Lecture 6: Multi-view Stereo & Structure from Motion

Prof. Rob Fergus

Many slides adapted from Lana Lazebnik and Noah Snavelly, who in turn adapted slides from Steve Seitz, Rick Szeliski, Martial Hebert, Mark Pollefeys, and others....
Overview

• Multi-view stereo

• Structure from Motion (SfM)

• Large scale Structure from Motion

• Kinect Fusion

• Dynamic Fusion
Multi-view Stereo

Point Grey’s Bumblebee XB3

Point Grey’s ProFusion 25

CMU’s 3D Room
Multi-view Stereo
Multi-view Stereo

Input: calibrated images from several viewpoints
Output: 3D object model

Figures by Carlos Hernandez

[Slide: N. Snavely]
Stereo: another view

error

depth

[Slide: N. Snavely]
Choosing the stereo baseline

What’s the optimal baseline?

– Too small: large depth error
– Too large: difficult search problem

[Slide: N. Snavely]
The Effect of Baseline on Depth Estimation

Figure 2: An example scene. The grid pattern in the background has ambiguity of matching.
Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$.

The horizontal axis is normalized such that $8bF = 1$.

Fig. 6. Combining two stereo pairs with different baselines.

Fig. 7. Combining multiple baseline stereo pairs.
Multibaseline Stereo

Basic Approach

• Choose a reference view
• Use your favorite stereo algorithm BUT
  > replace two-view SSD with SSSD over all baselines

Limitations

• Only gives a depth map (not an “object model”)
• Won’t work for widely distributed views:
Problem: \textit{visibility}

Some Solutions
- Match only nearby photos [Narayanan 98]
- Use NCC instead of SSD, Ignore NCC values > threshold [Hernandez & Schmitt 03]
Popular matching scores

- **SSD** (Sum Squared Distance)
  \[
  \sum_{x,y} |W_1(x, y) - W_2(x, y)|^2
  \]

- **NCC** (Normalized Cross Correlation)
  \[
  \frac{\sum_{x,y} (W_1(x, y) - \bar{W_1})(W_2(x, y) - \bar{W_2})}{\sigma_{W_1} \sigma_{W_2}}
  \]
  where \( \bar{W_i} = \frac{1}{n} \sum_{x,y} W_i \) and \( \sigma_{W_i} = \sqrt{\frac{1}{n} \sum_{x,y} (W_i - \bar{W_i})^2} \)

- what advantages might NCC have?
Reconstruction from Silhouettes

Binary Images →

Approach:
- *Backproject* each silhouette
- **Intersect** backprojected volumes
Volume intersection

Reconstruction Contains the True Scene

- But is generally not the same
- In the limit (all views) get *visual hull*
  - Complement of all lines that don’t intersect S
Voxel algorithm for volume intersection

Color voxel black if on silhouette in every image

- $O(MN^3)$, for $M$ images, $N^3$ voxels
- Don’t have to search $2^{N^3}$ possible scenes!
Properties of Volume Intersection

Pros

• Easy to implement, fast
• Accelerated via octrees [Szeliski 1993] or interval techniques [Matusik 2000]

Cons

• No concavities
• Reconstruction is not photo-consistent
• Requires identification of silhouettes
Multi-view stereo: Summary

- Multiple-baseline stereo
  - Pick one input view as reference
  - Inverse depth instead of disparity
- Volumetric stereo
  - Photo-consistency
  - Space carving
- Shape from silhouettes
  - Visual hull: intersection of visual cones
- Carved visual hulls
- Feature-based stereo
  - From sparse to dense correspondences

All assume calibrated cameras!
Overview

Multi-view stereo

Structure from Motion (SfM)

Large scale Structure from Motion
Structure from motion
Multiple-view geometry questions

• **Scene geometry (structure):** Given 2D point matches in two or more images, where are the corresponding points in 3D?

• **Correspondence (stereo matching):** Given a point in just one image, how does it constrain the position of the corresponding point in another image?

