COMP 558

Lecture 23:
RGBD Cameras, Point Clouds, ICP algorithm

3D Scan demo

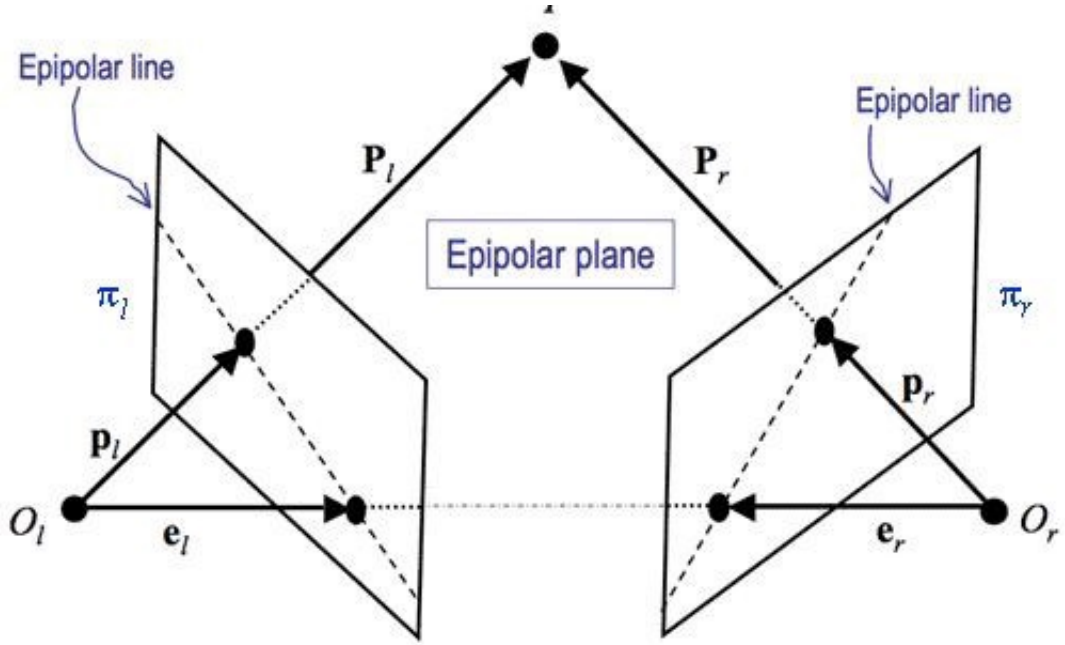
- Let’s start with a demo by scanning a 3D object by placing it on a small turntable. The same strategy would be used for scanning an entire scene.

Review of Stereo

- Recall that stereo involves two cameras at fixed locations with respect to one another
- In stereo calibration we seek to recover the extrinsics and intrinsics. If we have N>8 pixel-wise matches, we can estimate the entries of F, the fundamental matrix.
- Then, given F, and knowledge of camera intrinsics, since $F = K_2^{-T} E K_1^{-1,}$ we can estimate the essential matrix, E whose entries are $R_2 R_1^T [T_1]_x$
- The essential matrix captures the relationship between the two camera frames.

What if?

- We had a calibrated stereo rig where we knew in advance the positions of the cameras with respect to each other? In other words, we knew R_2, R_1^T , and $[T_1]_x$
- Then, a 3D scene point could be rendered in 3D with respect to one of the cameras.



