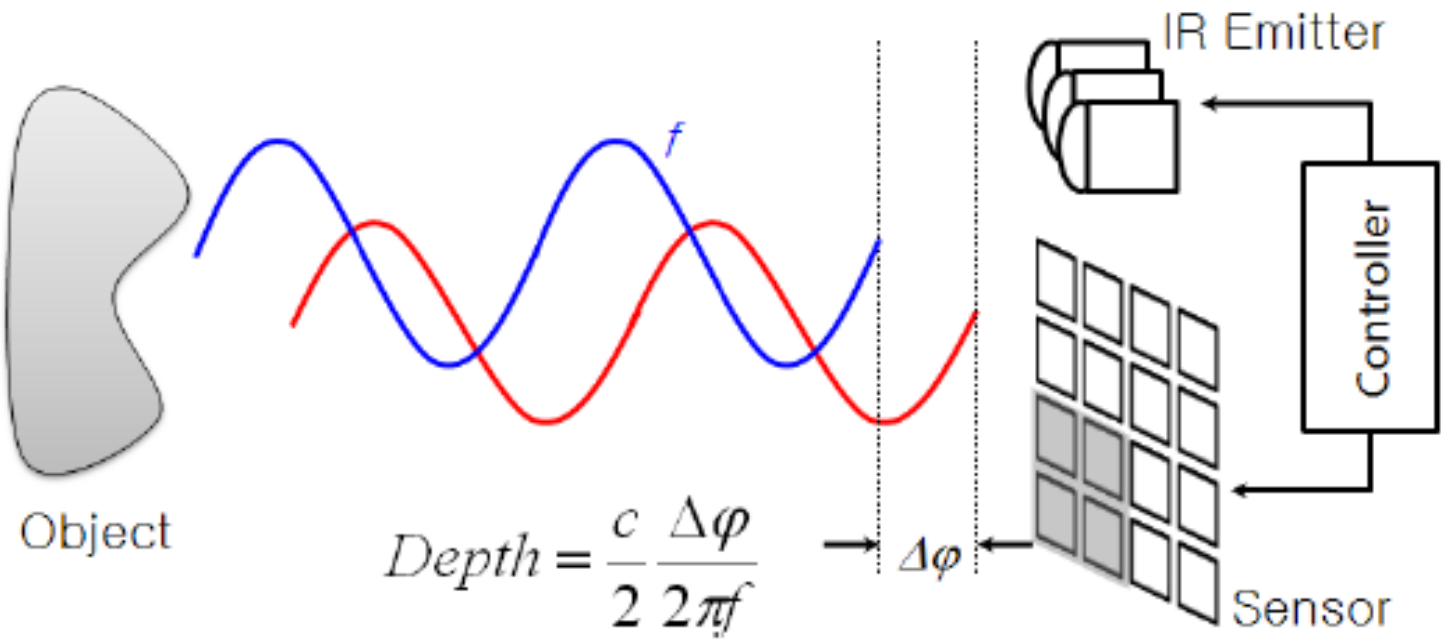


图 1　结构光三维视觉透视模型

陈彦军, 左旺孟, 王宽全, 等. 结构光编码方法综述[J]. 小型微型计算机系统, 2010 (9): 1856-1863.

TOF

- 测量发射与反射红外信号的相位延迟，计算每个sensor像素到目标物体的距离



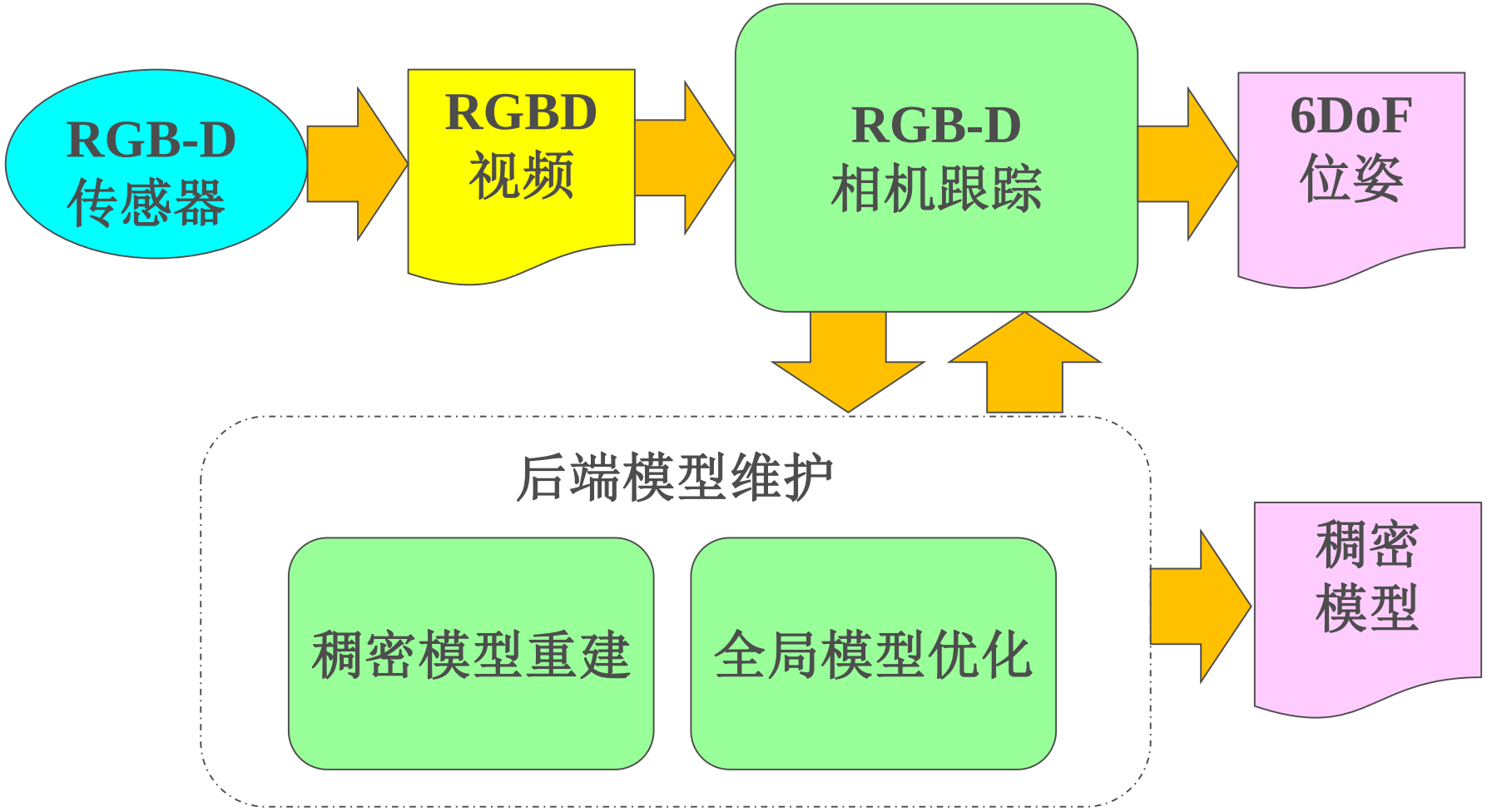
RGB-D 传感器

- 各方案的优劣
  - 双目：成本最低，但深度信息依赖纯软件算法得出，算法复杂度高，计算性能要求高，受光照等影响。
  - 结构光：技术成熟，深度图像分辨率可以做得比较高，但容易受光照影响，室外环境基本不能使用。
  - TOF：抗干扰性能好，视角更宽，深度图像分辨率较低，不适合高精度场合。受环境影响小，传感器芯片并不成熟，成本很高。

RGB-D 传感器

- 使用RGB-D 传感器优点是不需要计算特征点和描述子，就可以直接得到稠密或半稠密的深度图。
- 框架也相对传统SLAM 简单，可分为前端RGB-D相机跟踪与后端模型重建。

RGB-D SLAM算法流程



RGB-D 相机跟踪

RGB-D 相机跟踪

- 特征点法
- ICP
- RGB-D 对齐

RGB-D 相机跟踪

- 特征点法
  - 包含了传统的RGB信息，也可以使用特征点匹配求解相机位姿。
  - 一般场景都能提供丰富的特征点，场景适应性较好，能够利用特征点进行重定位。
  - 缺点是特征点计算法耗时；特征点利用到的信息太少，丢失了图像中的大部分信息和深度信息，在弱纹理环境下特征少等

RGB-D 相机跟踪

- 直接法
  - 包含了传统的RGB信息，也可以使用直接法求解相机位姿
  - 优点是不需要计算特征描述子，可以得到稠密或半稠密的地图；在特征缺失时也可正常使用
  - 缺点灰度不变假设在实际环境中不一定成立，要求像机运动速度不能太快，不能自动曝光等

RGB-D 相机跟踪

- 迭代最邻近算法（ICP）
  - 该算法重复进行选择对应关系点对， 计算最优刚体变换，然后应用变换，再寻找对应关系点对，计算新的最优变换，直到满足正确配准的收敛精度要求
  - 充分利用了深度图点云信息，利用点云的几何结构特征，不依靠RGB特征与光度

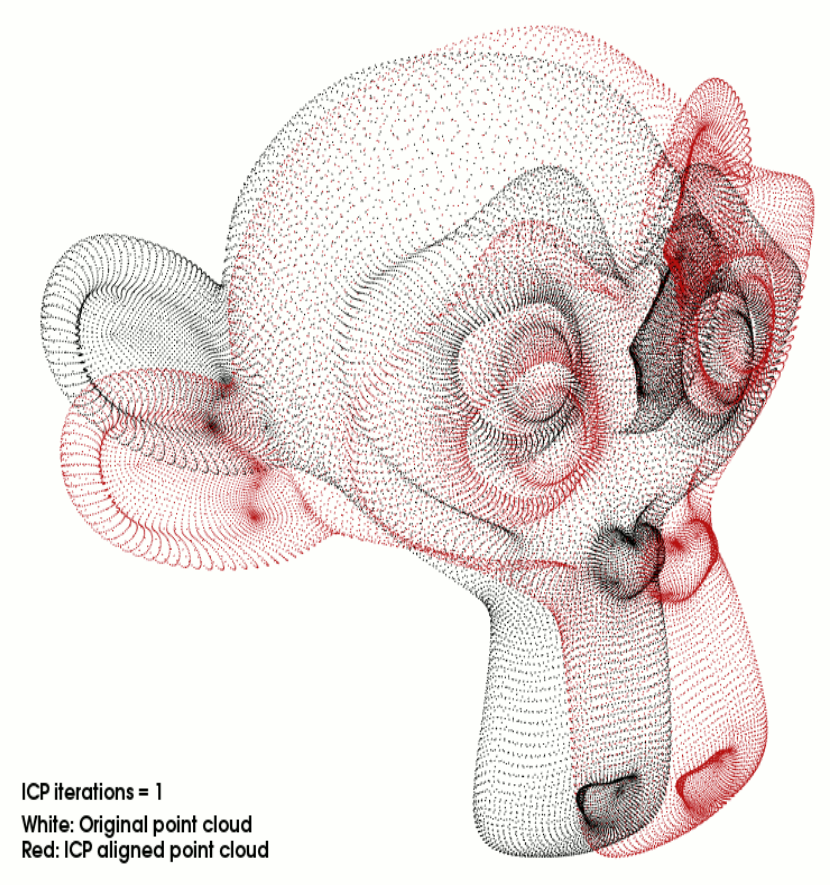
ICP（Iterative Closest Point)

- 目标

给定两组点云 P = $\{p_1, p_2, \cdots, p_m\}$， Q = $\{q_1, q_2, \cdots, q_n\}$，求解两组点云之间的变换 T 使得下式得到最优解：

$$T_{opt} = \arg\min_{\{T,A\}} \sum_{i=1}^{m} cost(T(p_i), q_{A(i)})$$

cost是代价函数，度量匹配的误差



http://pointclouds.org/documentation/tutorials/interactive_icp.php#interactive-icp

ICP

- 步骤
  - 固定相对变换 T，求解最优的对应关系 A，这一步即所谓的数据关联（data association）

$$A = \arg\min_{A} \sum_{i=1}^{m} cost(T(p_i), q_{A(i)})$$

  - 固定匹配关系 A，求解最优的相对变换 T：

$$T = \arg\min_{T} \sum_{i=1}^{m} cost(T(p_i) - q_{A(i)})$$

ICP

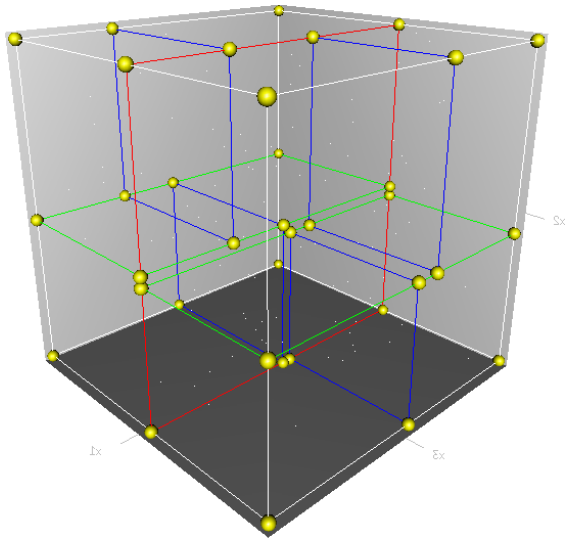
- 取样：选择一个或两个网格上的部分点作为样本
- 匹配：建立样本点的数据关联关系
- 置权：给关联样本点对设置权重
- 过滤：对不符合条件的关联点对做删除
- 误差度量：给关联点对设置误差度量
- 最优化：最优化误差度量