• **Camera geometry (motion):** Given a set of corresponding points in two or more images, what are the camera matrices for these views?
Structure from motion

- Given: $m$ images of $n$ fixed 3D points
  \[ x_{ij} = P_i X_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \]

- Problem: estimate $m$ projection matrices $P_i$ and $n$ 3D points $X_j$ from the $mn$ correspondences $x_{ij}$
Structure from motion ambiguity

• If we scale the entire scene by some factor $k$ and, at the same time, scale the camera matrices by the factor of $1/k$, the projections of the scene points in the image remain exactly the same:

$$x = PX = \left( \frac{1}{k} P \right) (kX)$$

It is impossible to recover the absolute scale of the scene!
Structure from motion ambiguity

- If we scale the entire scene by some factor $k$ and, at the same time, scale the camera matrices by the factor of $1/k$, the projections of the scene points in the image remain exactly the same.

- More generally: if we transform the scene using a transformation $Q$ and apply the inverse transformation to the camera matrices, then the images do not change.

$$x = PX = (PQ^{-1})(QX)$$
## Types of ambiguity

<table>
<thead>
<tr>
<th>Type</th>
<th>Transformation Matrix</th>
<th>Preserves...</th>
</tr>
</thead>
</table>
| Projective  | \[
\begin{bmatrix} A & t \\
 v^T & 1 \\
\end{bmatrix}
\] | Intersection and tangency                         |
| Affine      | \[
\begin{bmatrix} A & t \\
 0^T & 1 \\
\end{bmatrix}
\] | Parallelism, volume ratios                        |
| Similarity  | \[
\begin{bmatrix} s & R & t \\
 0^T & 1 \\
\end{bmatrix}
\] | Angles, ratios of length                          |
| Euclidean   | \[
\begin{bmatrix} R & t \\
 0^T & 1 \\
\end{bmatrix}
\] | Angles, lengths                                   |

- With no constraints on the camera calibration matrix or on the scene, we get a **projective** reconstruction.
- Need additional information to **upgrade** the reconstruction to affine, similarity, or Euclidean.
Projective ambiguity

\[ x = PX = \left( PQ_P^{-1} \right) \left( Q_P X \right) \]
Projective ambiguity
Affine ambiguity

\[ x = PX = \left( PQ_A^{-1} \right) (Q_A X) \]
Affine ambiguity
Similarity ambiguity

\[ x = PX = \left( PQ_s^{-1} \right) (Q_s X) \]
Similarity ambiguity
Structure from motion

- Let’s start with *affine cameras* (the math is easier)
Recall: Orthographic Projection

Special case of perspective projection

- Distance from center of projection to image plane is infinite

![Diagram of orthographic projection]

- Projection matrix:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
= 
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\Rightarrow (x, y)
\]
Affine cameras

Orthographic Projection

Parallel Projection
Affine cameras

- A general affine camera combines the effects of an affine transformation of the 3D space, orthographic projection, and an affine transformation of the image:

\[
P = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
[4 \times 4 \text{ affine}] = \begin{bmatrix}
a_{11} & a_{12} & a_{13} & b_1 \\
a_{21} & a_{22} & a_{23} & b_2 \\
0 & 0 & 0 & 1
\end{bmatrix} = \begin{bmatrix}
A & b \\
0 & 1
\end{bmatrix}
\]

- Affine projection is a linear mapping + translation in inhomogeneous coordinates

\[
x = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23}
\end{bmatrix} \begin{pmatrix} X \\ Y \\
Z \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = AX + b
\]

Projection of world origin
Affine structure from motion

- Given: $m$ images of $n$ fixed 3D points:
  \[ x_{ij} = A_i X_j + b_i, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \]

- Problem: use the $mn$ correspondences $x_{ij}$ to estimate $m$ projection matrices $A_i$ and translation vectors $b_i$, and $n$ points $X_j$