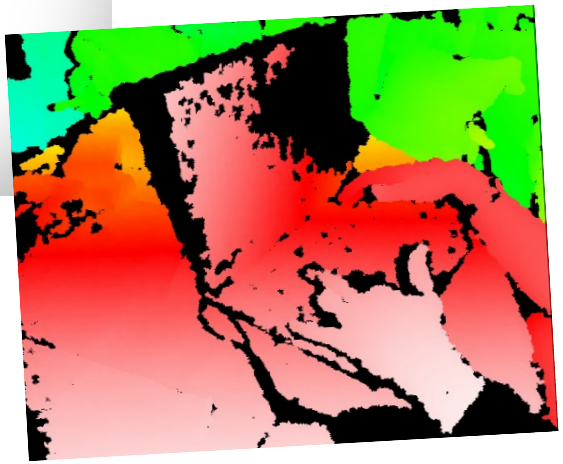
Stereo Reconstruction

- In the notes we used $\mathbf{X_1}$ and $\mathbf{X_2}$ place of P_l and P_r

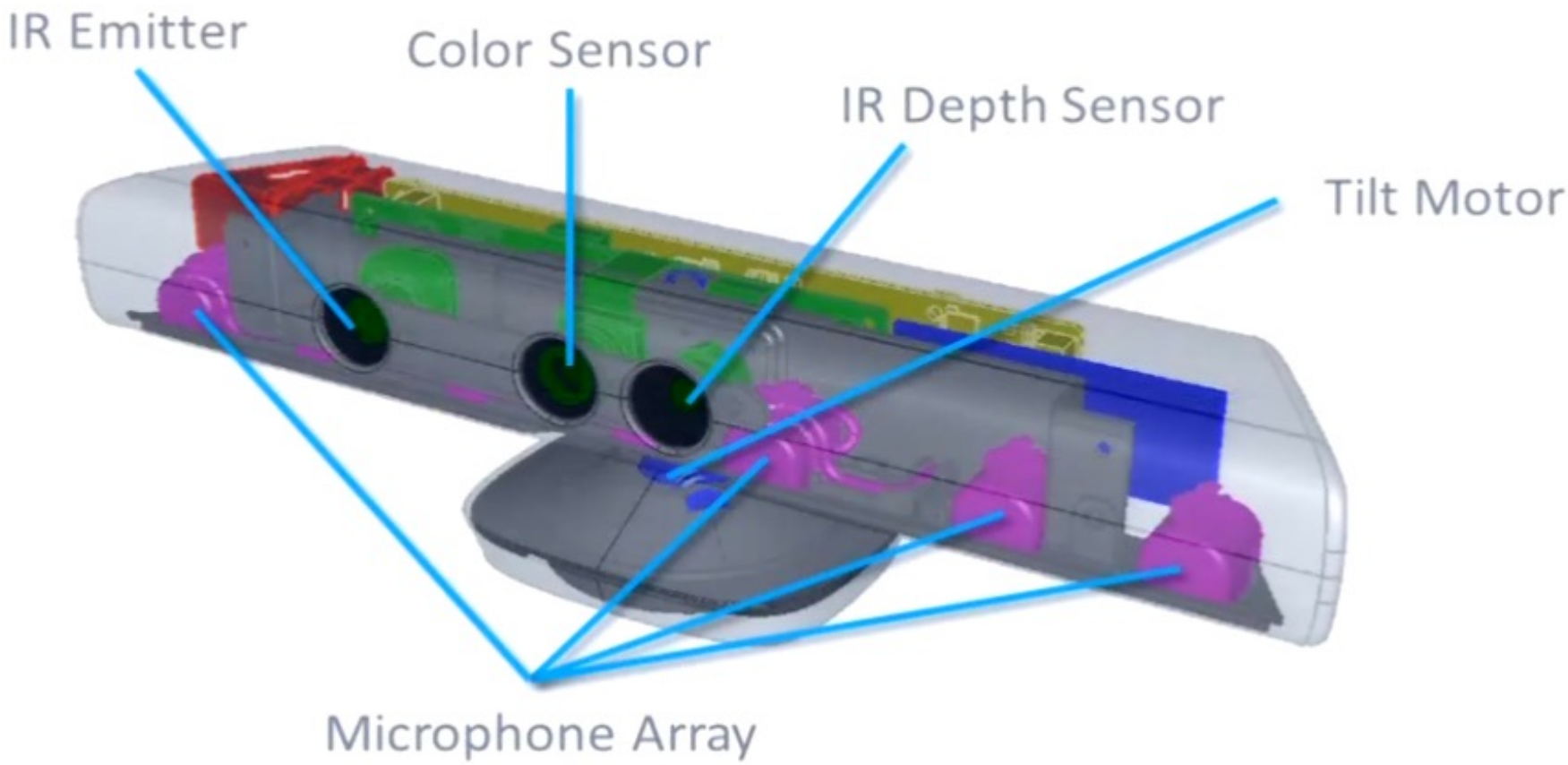
3D Reconstruction

RGBD Camera

- Is a particularly simple case of a stereo rig, in effect, except that it combines a single RGB camera and two infrared ones, the first being an emitter, and the second being a sensor.