ICP Variants

- 取样
  - Uniform Sampling
  - Random Sampling
  - Normal Space Sampling
    - 选择样本点使得样本点法向散布最大

ICP Variants

- Data association
  - Closet point
    - K-D Tree
  - Normal shooting



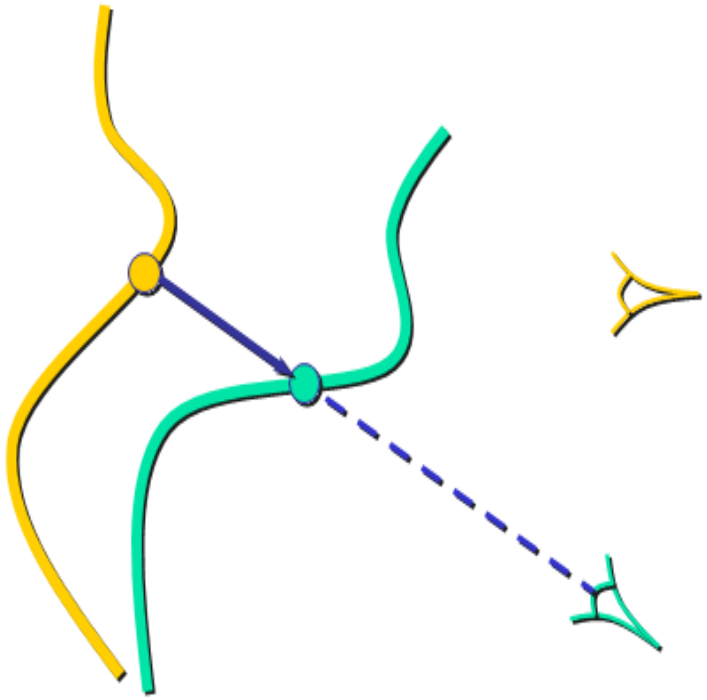
https://en.wikipedia.org/wiki/K-d_tree

ICP Variants

- Data association
  - Project
  - project and walk

$$x(u_d, vd, 1) = KMX_s$$

K 是相机内参数，X_s是待求匹配点的三维空间坐标，x是匹配点像素坐标，M 是两帧之间的位姿变换，在 ICP 算法中通过不断迭代求得



ICP Variants

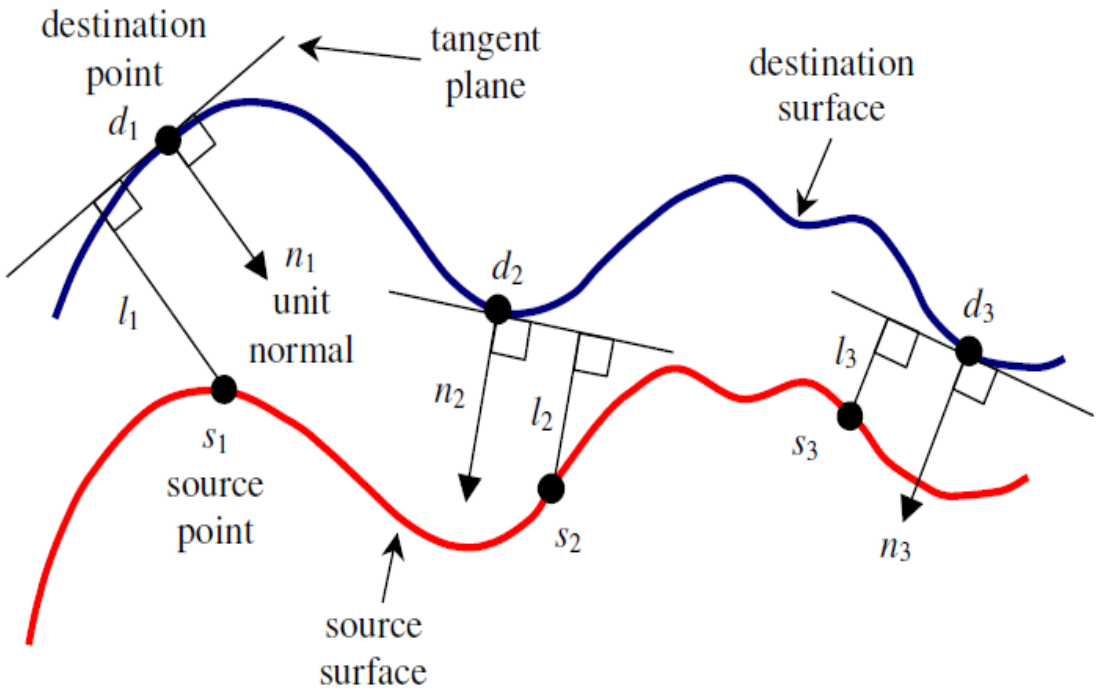
- 权值设置
  - Constant weight
  - Linear with distance
  - Compatibility of normals
  - Uncertainty
    - 基于相机噪声的权值设置

ICP Variants

- 过滤准则
  - 距离超过固定门限的关联点对删除
  - 距离最大的n%关联点对删除
  - 距离超过2.5倍标准差的关联点对删除

ICP Variants

- 误差度量
  - point-to-point
  - point-to-plane



Low K L. Linear least-squares optimization for point-to-plane icp surface registration[J]. Chapel Hill, University of North Carolina, 2004, 4: 1.

ICP

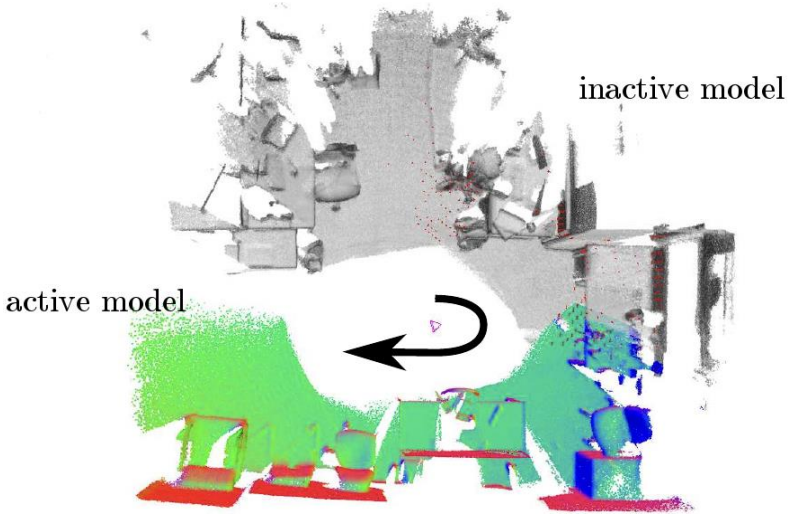
- 缺点
  - 对位姿初值很敏感，需要有较好的初值
  - 只利用了深度信息，没有利用RGB信息

ICP算法举例：KinectFusion

- Convert the depth image into
  - Vertex map
  - Normal map
- Pyramid ICP
  - Projective data association
  - Remove outliers by normal direction and distance
  - Minimize point to plane distance

ICP算法举例：ElasticFusion

- Frame-to-model ICP based on surfels
- A time window divides the model into active/inactive part
  - A surfel is active if it is most recent update time t
- Only active surfels are used to do camera pose estimation and depth map fusion



RGB-D 对齐

- 通过最小化逆深度误差和光度误差来求解两帧之间的相对相机姿态
  - 几何误差

$$E_z = \sum_{x_i} \rho_z \left(\frac{1}{z(X_j)} - Z_j(x_j) \right)$$

$z(X_j)$ 代表点X_j 在第i帧上的深度，$Z_j(x_j)$ 代表第j帧的深度图上点X_j的投影位置x_j对应的深度。ρ_z是相应的鲁棒化函数。

RGB-D 对齐

- 相对于ICP算法，RGB-D对齐不仅考虑到了深度信息的几何误差，还考虑了光度误差
- 光度误差

$$E_I = \sum_{x_i} \rho_I \left(I_i(x_i) - I_j(x_j) \right)$$

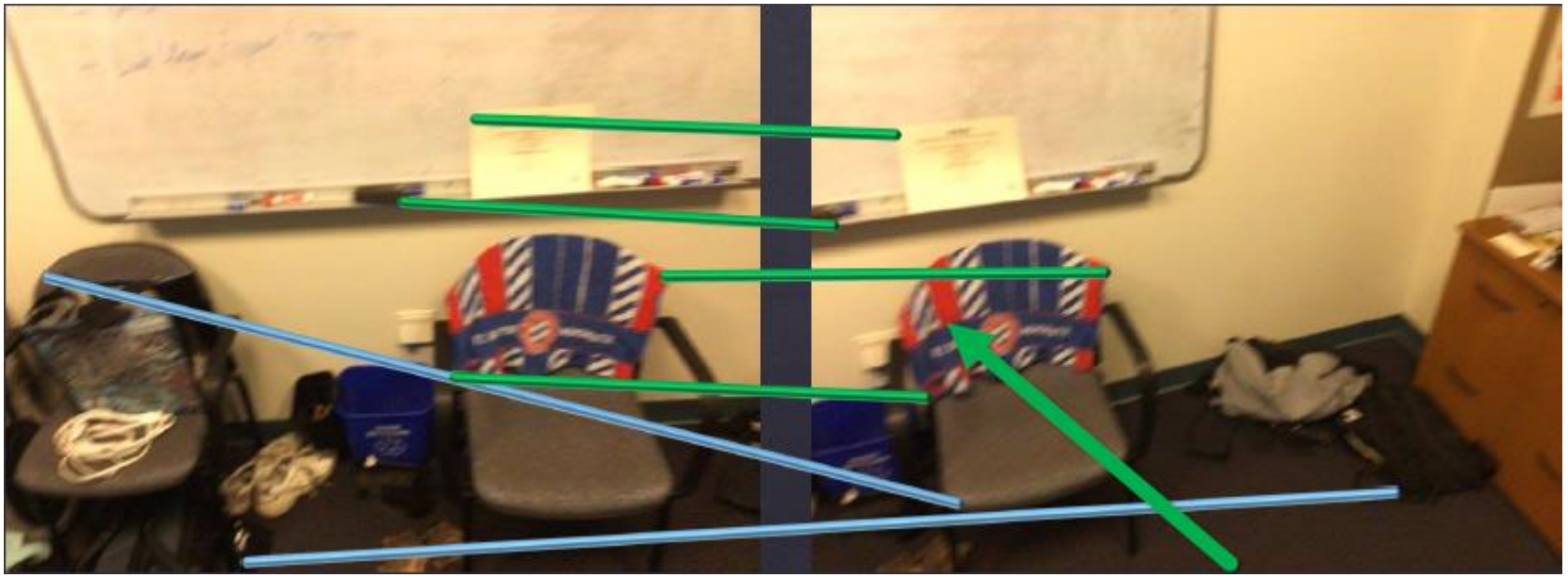
$I_i(x_i)$代表第i帧上x_i对应的光度。ρ_I 是相应的鲁棒化函数

- 总能量函数

$$\arg\min_T E_{align} = E_z + \alpha * E_I$$

RGB-D对齐算法举例：BundleFusion

- Correspondence Filtering



Consistent Correspondence

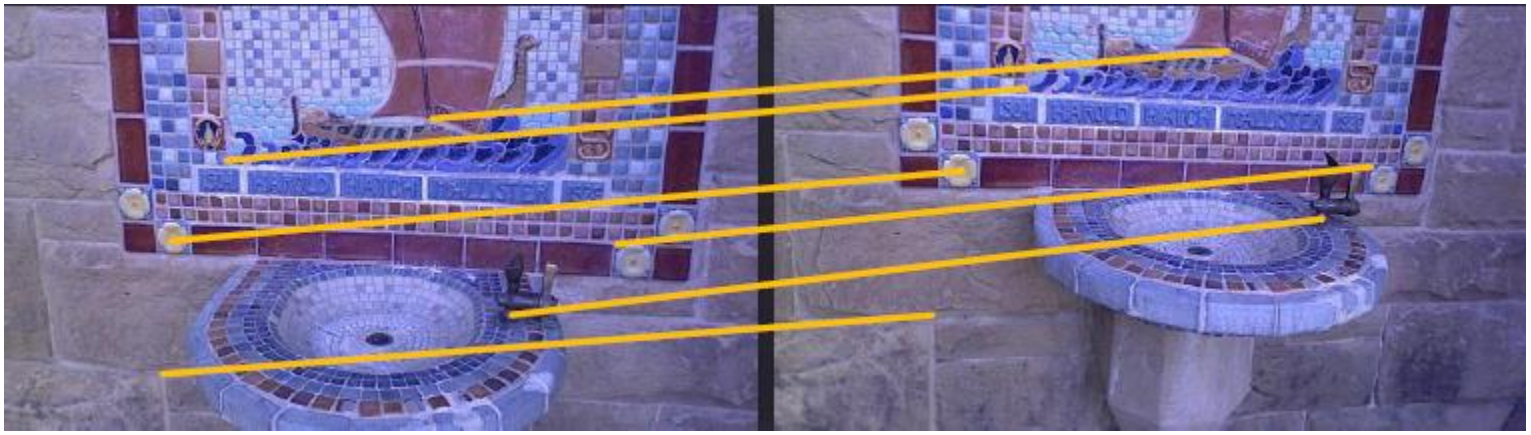
$$Reproj\ Err(i,j) = \sum_{k}^{\#pixels} \left| \left(p_k - T_i^{-1} T_j \pi_d^{-1} \left(D_j \left(\pi_d \left(T_j^{-1} T_i p_k \right) \right) \right) \right) \cdot n_k \right|$$