- The reconstruction is defined up to an arbitrary affine transformation $\mathbf{Q}$ (12 degrees of freedom):
  \[
  \begin{bmatrix}
  A & b \\
  0 & 1
  \end{bmatrix} \rightarrow \begin{bmatrix}
  A & b \\
  0 & 1
  \end{bmatrix} Q^{-1}, \quad \begin{bmatrix}
  X \\
  1
  \end{bmatrix} \rightarrow Q \begin{bmatrix}
  X \\
  1
  \end{bmatrix}
  \]

- We have $2mn$ knowns and $8m + 3n$ unknowns (minus 12 dof for affine ambiguity)

- Thus, we must have $2mn \geq 8m + 3n - 12$

- For two views, we need four point correspondences
Affine structure from motion

- Centering: subtract the centroid of the image points

\[
\hat{x}_{ij} = x_{ij} - \frac{1}{n} \sum_{k=1}^{n} x_{ik} = A_i X_j + b_i - \frac{1}{n} \sum_{k=1}^{n} (A_i X_k + b_i)
\]

\[
= A_i \left( X_j - \frac{1}{n} \sum_{k=1}^{n} X_k \right) = A_i \hat{X}_j
\]

- For simplicity, assume that the origin of the world coordinate system is at the centroid of the 3D points

- After centering, each normalized point \(x_{ij}\) is related to the 3D point \(X_i\) by

\[
\hat{x}_{ij} = A_i X_j
\]
Affine structure from motion

- Let’s create a $2m \times n$ data (measurement) matrix:

$$
\mathbf{D} = 
\begin{bmatrix}
\hat{x}_{11} & \hat{x}_{12} & \cdots & \hat{x}_{1n} \\
\hat{x}_{21} & \hat{x}_{22} & \cdots & \hat{x}_{2n} \\
\vdots & & & \\
\hat{x}_{m1} & \hat{x}_{m2} & \cdots & \hat{x}_{mn}
\end{bmatrix}
$$

Affine structure from motion

- Let’s create a $2m \times n$ data (measurement) matrix:

\[
D = \begin{bmatrix}
\hat{x}_{11} & \hat{x}_{12} & \cdots & \hat{x}_{1n} \\
\hat{x}_{21} & \hat{x}_{22} & \cdots & \hat{x}_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{x}_{m1} & \hat{x}_{m2} & \cdots & \hat{x}_{mn}
\end{bmatrix}
= \begin{bmatrix}
A_1 \\
A_2 \\
\vdots \\
A_m
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
\vdots \\
X_n
\end{bmatrix}
\]

- The measurement matrix $D = MS$ must have rank 3!

Factorizing the measurement matrix

\[ \text{Measurements} = \text{Motion} \times \text{Shape} \]

\[ D = MS \]

Source: M. Hebert
Factorizing the measurement matrix

- Singular value decomposition of $D$:

$$D = U W V^T$$

Source: M. Hebert
Factorizing the measurement matrix

- Singular value decomposition of $D$:

$$D = U W V^T$$

To reduce to rank 3, we just need to set all the singular values to 0 except for the first 3.
Factorizing the measurement matrix

- Obtaining a factorization from SVD:

\[
\begin{align*}
2m & \quad D \quad = \quad U_3 \times 3 \quad W_3 \quad \times \quad V_3^T \\
\end{align*}
\]
Factorizing the measurement matrix

• Obtaining a factorization from SVD:

\[ D = U_3 W_3^{1/2} S = W_3^{1/2} V_3^T \]

This decomposition minimizes \(|D-MS|^2|\)
Affine ambiguity

The decomposition is not unique. We get the same $D$ by using any $3 \times 3$ matrix $C$ and applying the transformations $M \rightarrow MC$, $S \rightarrow C^{-1}S$

That is because we have only an affine transformation and we have not enforced any Euclidean constraints (like forcing the image axes to be perpendicular, for example)