RGBD Camera


IR Emitter
Color Sensor
IR Depth Sensor
Tilt Motor
Microphone Array

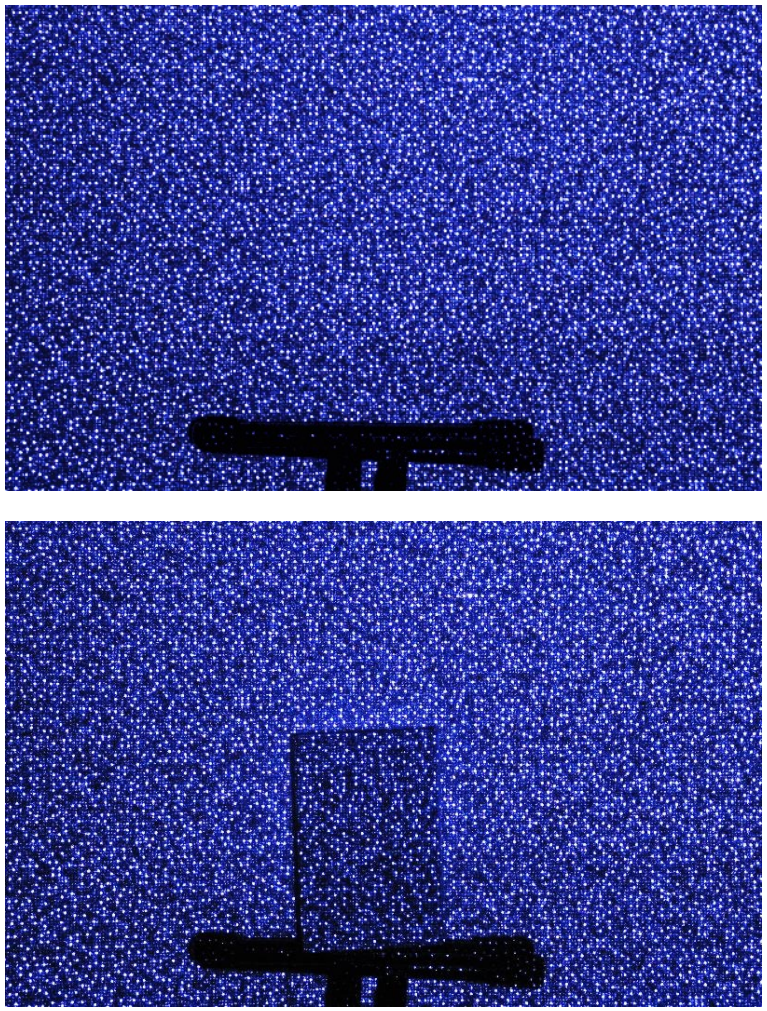
Depth Image

How does it work?

- First , observe that the IR emitter and IR sensor define a stereo system. If we treat the world coordinate frame as being associated with the position of the IR emitter, then $R_1=R_2^T = I_{3x3}$ and T_1 is just the IR sensor's position in the first camera's coordinate frame. In other words, the stereo geometry as given by the structure of the essential matrix E is very simple.
- Now imagine that the emitter is always displaying a fixed, pre-defined (structured light) pattern, which the IR sensor is sensing and is using to automatically compute matches. Then, 3D reconstruction becomes possible!