RGB-D对齐算法举例：BundleFusion

- Sparse-to-Dense Optimization

$$E(T) = w_{sparse} E_{sparse}(T) + w_{dense} E_{dense}(T)$$

$$E_{sparse}(T) = \sum_{i,j}^{\#frames} \sum_{k}^{\#corresp.} \left\| T_i p_{ik} - T_j p_{jk} \right\|_2^2$$



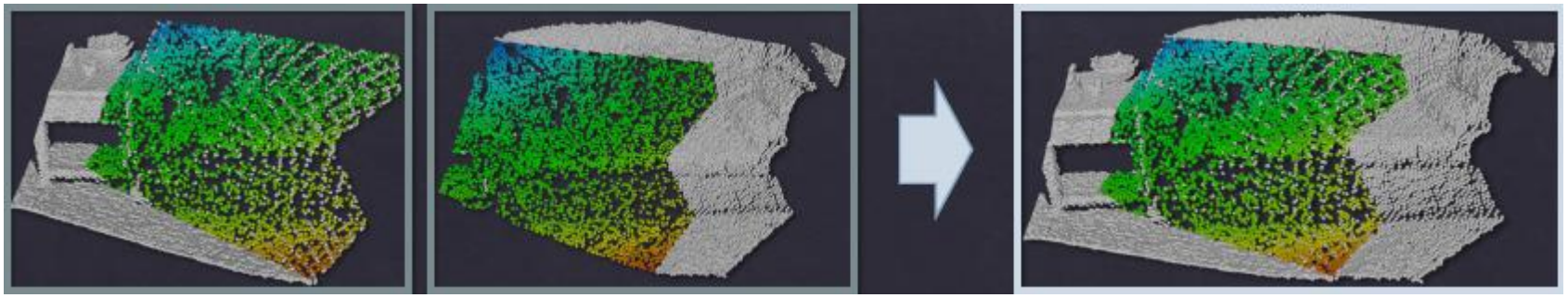
RGB-D对齐算法举例：BundleFusion

- Sparse-to-Dense Optimization

$$E(T) = w_{sparse} E_{sparse}(T) + w_{dense} E_{dense}(T)$$

$$E_{dense}(T) = w_{depth} E_{depth}(T) + w_{color} E_{color}(T)$$

$$E_{depth}(T) = \sum_{i,j}^{\#\text{frames}} \sum_{k}^{\#\text{pixels}} \left\| \left(p_k - T_i^{-1} T_j \pi_d^{-1} \left(D_j \left(\pi_d \left(T_j^{-1} T_i p_k \right) \right) \right) \right) \cdot n_k \right\|_2^2$$



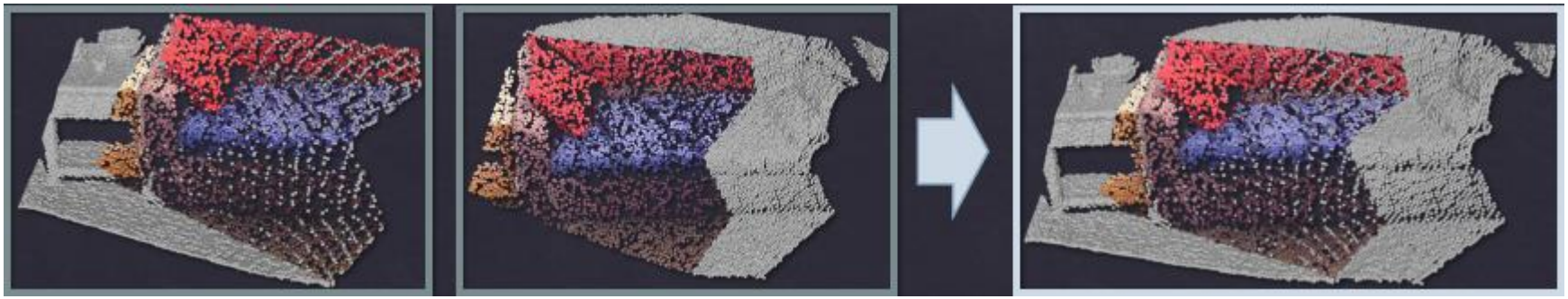
RGB-D对齐算法举例：BundleFusion

- Sparse-to-Dense Optimization

$$E(T) = w_{sparse} E_{sparse}(T) + w_{dense} E_{dense}(T)$$

$$E_{dense}(T) = w_{depth} E_{depth}(T) + w_{color} E_{color}(T)$$

$$E_{color}(T) = \sum_{i,j}^{\#\text{frames}} \sum_{k}^{\#\text{pixels}} \left\| \nabla I(\pi_c(p_k)) - \nabla I(\pi_c(T_j^{-1} T_i p_k)) \right\|_2^2$$



模型表示与重建

模型表示与重建

- 渐进式在线重建
- 可以支持在线高效更新
  - 网格表达不适合在线更新
- 两种常见模型表示
  - TSDF
  - Surfel
- 一些代表性算法
  - KinectFusion
    - Kintinuous
    - InfiniTAM
  - ElasticFusion
  - BundleFusion

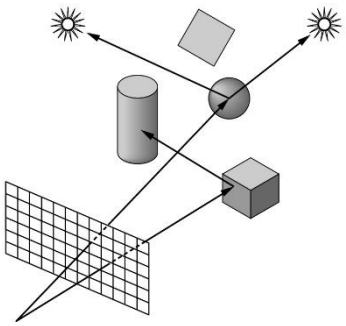
TSDF

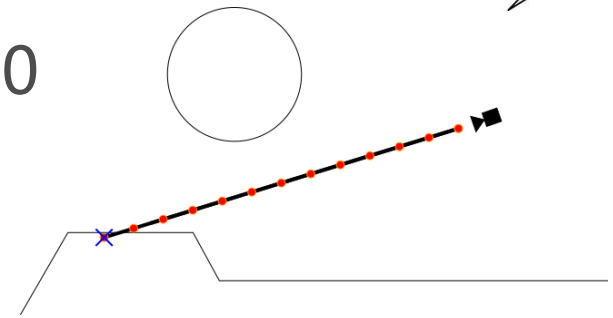
- 带符号距离函数(Signed Distance Function):
  - sdf: $R^3 \rightarrow R$
  - 对于点 p, sdf(p) 记录了把它映射到最近表面的距离
  - sdf(p) 为 0 的集合即为表面
- 截断带符号距离函数(Truncated Signed Distance Function)
  - 实际上，对于三维重建而言，我们仅仅需要找到SDF为0的点，因此只有表面附近的点是有用的，因此通常会对SDF进行截断，不再存储离表面较远处点的SDF

TSDF: rendering by raycasting

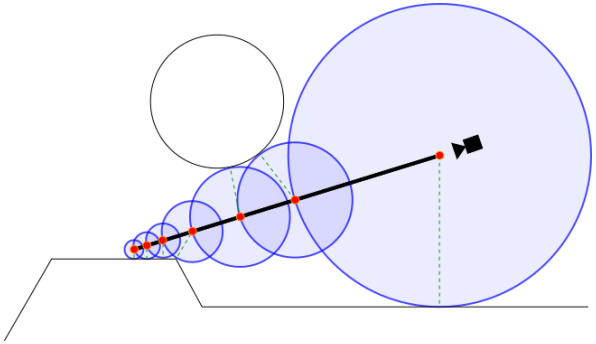
- 渲染一个像素x时
  - 从相机中心投射一条射线，寻找它和模型

  的交点
  - 这个交点的TSDF函数值为0
- 一个简单的实现:
  - 采用固定步长





- 但是我们可以用TSDF来加速:
  - 渲染时间近似与图像的尺寸线性相关

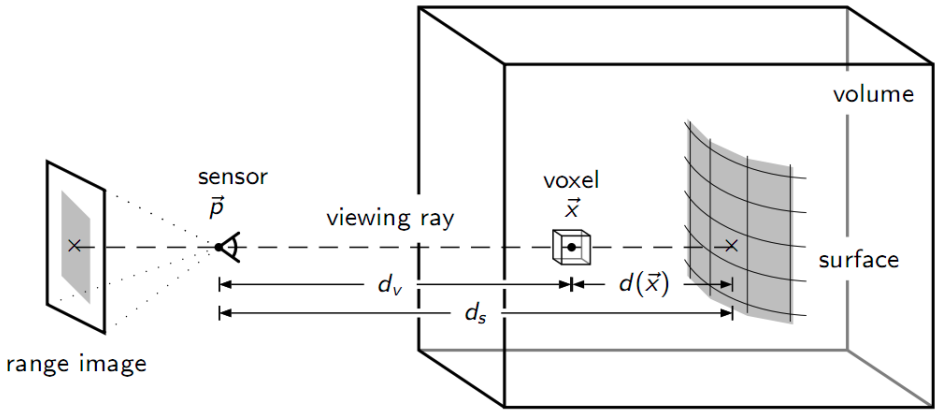


http://flafla2.github.io/2016/10/01/raymarching.html

基于TSDF的KinectFusion

基于TSDF的KinectFusion – Integration

- For each voxel
  - Project the voxel to image space
  - Compute the TSDF observation d(x)
  - Update by weighted sum