Source: M. Hebert
Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and of unit length

\[ \mathbf{a}_1 \cdot \mathbf{a}_2 = 0 \]

\[ |\mathbf{a}_1|^2 = |\mathbf{a}_2|^2 = 1 \]

Source: M. Hebert
Solve for orthographic constraints

Three equations for each image $i$

\[
\begin{align*}
\tilde{a}_{i1}^T CC^T \tilde{a}_{i1} &= 1 \\
\tilde{a}_{i2}^T CC^T \tilde{a}_{i2} &= 1 \\
\tilde{a}_{i1}^T CC^T \tilde{a}_{i2} &= 0
\end{align*}
\]

- Solve for $L = CC^T$
- Recover $C$ from $L$ by Cholesky decomposition: $L = CC^T$
- Update $A$ and $X$: $A = \tilde{A}C, \quad X = C^{-1}\tilde{X}$
Algorithm summary

- Given: \( m \) images and \( n \) features \( x_{ij} \)
- For each image \( i \), center the feature coordinates
- Construct a \( 2m \times n \) measurement matrix \( D \):
  - Column \( j \) contains the projection of point \( j \) in all views
  - Row \( i \) contains one coordinate of the projections of all the \( n \) points in image \( i \)
- Factorize \( D \):
  - Compute SVD: \( D = U W V^T \)
  - Create \( U_3 \) by taking the first 3 columns of \( U \)
  - Create \( V_3 \) by taking the first 3 columns of \( V \)
  - Create \( W_3 \) by taking the upper left \( 3 \times 3 \) block of \( W \)
- Create the motion and shape matrices:
  - \( M = U_3 W_3^{1/2} \) and \( S = W_3^{1/2} V_3^T \) (or \( M = U_3 \) and \( S = W_3 V_3^T \))
- Eliminate affine ambiguity

Source: M. Hebert
Dealing with missing data

• So far, we have assumed that all points are visible in all views.
• In reality, the measurement matrix typically looks something like this:
Dealing with missing data

- Possible solution: decompose matrix into dense sub-blocks, factorize each sub-block, and fuse the results
  - Finding dense maximal sub-blocks of the matrix is NP-complete (equivalent to finding maximal cliques in a graph)

- Incremental bilinear refinement

1. Perform factorization on a dense sub-block
2. Solve for a new 3D point visible by at least two known cameras (linear least squares)
3. Solve for a new camera that sees at least three known 3D points (linear least squares)

Projective structure from motion

- Given: \( m \) images of \( n \) fixed 3D points

\[
z_{ij} \mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n
\]

- Problem: estimate \( m \) projection matrices \( \mathbf{P}_i \) and \( n \) 3D points \( \mathbf{X}_j \) from the \( mn \) correspondences \( \mathbf{x}_{ij} \)
Projective structure from motion

• Given: \( m \) images of \( n \) fixed 3D points

\[
z_{ij} \mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n
\]

• Problem: estimate \( m \) projection matrices \( \mathbf{P}_i \) and \( n \) 3D points \( \mathbf{X}_j \) from the \( mn \) correspondences \( \mathbf{x}_{ij} \)

• With no calibration info, cameras and points can only be recovered up to a 4x4 projective transformation \( \mathbf{Q} \):

\[
\mathbf{X} \rightarrow \mathbf{QX}, \quad \mathbf{P} \rightarrow \mathbf{PQ}^{-1}
\]

• We can solve for structure and motion when

\[
2mn \geq 11m + 3n - 15
\]

• For two cameras, at least 7 points are needed
Projective SFM: Two-camera case

- Compute fundamental matrix $F$ between the two views
- First camera matrix: $[I|0]$
- Second camera matrix: $[A|b]$
- Then $b$ is the epipole ($F^Tb = 0$), $A = -[b \times]F$

F&P sec. 13.3.1
Sequential structure from motion

• Initialize motion from two images using fundamental matrix

• Initialize structure by triangulation

• For each additional view:
  • Determine projection matrix of new camera using all the known 3D points that are visible in its image – calibration
Sequential structure from motion