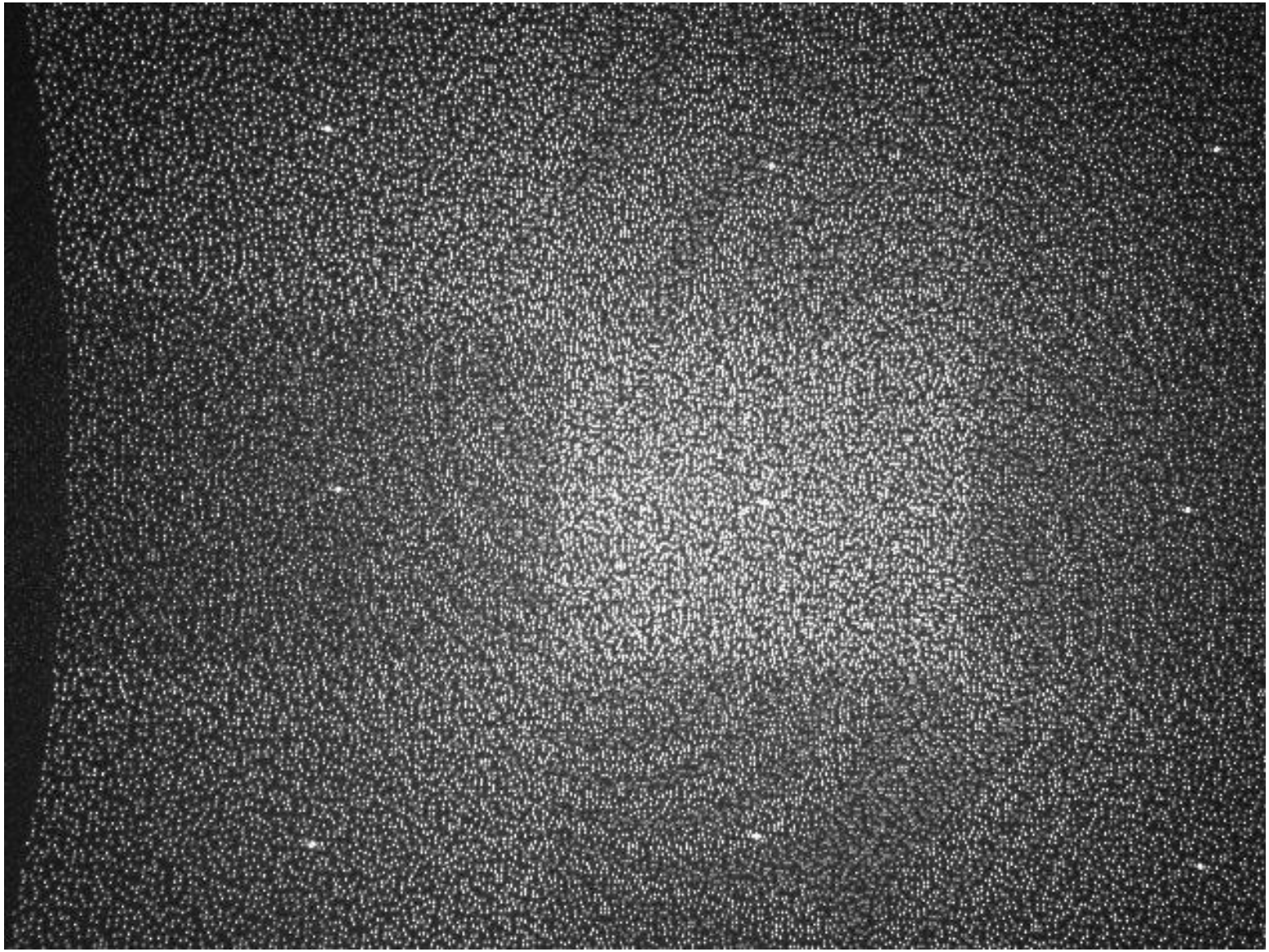
A Pattern of Dots

- IR emitter emits predefined Dotted Pattern
- Lateral shift between emitter and sensor
- Shift in dots determines Depth of Region