$$D(x) \leftarrow \frac{D(x) * W(x) + d(\bar{x}))}{1 + W(x)}$$

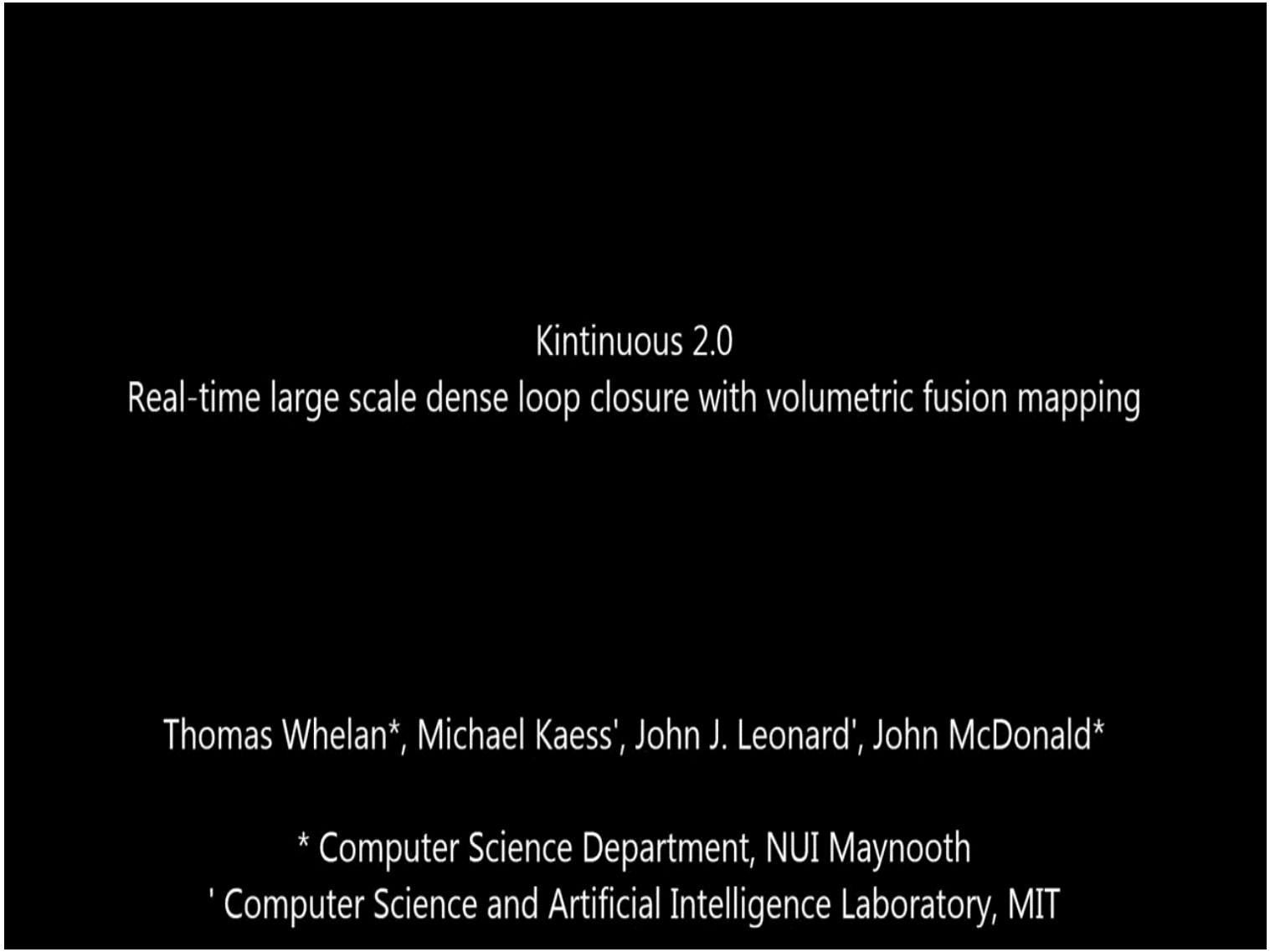
$$W(x) \leftarrow W(x) + 1$$

http://ifp.uni-stuttgart.de/lehre/diplomarbeiten/korcz/index.html

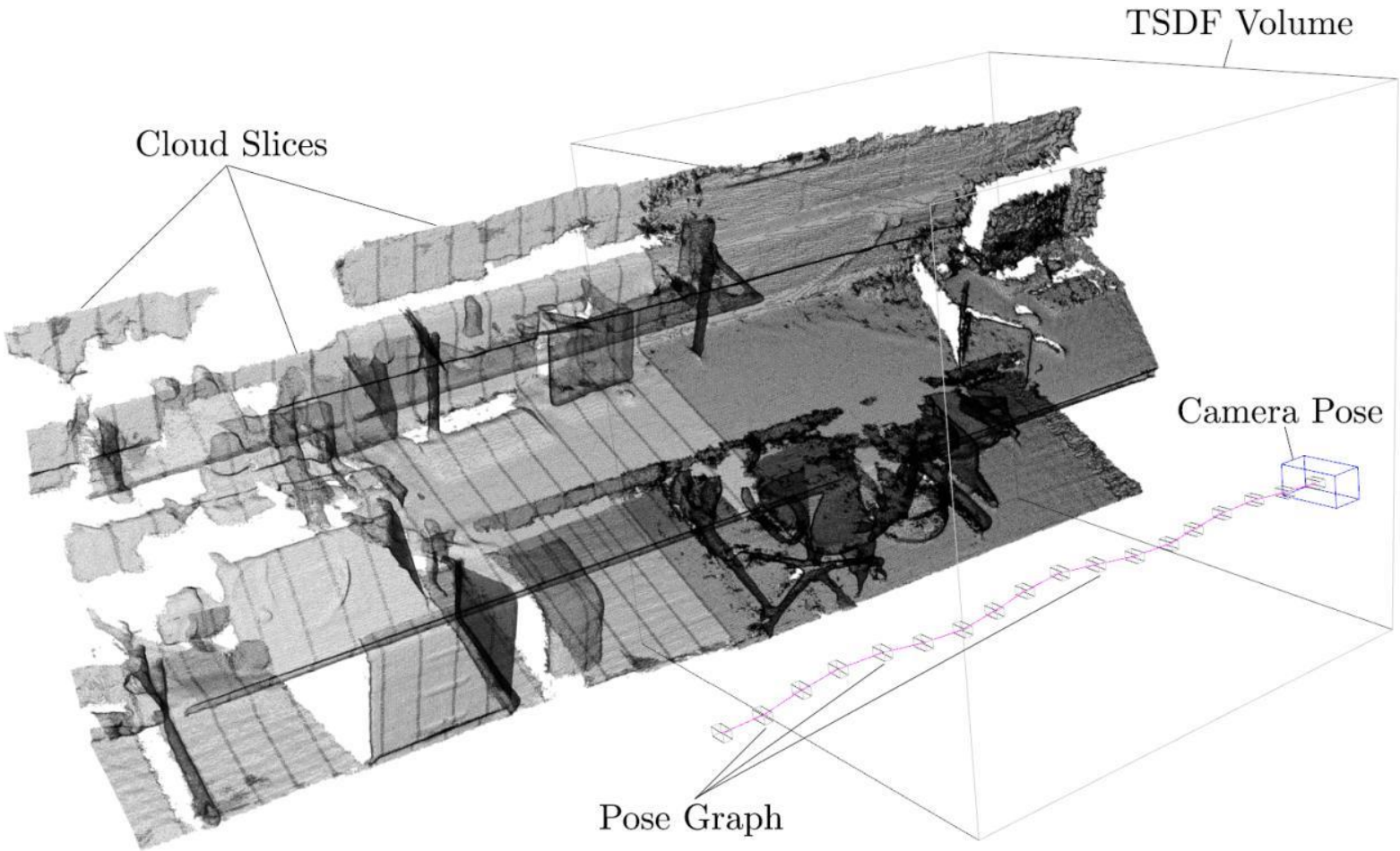
基于TSDF的KinectFusion - Limitations

- Only use depth image
  - Robust to lighting
  - Works on dark environment
  - But not works on structureless scene
- High memory consumption
  - Each voxel takes 8 byte
  - A 2mx2mx2m scene, takes ~0.5GB at 5mm resolution (geometry only, double if add color)

TSDF的改进：Kintinuous



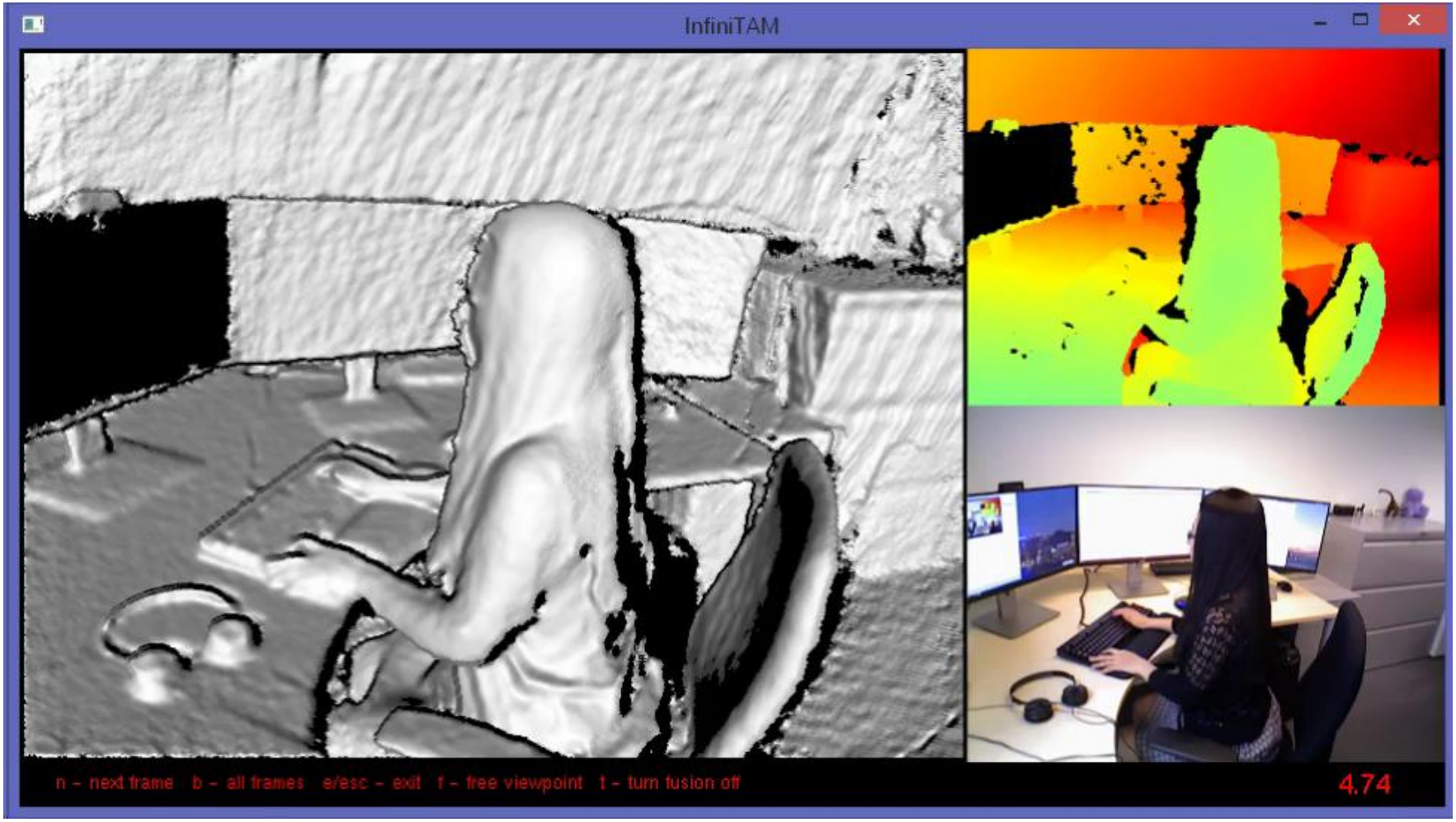
TSDF的改进：Kintinuous



Whelan T, Kaess M, Fallon M, et al. Kintinuous: Spatially extended kinectfusion[J]. 2012.

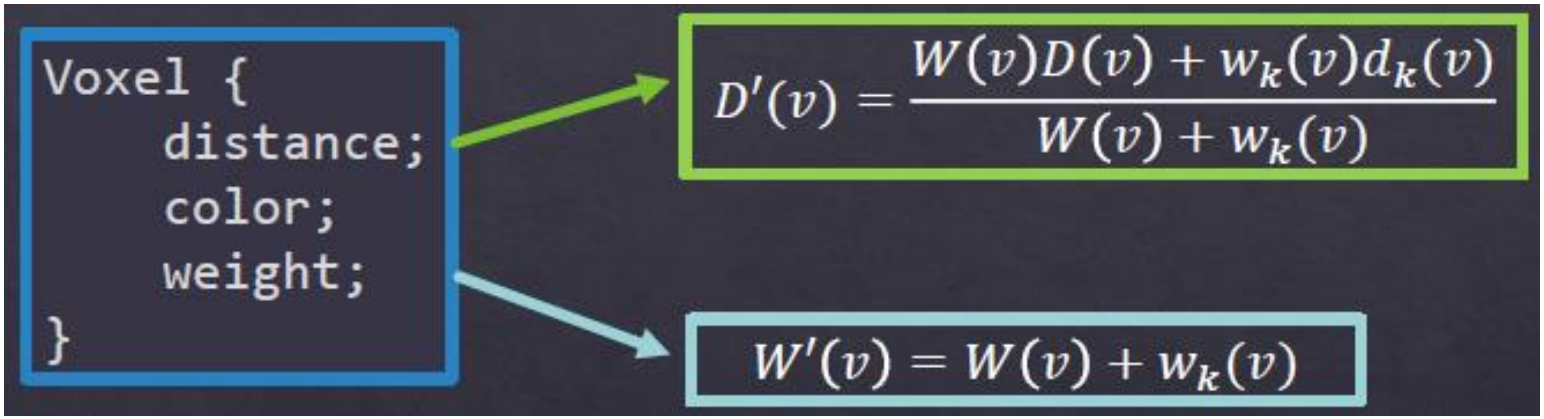
TSDF的改进：InfiniTAM

- Voxel Hashing



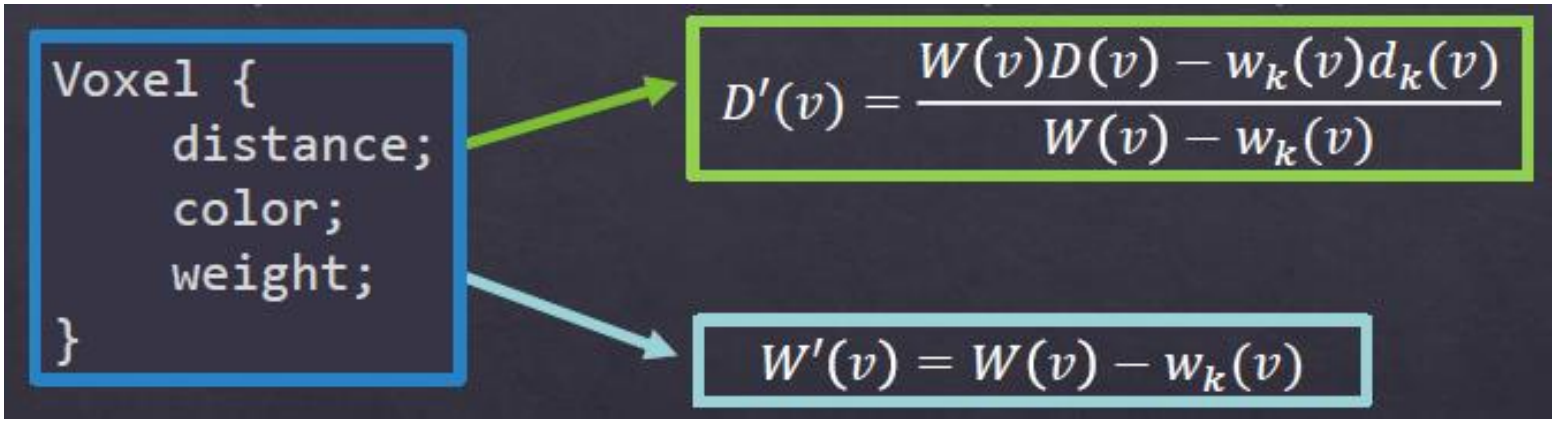
基于TSDF Re-integration的BundleFusion

- On-the-fly Scene Updates
  - Surface integration [Curless and Levoy 96]