• Initialize motion from two images using fundamental matrix

• Initialize structure by triangulation

• For each additional view:
  • Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
  • Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*
Sequential structure from motion

• Initialize motion from two images using fundamental matrix

• Initialize structure by triangulation

• For each additional view:
  • Determine projection matrix of new camera using all the known 3D points that are visible in its image – calibration
  • Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – triangulation

• Refine structure and motion: bundle adjustment
Bundle adjustment

- Non-linear method for refining structure and motion
- Minimizing reprojection error

\[ E(P, X) = \sum_{i=1}^{m} \sum_{j=1}^{n} D\left(x_{ij}, P_i X_j\right)^2 \]
Self-calibration

• Self-calibration (auto-calibration) is the process of determining intrinsic camera parameters directly from uncalibrated images.

• For example, when the images are acquired by a single moving camera, we can use the constraint that the intrinsic parameter matrix remains fixed for all the images.
  
  • Compute initial projective reconstruction and find 3D projective transformation matrix $Q$ such that all camera matrices are in the form $P_i = K [R_i | t_i]$.

• Can use constraints on the form of the calibration matrix: zero skew.
Review: Structure from motion

- Ambiguity
- Affine structure from motion
  - Factorization
- Dealing with missing data
  - Incremental structure from motion
- Projective structure from motion
  - Bundle adjustment
  - Self-calibration
Summary: 3D geometric vision

- **Single-view geometry**
  - The pinhole camera model
    - Variation: orthographic projection
  - The perspective projection matrix
  - Intrinsic parameters
  - Extrinsic parameters
  - Calibration

- **Multiple-view geometry**
  - Triangulation
  - The epipolar constraint
    - Essential matrix and fundamental matrix
  - Stereo
    - Binocular, multi-view
  - Structure from motion
    - Reconstruction ambiguity
    - Affine SFM
    - Projective SFM
Overview

Multi-view stereo

Structure from Motion (SfM)

Large scale Structure from Motion
Large-scale Structure from motion

Given many images from photo collections how can we
a) figure out where they were all taken from?
b) build a 3D model of the scene?

This is (roughly) the **structure from motion** problem

Slides from N. Snavely
Challenges

appearance variation

resolution

massive collections

82,754 results for photos matching *notre* and *dame* and *paris*.
Large-scale structure from motion

Dubrovnik, Croatia. 4,619 images (out of an initial 57,845).
Total reconstruction time: 23 hours
Number of cores: 352
Structure from motion

- Input: images with points in correspondence $p_{i,j} = (u_{i,j}, v_{i,j})$

- Output
  - structure: 3D location $x_i$ for each point $p_i$
  - motion: camera parameters $R_j, t_j$ possibly $K_j$

- Objective function: minimize *reprojection error*
First step: how to get correspondence?

Feature detection and matching
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]
Feature matching

Match features between each pair of images
Feature matching

Refine matching using RANSAC to estimate fundamental matrix between each pair
Slide: N. Snavely
Structure from motion

minimize $g(R, T, P)$
SfM objective function

Given point $x$ and rotation and translation $R, t$

\[
\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = Rx + t
\]

\[ u' = \frac{fx'}{z'} \quad v' = \frac{fy'}{z'} \]

Minimize sum of squared reprojection errors:

\[
g(X, R, T) = \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| P(x_i, R_j, t_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2
\]

predicted image location

observed image location
Solving structure from motion

Minimizing $g$ is difficult

- $g$ is non-linear due to rotations, perspective division
- lots of parameters: 3 for each 3D point, 6 for each camera
- difficult to initialize
- gauge ambiguity: error is invariant to a similarity transform (translation, rotation, uniform scale)

Many techniques use non-linear least-squares (NLLS) optimization (*bundle adjustment*)

- Levenberg-Marquardt is one common algorithm for NLLS
Extensions to SfM

Can also solve for intrinsic parameters (focal length, radial distortion, etc.)