A Pattern of Dots

The Kinect Pattern

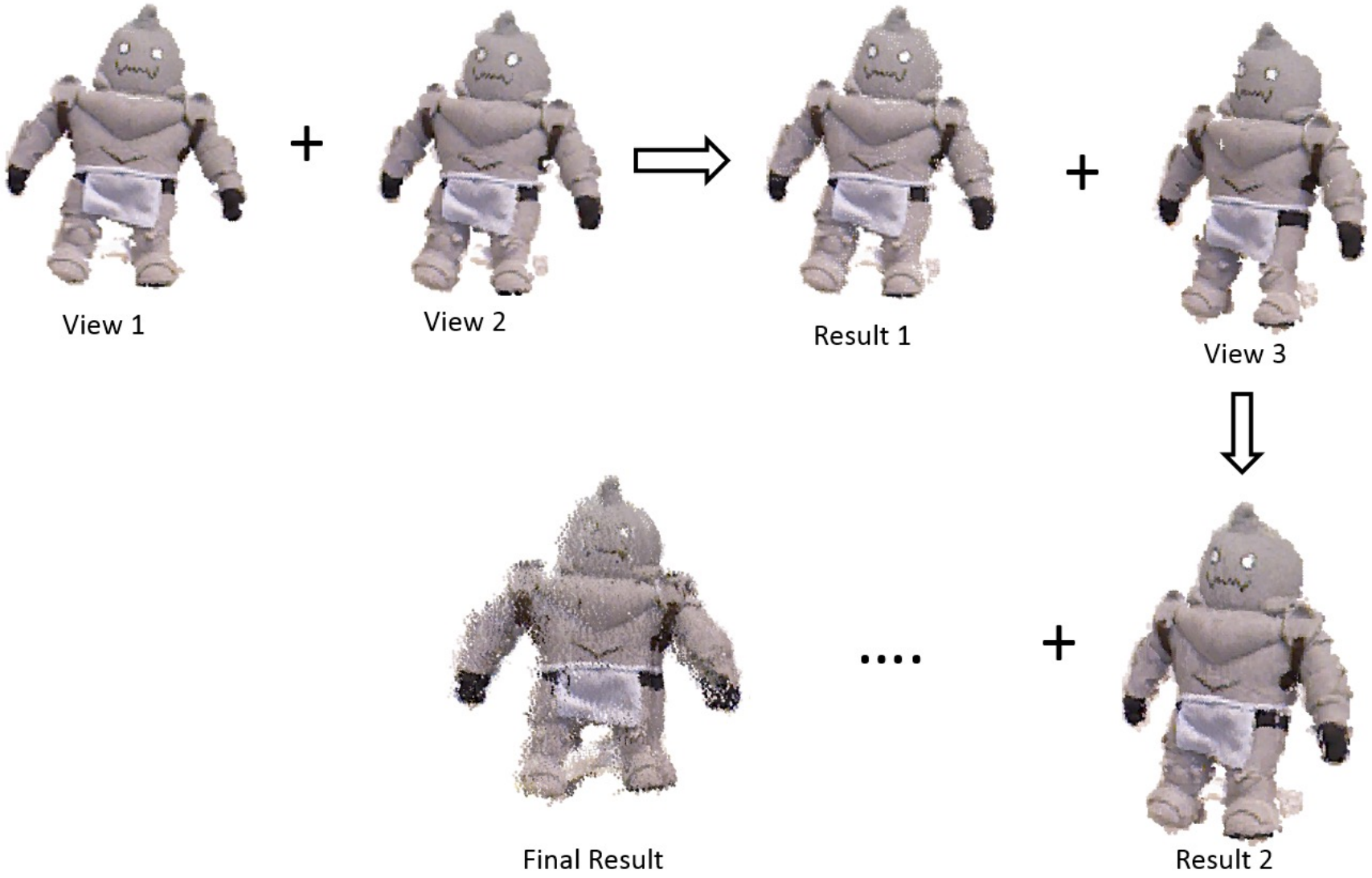


3D Reconstruction

RGB (single view)

RGB rendered on point cloud

How Do We Align Different Views?



A Pair of Views

- We are assuming that the object being sensed is rigid and that the camera motion between successive views is small.
- Therefore, given a point cloud sensed from view 1, and a point cloud sensed from view 2, the two would be related by the (unrecorded) camera motion, or equivalently the rotation and translation of the camera from view 1 to view 2.
- The trick now is to find correspondences between sensed 3D points from the 2 views.
- If we can do this for a sufficient number of points, we can estimate the rotation and translation.

