Midterm Project
## Midterm Project

<table>
<thead>
<tr>
<th>Activity</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving a target item from a multi-item area into the order bin</td>
<td>+20 points</td>
</tr>
<tr>
<td>Moving a target item from a double-item area into the order bin</td>
<td>+15 points</td>
</tr>
<tr>
<td>Moving a target item from a single-item area into the order bin</td>
<td>+10 points</td>
</tr>
<tr>
<td>Moving a non-target item from the table top</td>
<td>-12 points</td>
</tr>
<tr>
<td>Damaging any item or packaging</td>
<td>-5 points</td>
</tr>
<tr>
<td>Dropping a target item from a height above 0.3 meters</td>
<td>-3 points</td>
</tr>
</tbody>
</table>
Midterm Project

Safety First
Point Cloud Library

Features
Filters
Keypoints
Registration
Segmentation
Sample Consensus
Surface
Range Image
I/O
People
Simulation
Out-of-core
Visualization
Segmentation
2D
ML
Recognition
Kd-tree
Octree
Point Cloud Library

- PCL can deal with both **organized** (e.g. range maps) and **unorganized** point clouds.
  - If the underlying 2D structure is available, efficient schemes can be used (e.g. **integral images** instead of kd-tree for nearest neighbor search).
- Both are handled by the same data structure (**pcl::PointCloud**, templated thus highly customizable).
  - Points can be XYZ, XYZ+normals, XYZI, XYZRGB, ...
  - Support for **RGB-D data**
- **Voxelized** representations are implemented by **pcl::PointCloud + voxelization functions** (e.g. voxel sampling).
  - No specific types for voxelized maps.
- Currently rather limited support for **3D meshes**
3D Object Recognition Using PCL

- **Definition (typical setting):**
  - A. a set of 3D models (often, in the form of views)
  - B. one scene (at a time) including one or more models, possibly (partially) occluded, + clutter.

- Models can be present in **multiple instances** in the same scene

- **Goal(s):**
  - determine which model is present in the current scene
  - (often) estimate the 6DoF pose of the model wrt. the scene

- Applications: industrial robotics, quality control, service robotics, autonomous navigation, ..
Representation for 3D Object Recognition

- Usually Object Recognition in clutter is done on 2.5 data (model views against scene views)
- Can be done also 3D vs 3D, although scenes are usually 2.5D (and 3D vs. 2.5D does not work good)
- When models are 3D, we can render 2.5D views simulating input from a depth sensor:

```cpp
pcl::apps::RenderViewsTesselatedSphere render_views;
render_views.setResolution (resolution_);
render_views.setTessellationLevel (1); // 80 views
render_views.addModelFromPolyData (model); // vtk model
render_views.generateViews ();
std::vector< pcl::PointCloud<pcl::PointXYZ>::Ptr > views;
std::vector< Eigen::Matrix4f > poses;
render_views.getViews (views);
render_views.getPoses (poses);
```
Pipeline: 3D Object Recognition