Can use a more robust function than squared error, to avoid fitting to outliers

Problem size

Trevi Fountain collection

466 input photos

+ > 100,000 3D points

= very large optimization problem
Incremental structure from motion
Incremental structure from motion
Incremental structure from motion
Photo Explorer
KinectFusion: Real-Time Dense Surface Mapping and Tracking

by Richard.A Newcombe et al.

Presenting: Boaz Petersil
Video
Motivation

Augmented Reality

3d model scanning

Robot Navigation

Etc..

[Slide: B. Petersil]
Related Work

Tracking (&sparse Mapping)
- PTAM
- Bundle-adjustment (offline)

Tracking&Mapping
- Kinect Fusion

Dense Mapping
- Levoy et al. Digital Michelangelo SIGGRAPH '00
- Jan-Michael Frahm et al. Building Rome on a cloudless day ECCV'10

[Slide: B. Petersil]
Challenges

• Tracking Camera Precisely
• Fusing and De-noising Measurements
• Avoiding Drift
• Real-Time
• Low-Cost Hardware
Proposed Solution

• Fast Optimization for Tracking, Due to High Frame Rate.
• Global Framework for fusing data
• Interleaving Tracking & Mapping
• Using Kinect to get Depth data (low cost)
• Using GPGPU to get Real-Time Performance (low cost)
How does Kinect work?
Method

a) 3D Vertex & Normal Map
b) ICP Camera Tracking
c) Volume Data Integration
d) Ray-casted 3D Rendering
e) ICP Outliers
f) Generated 3D Vertex & Normal Map

[Slide: B. Petersil]
Tracking

• Finding Camera position is the same as fitting frame’s Depth Map onto Model
Tracking – ICP algorithm

- icp = iterative closest point
- Goal: fit two 3d point sets
- Problem: What are the correspondences?
- Kinect fusion chosen solution:
  1) Start with $T_0$
  2) Project model onto camera
  3) Correspondences are points with same coordinates
  4) Find new T with Least - Squares
  5) Apply T, and repeat 2-5 until convergence
Tracking – ICP algorithm

- Assumption: frame and model are roughly aligned.
- True because of high frame rate
Mapping

- Mapping is Fusing depth maps when camera poses are known
- Problems:
  - measurements are noisy
  - Depth maps have holes in them
- Solution:
  - using implicit surface representation
  - Fusing = estimating from all frames relevant
Mapping – surface representation

- Surface is represented implicitly - using Truncated Signed Distance Function (TSDF)

- Numbers in cells measure voxel distance to surface – \( D \)
Mapping
Mapping

d = [pixel depth] – [distance from sensor to voxel]

[Slide: B. Petersil]
Mapping
Mapping
Mapping

• Each Voxel also has a weight $W$, proportional to grazing angle
• Voxel $D$ is the weighted average of all measurements

$$F_k(p) = \frac{W_{k-1}(p)F_{k-1}(p) + W_{Rk}(p)F_{Rk}(p)}{W_{k-1}(p) + W_{Rk}(p)}$$

$$W_k(p) = W_{k-1}(p) + W_{Rk}(p)$$

[Slide: B. Petersil]
Handling drift

- Drift would have happened if tracking was done from frame to previous frame.
- Tracking is done on built model.

[Slide: B. Petersil]
Thousands of particles simulated directly on 3D reconstruction (as room is being reconstructed)
Pros & Cons

• Pros:
  – Really nice results!
    • Real time performance (30 HZ)
    • Dense model
    • No drift with local optimization
    • Robust to scene changes
  – Elegant solution

• Cons:
  – 3d grid can’t be trivially up-scaled
Limitations

• Doesn’t work for large areas (Voxel-Grid)
• Doesn’t work far away from objects (active ranging)
• Doesn’t work out-doors (IR)
• Requires powerful Graphics card
• Uses lots of battery (active ranging)
• Only one sensor at a time