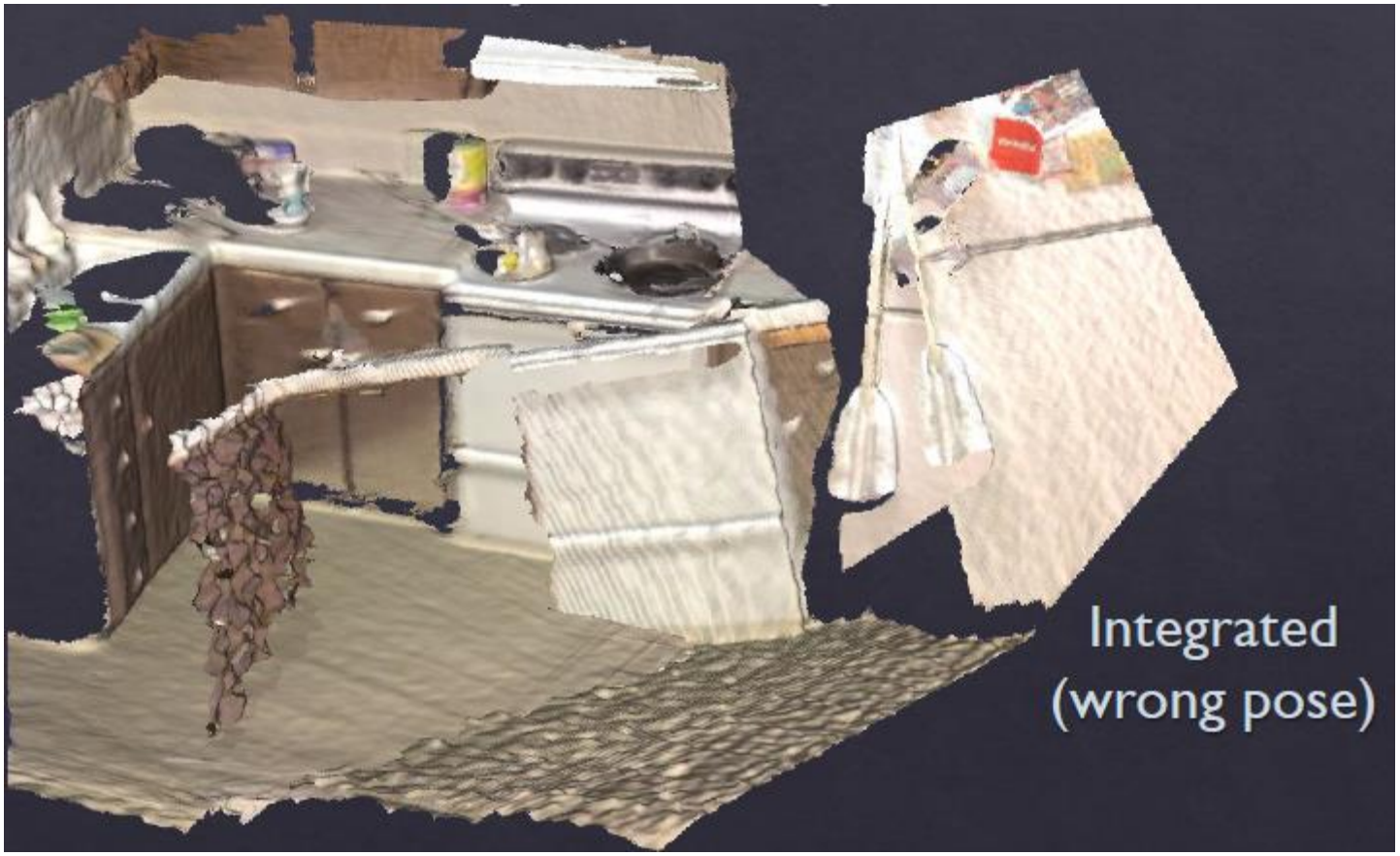
基于TSDF Re-integration的BundleFusion

- On-the-fly Scene Updates
  - Surface De-integration



基于TSDF Re-integration的BundleFusion

- On-the-fly Scene Updates



基于TSDF Re-integration的BundleFusion

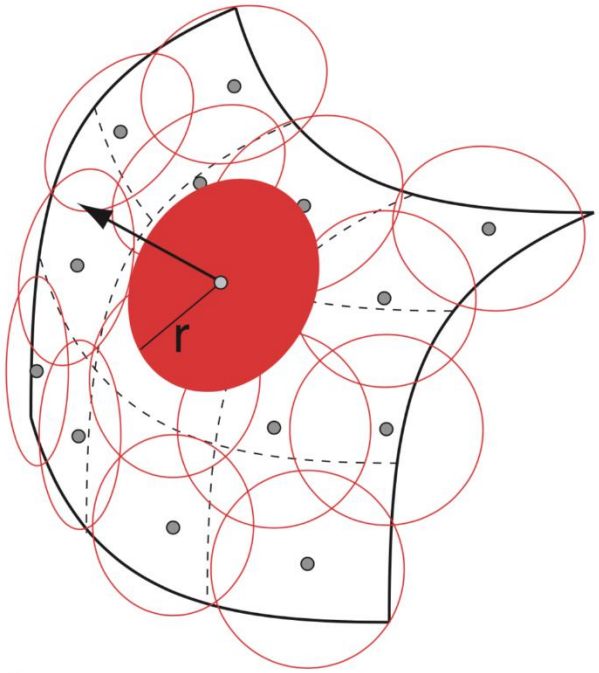
- On-the-fly Scene Updates

基于TSDF Re-integration的BundleFusion

- On-the-fly Scene Updates

Surfel

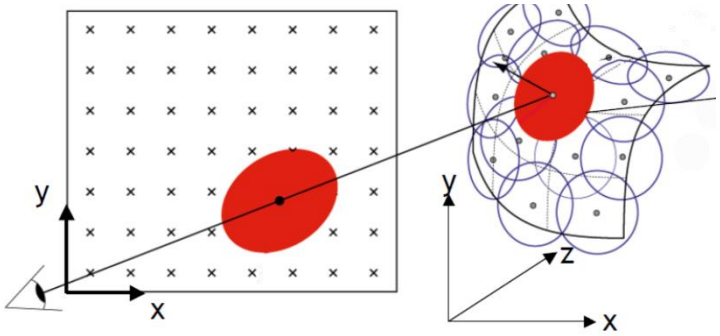
- Surfel: 表面的一个小的面元
- 几何属性
  - 位置
  - 法向
  - 半径 ....
- 反射属性
  - 颜色 ....
- model:面元的集合



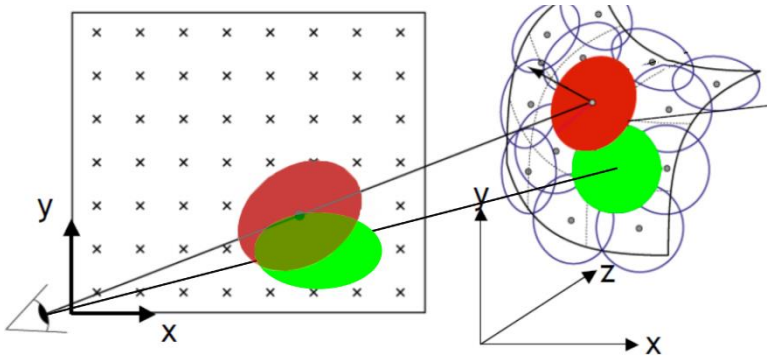
Pfister H, Zwicker M, Van Baar J, et al. Surfels: Surface elements as rendering primitives[C]//Proceedings of the 27th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 2000: 335-342.

Surfel: splat rendering

- 渲染一个Surfel是非常简单的:



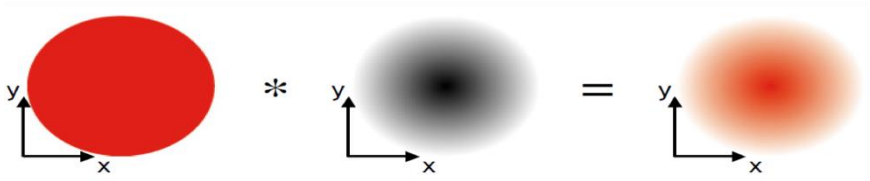
- 直接单独渲染一个Surfel是不行的，因为面元之间可能有重叠!



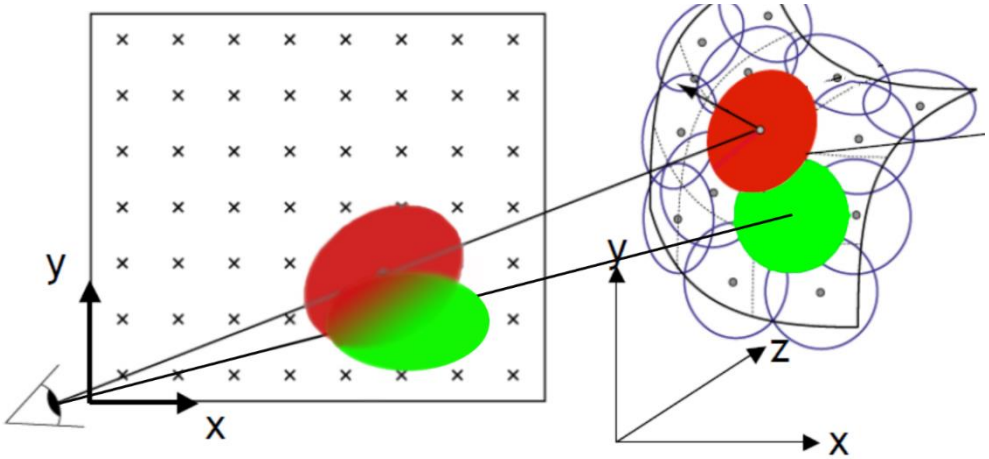
http://www.cs.rug.nl/~roe/courses/acg/gpurendering

Surfel: splat rendering (cont.)

- 使用高斯核卷积

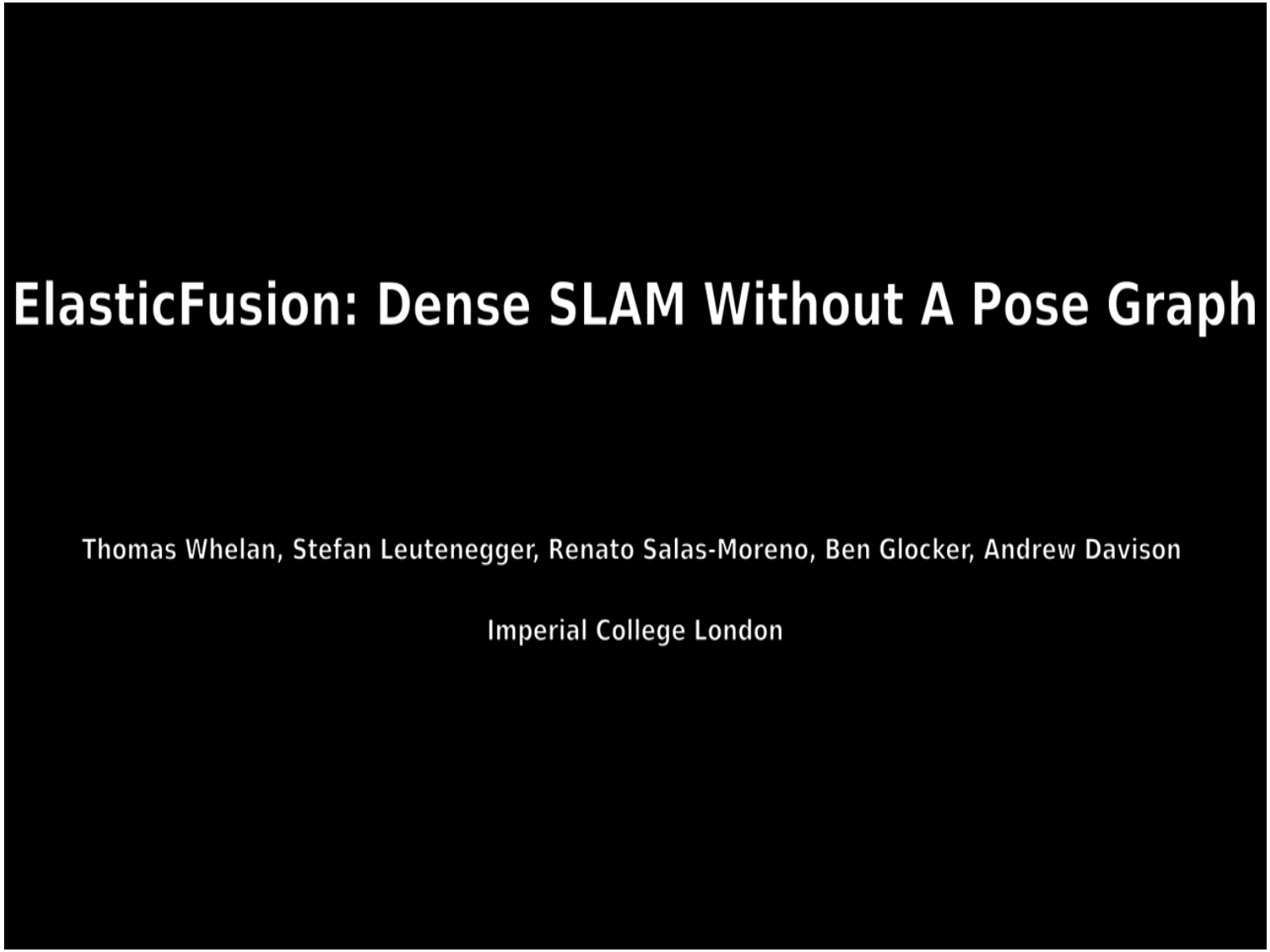


- 取加权和



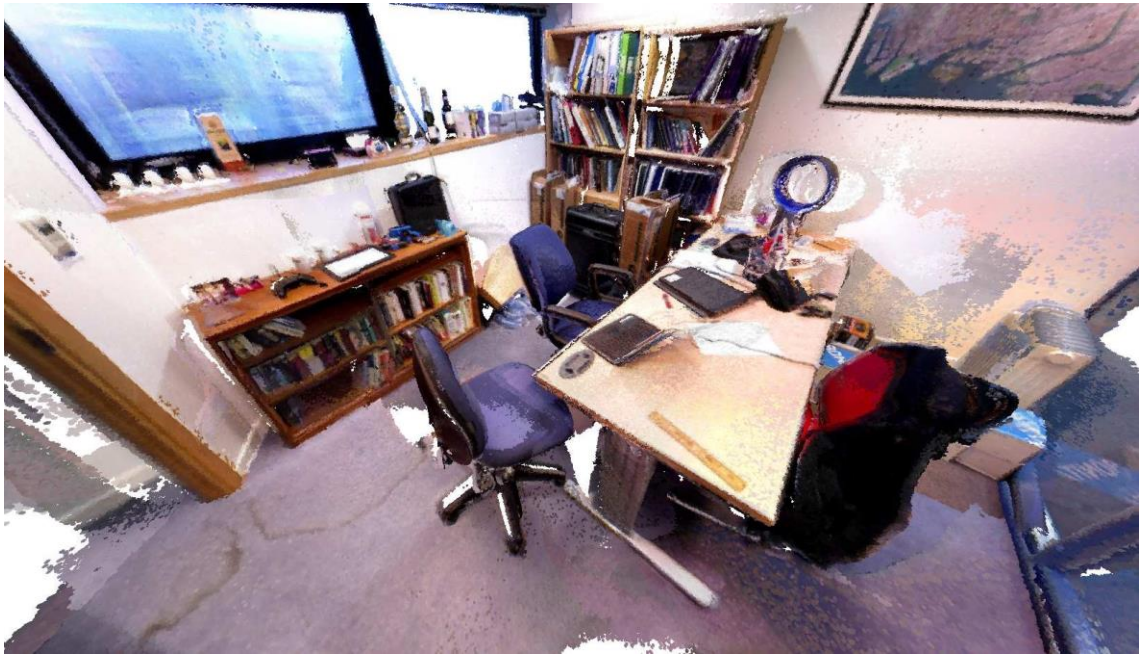
http://www.cs.rug.nl/~roe/courses/acg/gpurendering

基于Surfel的ElasticFusion



基于Surfel的ElasticFusion

- Surfel based representation
- Support loop closure, online model adjustment



Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, Andrew Davison. Elasticfusion: Dense slam without a pose graph[C]. Robotics: Science and Systems, 2015.