Iterative Closest Points (ICP)

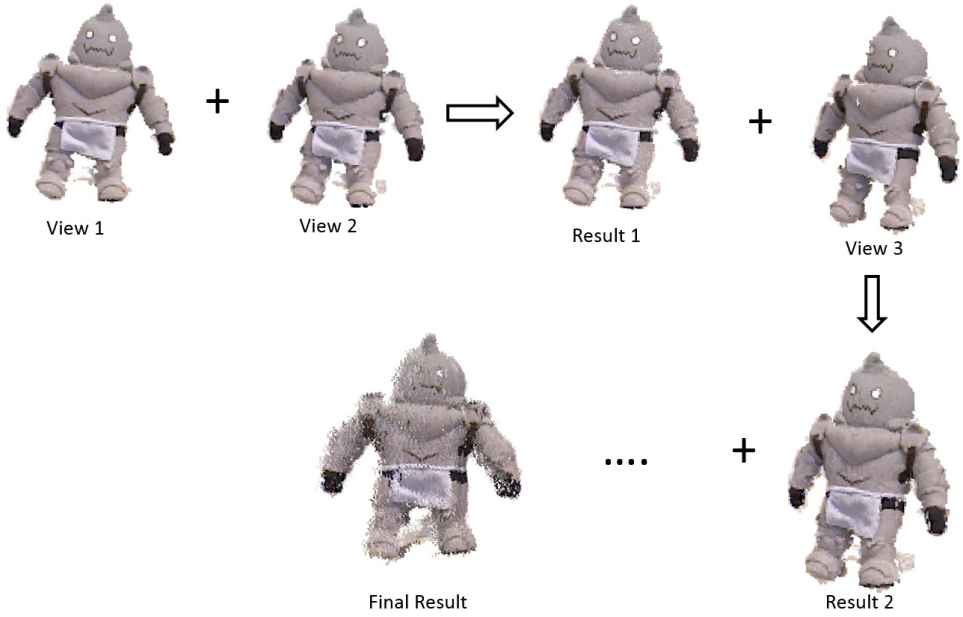
- The ICP algorithm, due to Besl and Mckay (1992), is an iterative algorithm to register 2 sets of geometric entities in 3D (points, lines, etc.) under the assumption that they are related by a rigid transformation (a rotation and a translation).
- Let S_1 be the set of points in camera 1's coordinate frame, which we take to be the world coordinate frame, and S_2 be the same set of points viewed by camera 2.
- The algorithm seeks to find the best rigid transformation in the sense that when applied to S_1 the transformed points are best aligned with S_2.

Iterative Closest Points (ICP)

- The typical notion of a good alignment for point clouds is a point to point distance, but other measures can also be used.
- The algorithm is as follows:
  - 1) For each point in S_1 the closest point in the sense of Euclidean distance is chosen from $S_{2,}$ to be its corresponding point. However, if this distance is beyond a certain tolerance, that point in S_1 is simply ignored.
  - 2) Then the average of the Euclidean distances between the matched points is considered as an objective function to be minimized, and the optimal rigid body transformation (rotation and translation) between S_1 and S_2 is found, in a least squares sense.
  - 3) That transformation is applied to S_1 and then, the process iterates till the average point-to-point error falls below a threshold.

Iterative Closest Points (ICP)