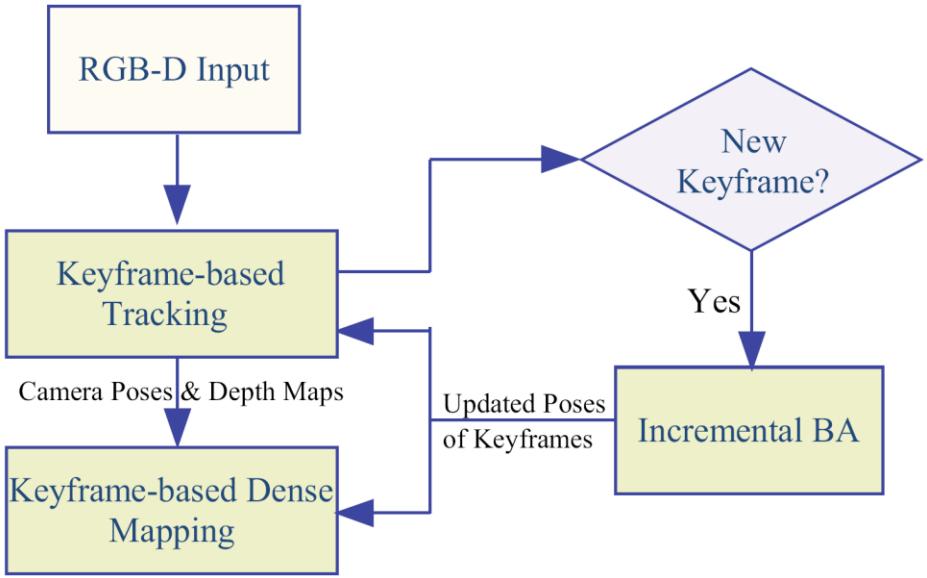
基于Surfel的ElasticFusion

- The model is an unordered list of Surfels
- Each surfel contains:
  - Position p
  - Normal n
  - Radius r
  - Color c
  - Confidence (how often it’ s observed)
  - Create timestamp t0, most recent update timestamp t

RKD-SLAM

RKD-SLAM系统框架

- 非常快速鲁棒的基于RGB-D的跟踪方法(单CPU下约70-200 fps)
- 非常快速的增量集束调整算法
- 非常高效的基于关键帧的深度表达和融合方法



支持快速运动、回路闭合、重定位和长时间运行

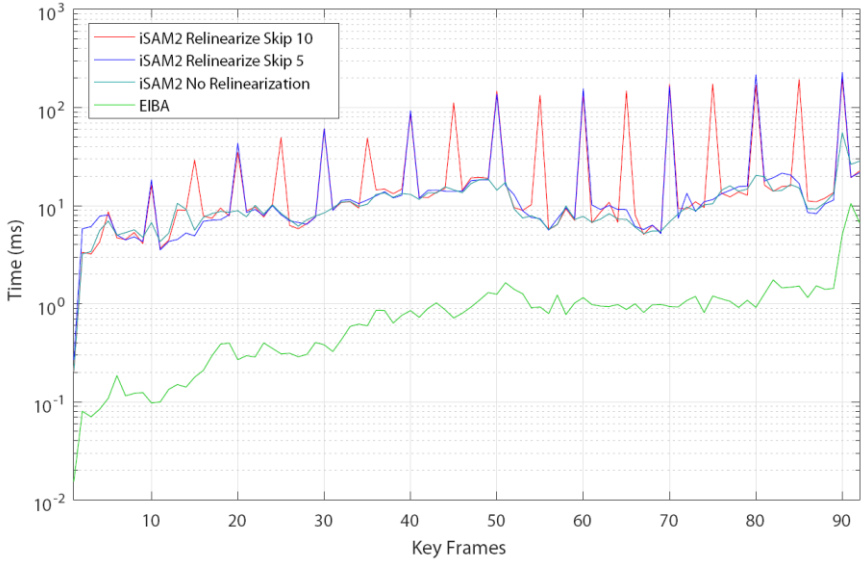
Efficient Incremental BA

- 提出了一个非常高效的Incremental Schur complement计算方法；
- 采用Preconditioned Conjugated Gradient进行求解，比Factorization的方法要快；
- 速度比iSAM2快一个数量级

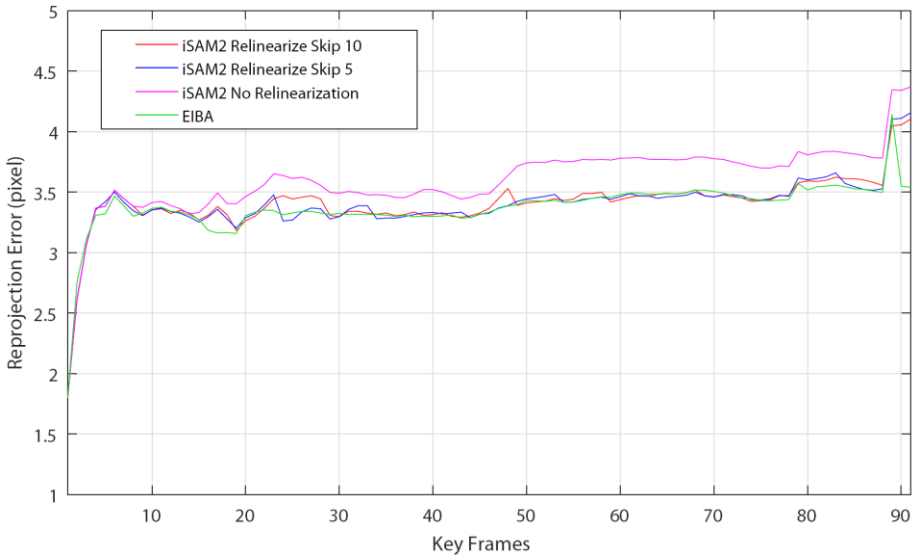
Sequence	Num. of Camera / Points	Num. of Observations	EIBA	iSAM2		
				No relinearization	relinearizeSkip = 10	relinearizeSkip = 5
fr3_long_office	92 / 4322	12027	88.9ms	983.9ms	1968.2ms	2670.9ms
fr2_desk	63 / 2780	6897	34.8ms	507.8ms	850.4ms	1152.0ms

Efficient Incremental BA

- 与iSAM2的对比



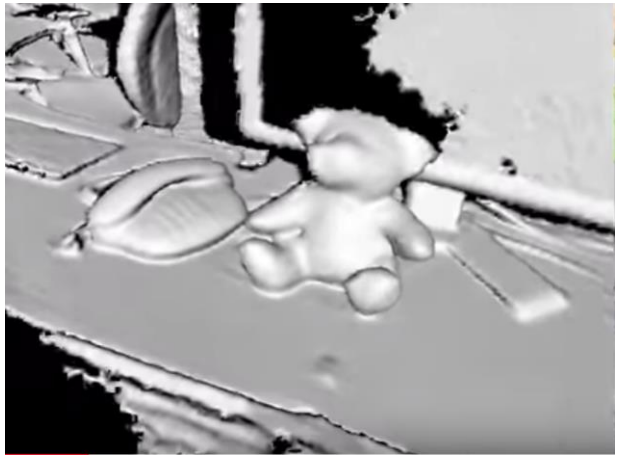
运行时间



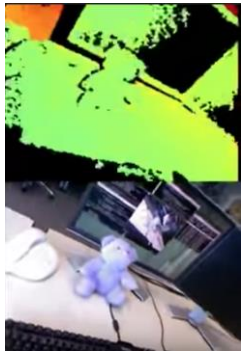
Reprojection Error

Integration & De-integration

- 使用de-integration可以将模型复原到integration前

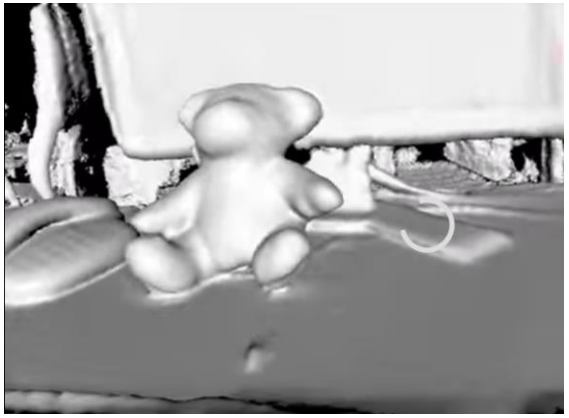


+

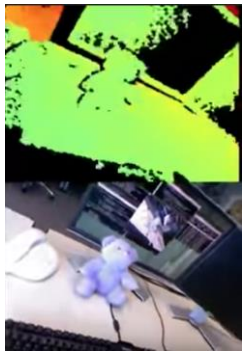


$$\mathbf{D}'(\mathbf{v}) = \frac{\mathbf{D}(\mathbf{v})\mathbf{W}(\mathbf{v}) + w_i(\mathbf{v})d_i(\mathbf{v})}{\mathbf{W}(\mathbf{v}) + w_i(\mathbf{v})}$$

$$\mathbf{W}'(\mathbf{v}) = \mathbf{W}(\mathbf{v}) + w_i(\mathbf{v}).$$



-



$$\mathbf{D}'(\mathbf{v}) = \frac{\mathbf{D}(\mathbf{v})\mathbf{W}(\mathbf{v}) - w_i(\mathbf{v})d_i(\mathbf{v})}{\mathbf{W}(\mathbf{v}) - w_i(\mathbf{v})}$$

$$\mathbf{W}'(\mathbf{v}) = \mathbf{W}(\mathbf{v}) - w_i(\mathbf{v}).$$

Keyframe-based Fusion

- 对于新来的一帧F_i
  - 如果是关键帧,则integrate到TSDF

$$\mathbf{D}'(\mathbf{v}) = \frac{\mathbf{D}(\mathbf{v})\mathbf{W}(\mathbf{v}) + w_i(\mathbf{x})\min(\mu, \phi(\mathbf{x}))}{\mathbf{W}(\mathbf{v}) + w_i(\mathbf{x})}, \mathbf{W}'(\mathbf{v}) = \mathbf{W}(\mathbf{v}) + w_i(\mathbf{x}),$$

  - 如非关键帧，则选出重合度最大的关键帧F_{k_i}进行de-integrate

$$\mathbf{D}'(\mathbf{v}) = \frac{\mathbf{D}(\mathbf{v})\mathbf{W}(\mathbf{v}) - w_i(\mathbf{x})\min(\mu, \phi(\mathbf{x}))}{\mathbf{W}(\mathbf{v}) - w_i(\mathbf{x})}, \mathbf{W}'(\mathbf{v}) = \mathbf{W}(\mathbf{v}) - w_i(\mathbf{x}).$$

  - 然后将该帧深度fuse到F_{k_i}上
  - 然后将fuse后的关键帧re-integrate到TSDF

$$D'(\mathbf{y}) = \frac{w_{K_i}(\mathbf{y})D_{K_i}(\mathbf{y}) + w_i(\mathbf{x})z_{\mathbf{x}}^{i\to K_i}}{w_{K_i}(\mathbf{y}) + w_i(\mathbf{x})}, w_{K_i} = w_{K_i}(\mathbf{y}) + w_i(\mathbf{x}).$$

Keyframe-based Fusion

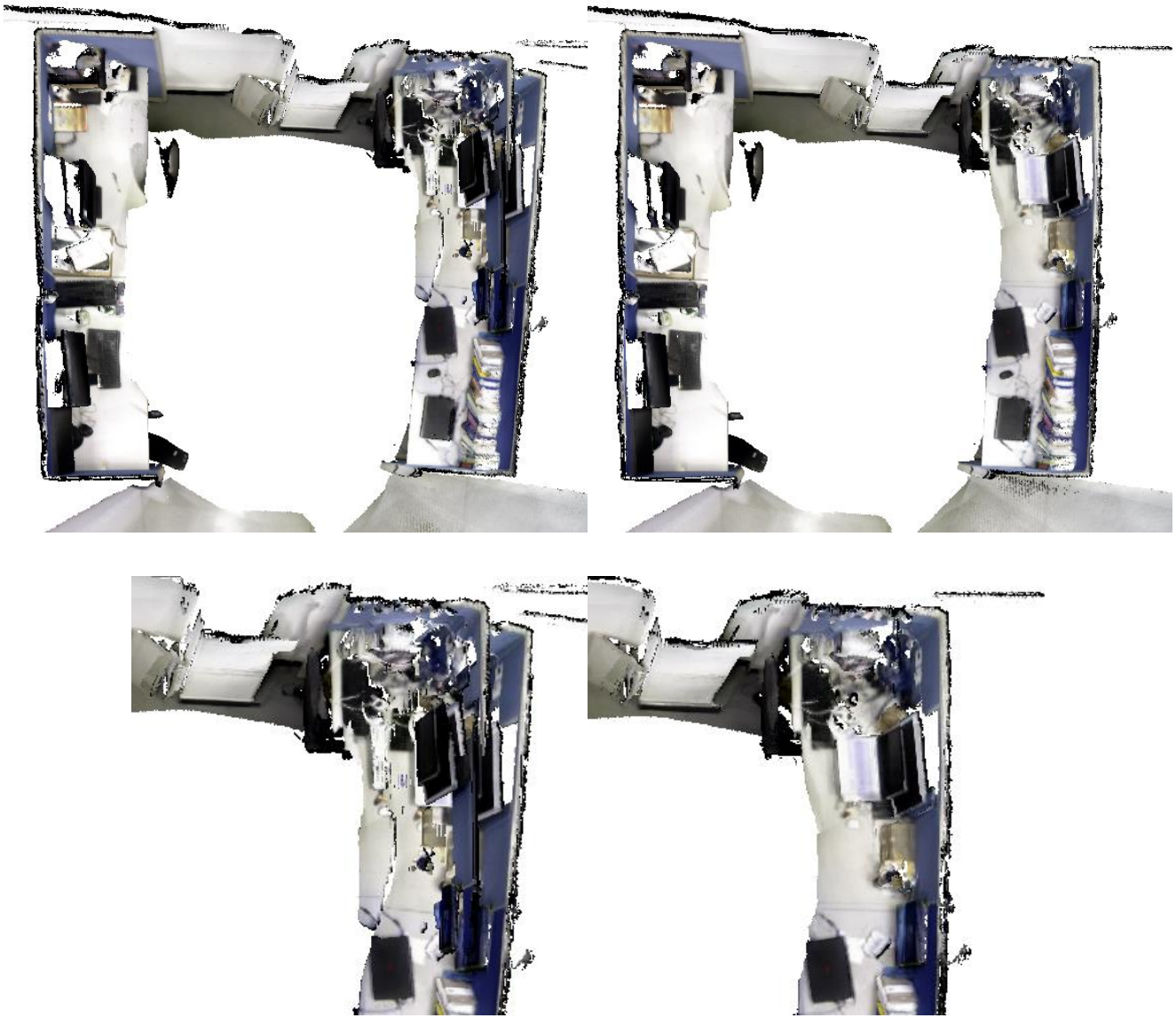
- 当关键帧的姿态发生改变（EIBA优化后）
  - 根据EIBA的优化结果，对姿态改变的关键帧进行re-integration.
  - 维护一个关键帧更新队列
    - 优先更新姿态改变最大的关键帧；
    - 每个时刻只对固定数量的关键帧进行re-integration，没有更新的关键帧会在放在后面的时刻更新。

Comparison of ATE RMSE on all of the sequences on TUM RGB-D Benchmark

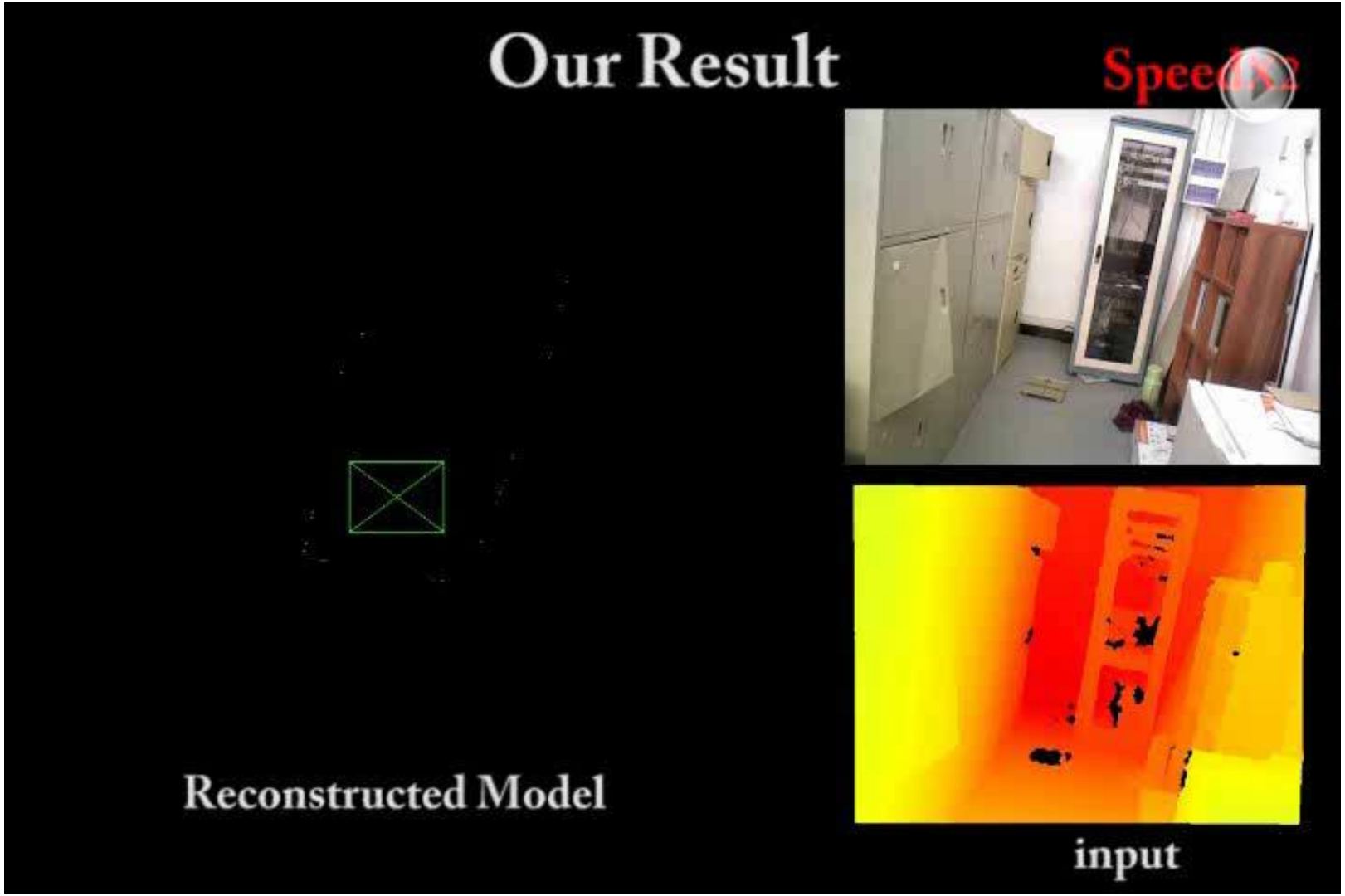
	Ours (all frames)	Ours (key frames)	Kintinuous	ElasticFusion	DVO-SLAM	RGB-D SLAM	MRSMap	BundleFusion
fr1_360	13.0cm	10.9cm		10.8cm	**8.3cm**			
fr1_desk	2.5cm	2.1cm	3.7cm	2.0cm	2.1cm	2.3cm	4.3cm	**1.6cm**
fr1_desk2	2.8cm	**2.4cm**	7.1cm	4.8cm	4.6cm	4.3cm	4.9cm	
fr1_floor	325.3cm	26.2cm		-				
fr1_plant	5.0cm	3.8cm	4.7cm	**2.2cm**	2.8cm	9.1cm	2.6cm	
fr1_room	14.8cm	13.4cm	7.5cm	6.8cm	**5.3cm**	8.4cm	6.9cm	
fr1_rpy	2.2cm	3.7cm	2.8cm	2.5cm	**2.0cm**	2.6cm	2.7cm	
fr1_teddy	18.7cm	15.7cm		8.3cm	**3.4cm**			
fr1_xyz	1.0cm	**0.7cm**	1.7cm	1.1cm	1.1cm	1.4cm	1.3cm	
fr2_360_hemisphere	37.6cm	**31.1cm**		-				
fr2_360_kidnap	132.6cm	**6.1cm**		-				
fr2_coke	**17.2cm**	20.2cm		-				
fr2_desk	7.2cm	7.1cm	3.4cm	7.1cm	**1.7cm**	5.7cm	5.2cm	
fr2_dishes	8.4cm	**7.9cm**		-				
fr2_large_no_loop	-	-		-			8.6cm	
fr2_large_with_loop	198.3cm	**196.7cm**		-				
fr2_metallic_sphere	**34.1cm**	44.3cm		-				
fr2_metallic_sphere2	11.1cm	**8.4cm**		-				
fr2_pioneer360	40.5cm	**35.8cm**		-				
fr2_pioneer_slam	91.2cm	**85.5cm**		-				
fr2_pioneer_slam2	169.7cm	**3.3cm**		-				
fr2_pioneer_slam3	28.1cm	**19.1cm**		-				
fr2_rpy	0.8cm	**0.6cm**		1.5cm				
fr2_xyz	1.2cm	1.2cm	2.9cm	**1.1cm**	1.8cm	0.8cm	2.0cm	**1.1cm**
fr2_flowerbouquet	7.0cm	**5.0cm**						
fr2_flowerbouquet_brownbackground	53.3cm	**51.7cm**						
fr2_desk_with_person	4.7cm	**4.5cm**						
fr3_cabinet	39.9cm	**7.9cm**		-				
fr3_large_cabinet	20.9cm	14.8cm		**9.9cm**				
fr3_long_office_household	3.2cm	2.8cm	3.0cm	**1.7cm**	3.5cm	3.2cm	4.2cm	2.2cm
fr3_nostructure_notexture_far	-	-		-				
fr3_nostructure_notexture_near_with_loop	-	-		-				
fr3_nostructure_texture_far	10.8cm	**5.3cm**		7.4cm				
fr3_nostructure_texture_near_withloop	2.9cm	2.7cm	3.1cm	1.6cm	1.8cm	1.7cm	201.8cm	**1.2cm**
fr3_structure_notexture_far	-	-		**3.0cm**				
fr3_structure_notexture_near	-	-		**2.1cm**				
fr3_structure_texture_far	1.8cm	1.6cm		**1.3cm**				
fr3_structure_texture_near	1.6cm	1.8cm		**1.5cm**				
fr3_nostructure_notexture_near	-	-						
fr3_teddy	-	-		**4.9cm**				
fr3_sitting_xyz	2.1cm	**1.7cm**						
fr3_walking_xyz	2.8cm	**2.4cm**						
fr3_sitting_halfsphere	**1.7cm**	1.9cm						
fr3_sitting_static	1.3cm	**0.9cm**						
fr3_walking_static	5.2cm	**3.9cm**						
fr3_walking_rpy	42.0cm	**33.7cm**						
fr3_sitting_rpy	**2.7cm**	4.1cm						
fr3_walking_halfsphere	25.6cm	**18.2cm**						

Reconstruction Result

Comparations without/with Re-integration

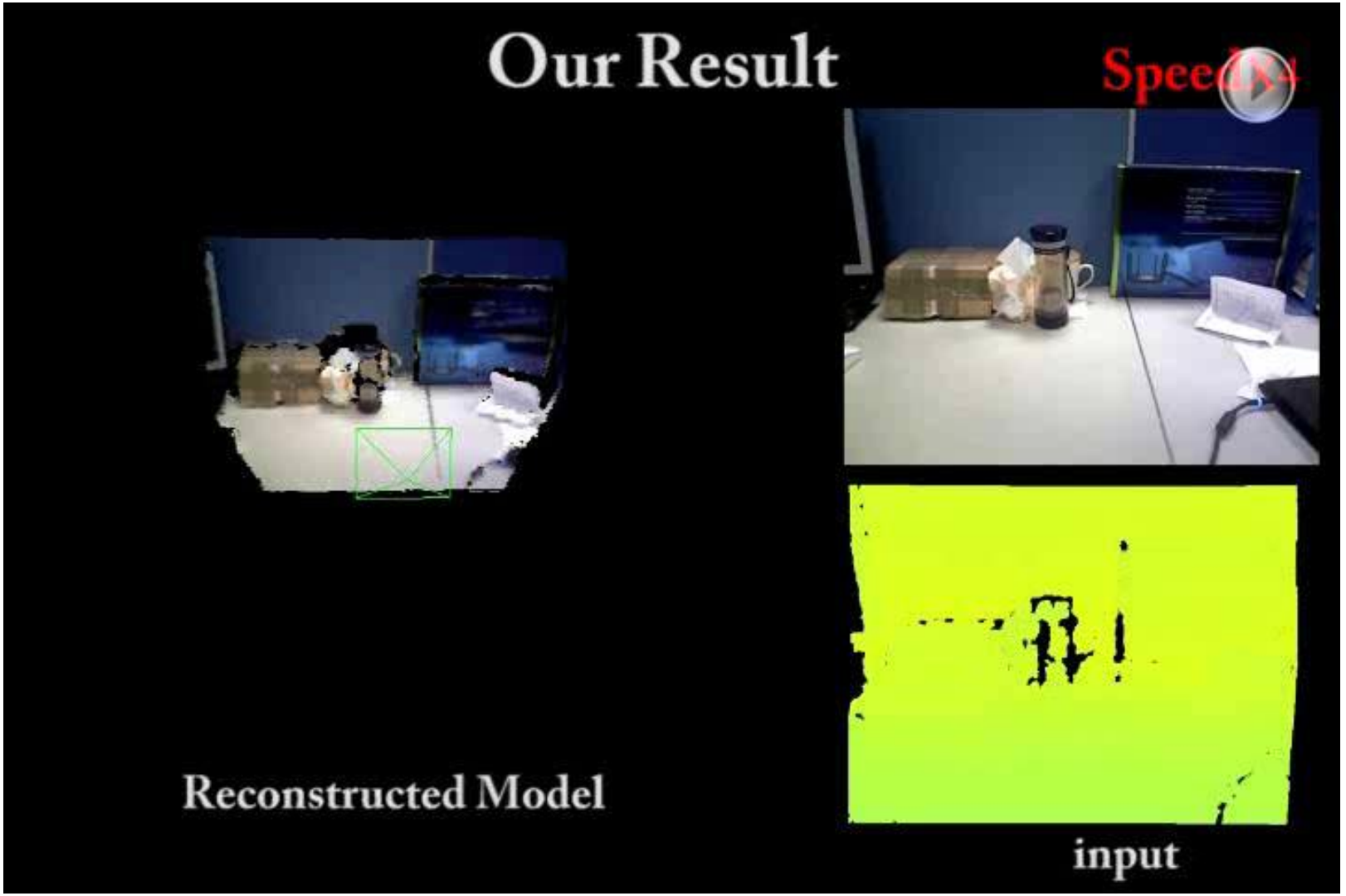


鲁棒处理快速运动



https://arxiv.org/abs/1711.05166

在线的回路闭合和三维表面调整



https://arxiv.org/abs/1711.05166

推荐开源系统

- Kintinuous
  - https://github.com/mp3guy/Kintinuous
- InfiniTAM
  - https://github.com/victorprad/InfiniTAM
- ElasticFusion
  - https://github.com/mp3guy/ElasticFusion
- BundleFusion
  - https://github.com/niessner/BundleFusion