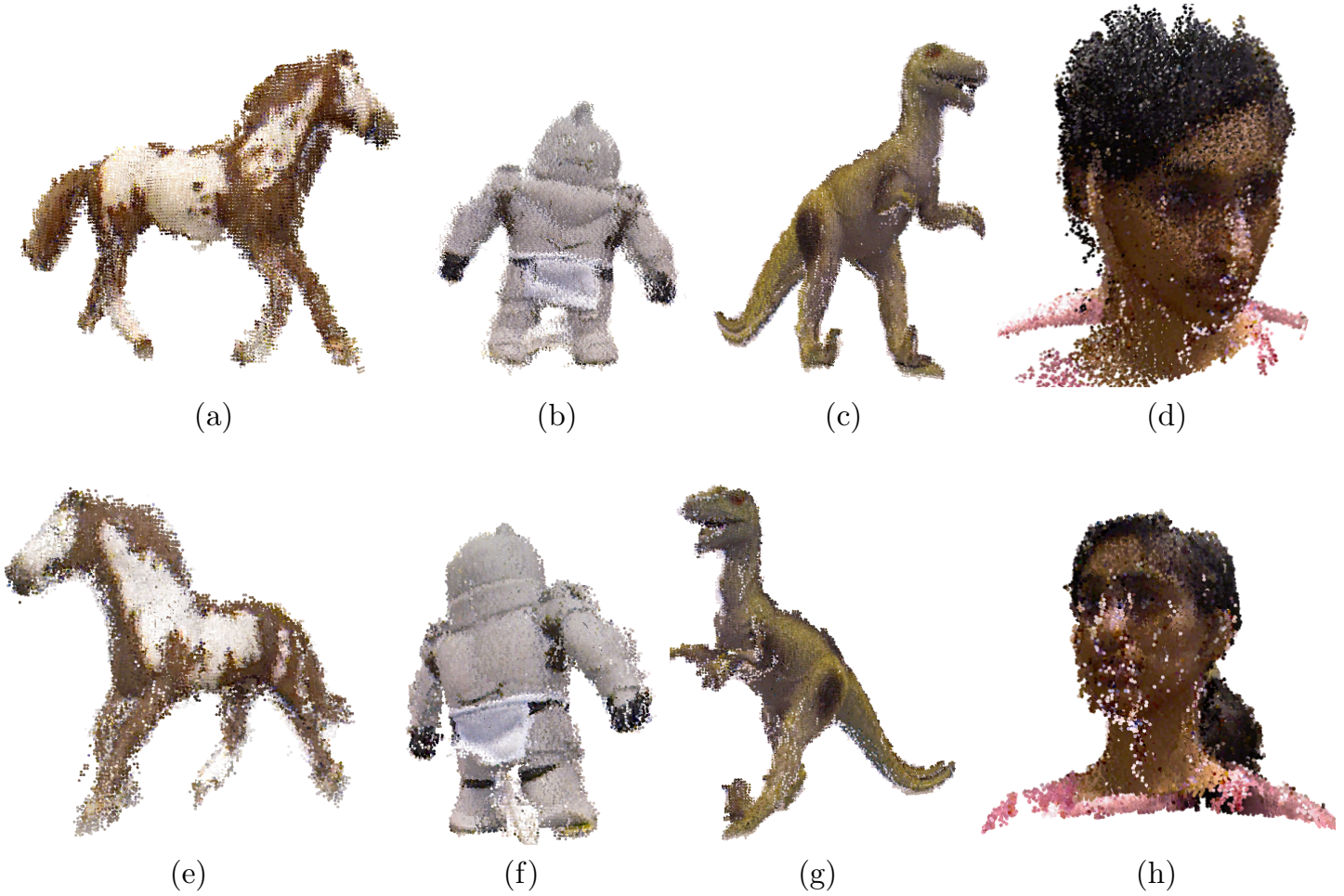
- At the end of this process, there is a sequence of transformations that "aligns" S_1 with S_2, allowing them to be drawn together in camera 2's coordinate frame.
- In the sequential 3D scanning demo we saw in class, ICP is applied between each pair of successive frames so that the merged point cloud can be rendered with respect to the final camera's view.



Iterative Closest Points (ICP)

- Because successive frames see new 3D scene points, this process of recovering the frame-to-frame transformations allows one to scan all the way around an object, in a free form manner. In a sense one has multiple pair-wise stereo problems.
- The technique could be improved by combining it with RANSAC, particularly if one wishes to ignore outliers due to sensing noise. This tends to happen near the occluding contours of objects, where the notion of a match is not always well-defined.

Typical 3D Scans



(a) (b) (c) (d)

(e) (f) (g) (h)