产业化落地

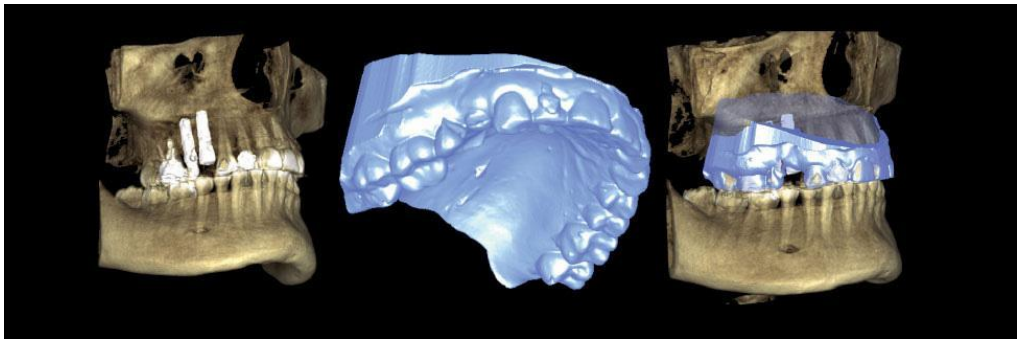
RGBD-SLAM的产业应用

- 落地场景
  - 商用级三维扫描重建
  - 移动端三维扫描重建
  - 移动端增强现实

- 难点和挑战
  - 深度质量
  - 平台算力
  - 复杂场景
    - 高反光
    - 透明
    - 运动物体

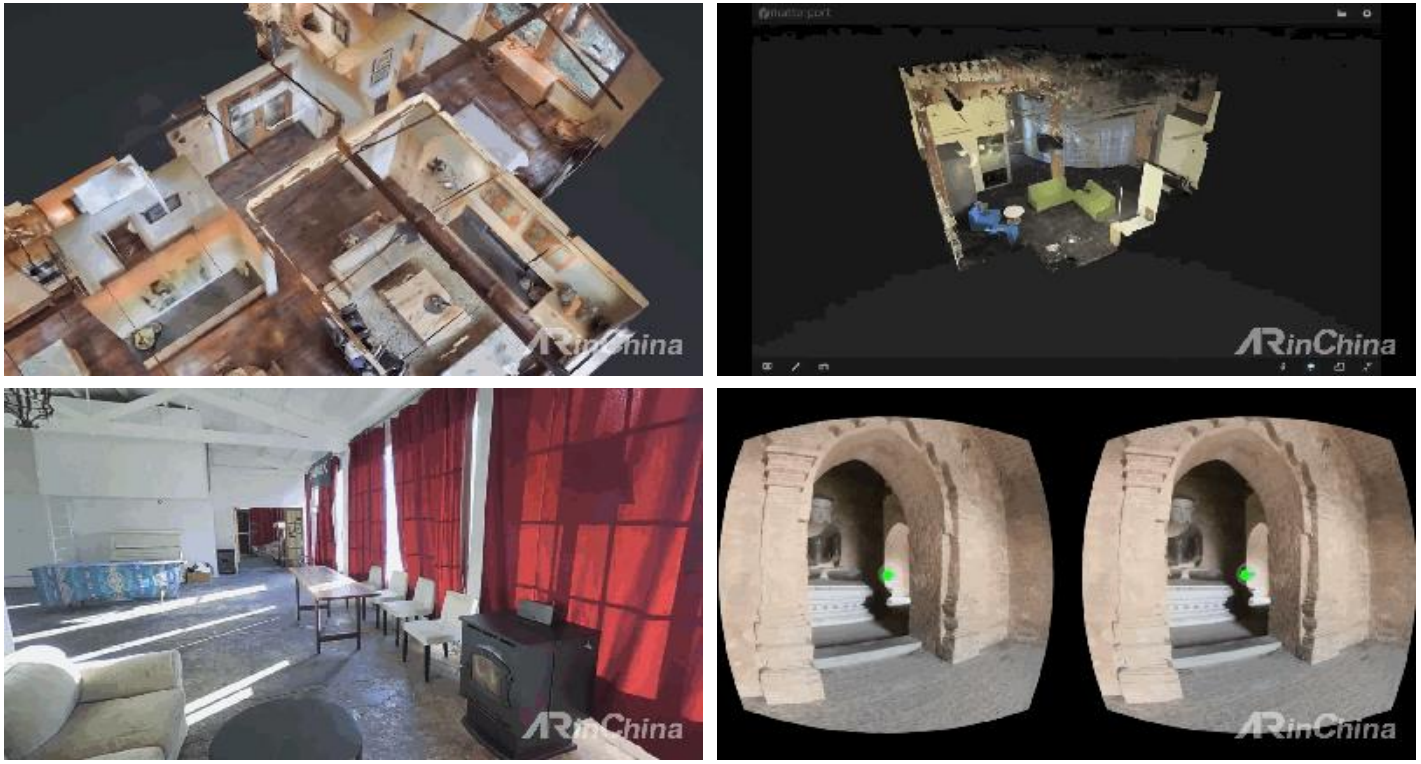
商用级三维扫描重建

- 针对物体的扫描
- 应用领域
  - 考古研究数字化保护
  - 高仿真玩具
  - 医疗诊断



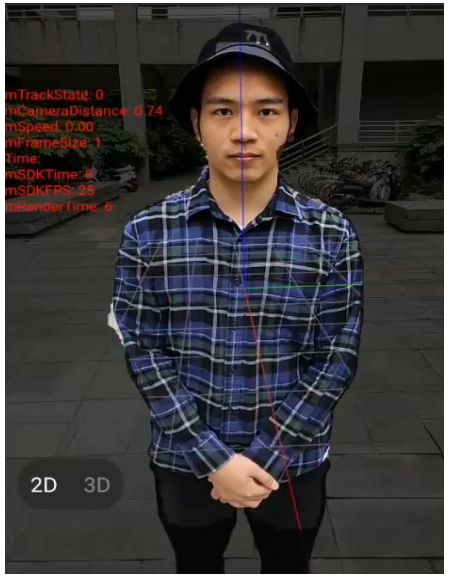
商用级三维扫描重建

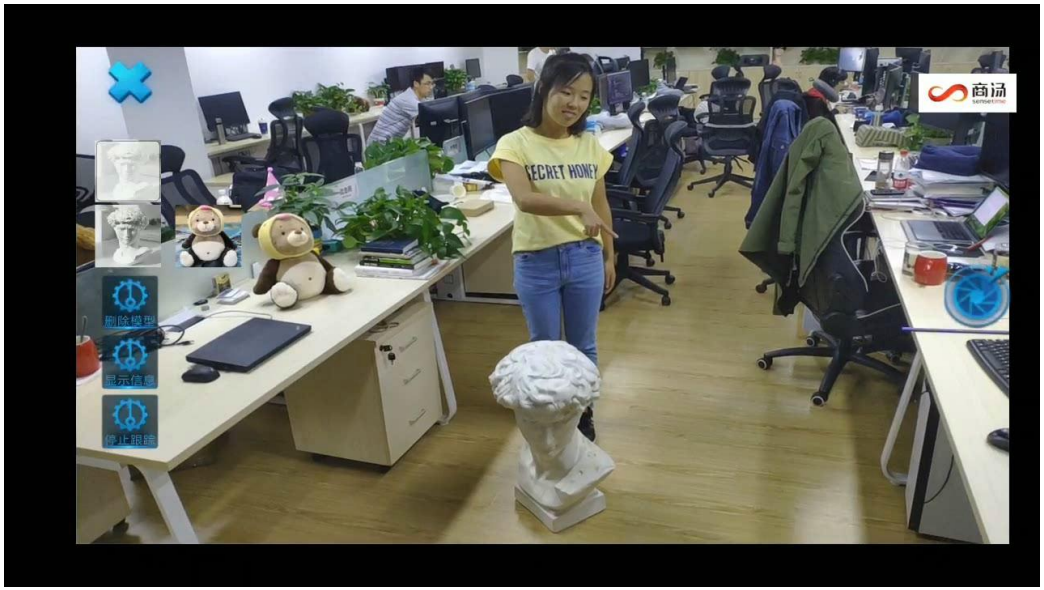
- 针对环境的扫描
- 应用领域
  - 三维场景展示漫游



移动端三维物体扫描重建

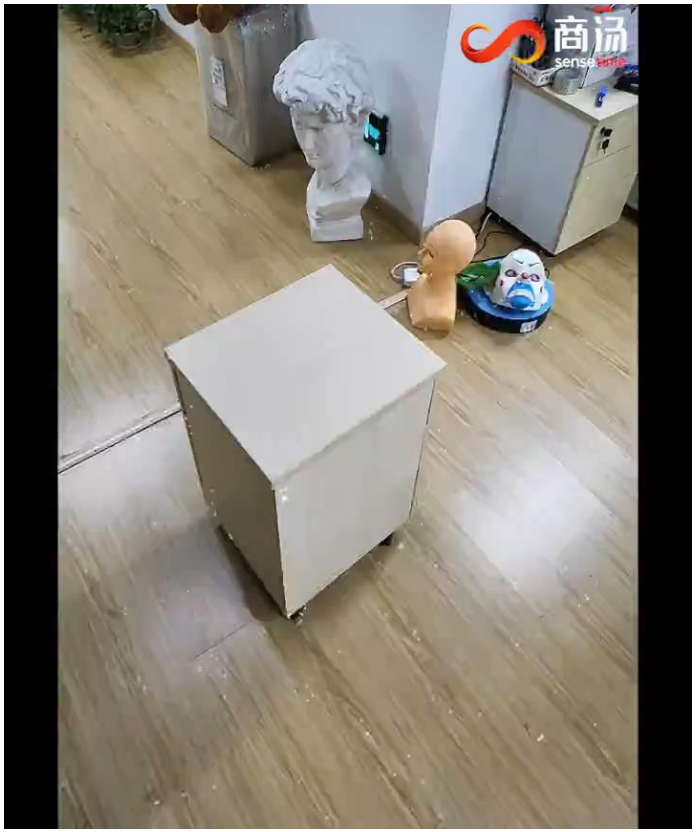
- 手机端在线扫描技术
  - AR内容生成创作
  - 手机平台实时扫描
  - 快速三维重建
    - 几何重建
    - 纹理贴图

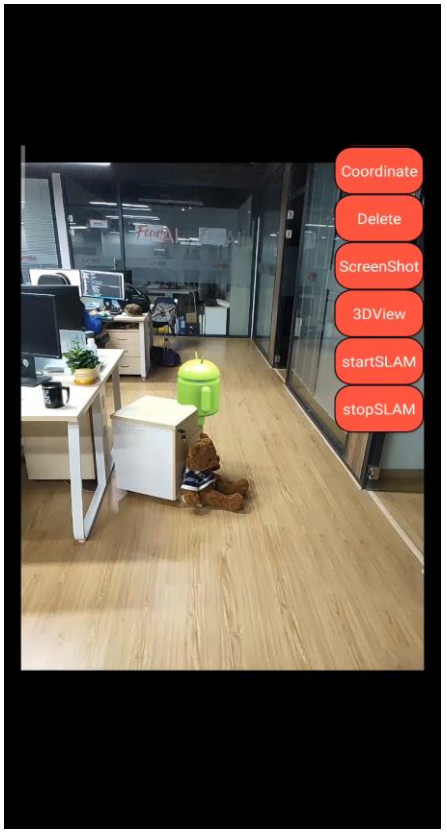




移动平台增强现实

- SenseAR RGBD-SLAM
- SenseAR实时稠密三维环境重建
  - 虚实交互
  - 物理碰撞
  - 遮挡处理
  - 复杂阴影投射





Thank you!

Q & A