Sricharana Rajgopal, M.Sc. Thesis, SOCS, 2015

Finding the Rotation and Translation

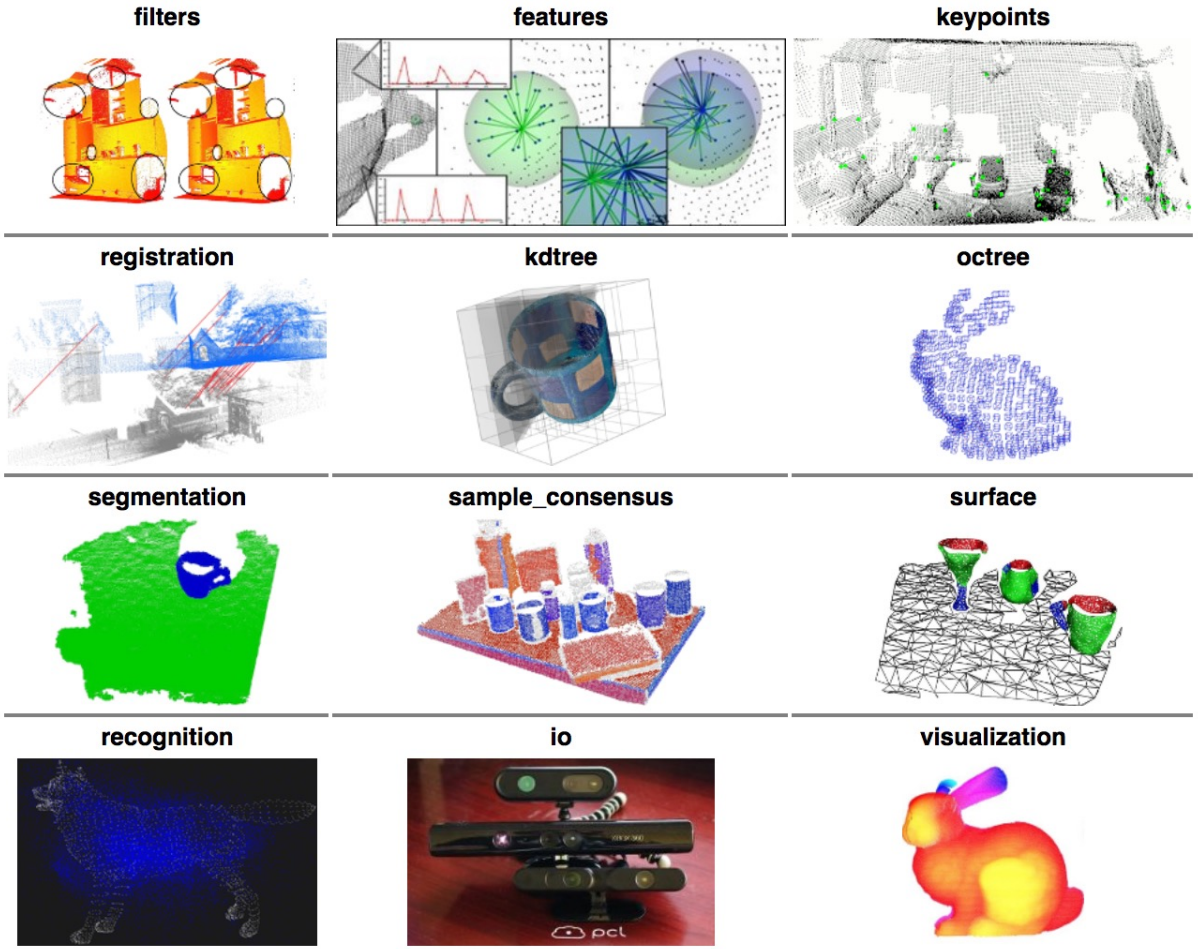
- How is the optimal rigid body transformation found in each iteration of ICP, in a least squares sense? We ignored this detail, but we already know the answer.
- We saw a very similar problem in the lecture where we estimated the matrix **P** for a finite projective camera using least squares.
- The problem though is different in a subtle way, since we do not need to project to pixel coordinates, because we assume the we have the 3D camera coordinates of matched points. In other words, we do not need the matrix **K** of camera intrinsics.

Finding the Rotation and Translation

- Consider the matrix $\begin{bmatrix} R_1 & R_2 & R_3 & q_4 \\ R_4 & R_5 & R_6 & q_5 \\ R_7 & R_8 & R_9 & q_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Then this matrix pre-multiplying a point X_1 in camera 1's coordinates, will give the corresponding point X_2 in camera 2's coordinates. There are 12 unknowns to estimate, the entries of the rotation matrix, and the translation vector (q_4,q_5,q_6). Therefore, we need a certain number of pairwise correspondences to estimate them.
- In the end we end up with something very similar to what we had in Lecture 18 on camera calibration (pp. 1 and 2). We set up a 2N x 12 system of equations, and use SVD.

Software Tools: PCL

- http://pointclouds.org
- Many useful processing functions for handling 3D data



Software Tools: OpenNI

- https://structure.io/openni
- Software for recording frames with support for many different RGBD cameras, including Kinect and Primesense.