3D Perception: Point Cloud Data Processing and Visualization

Pat Marion

Website: www.kitware.com   Email: pat.marion@kitware.com

Scientific Computing, Kitware, Inc, 28 Corporate Drive, Clifton Park, NY 12065.

Point Cloud Library

The Point Cloud Library (PCL) is a large scale, open-source project for 2D/3D image and point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting, and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, and create surfaces from point clouds and visualize them.

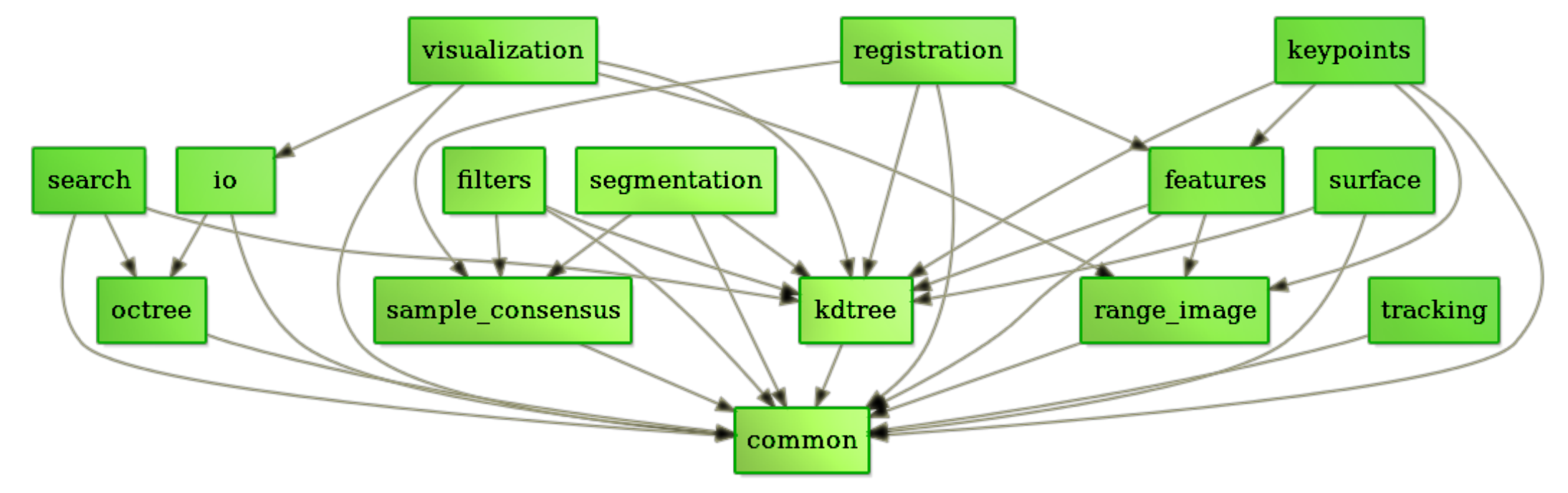

Figure 1: PCL modules (C++ libraries) displayed as a dependency graph.

PCL contains a visualization module that offers a convenient API for users to quickly prototype scenes consisting of point clouds, meshes, and 2D images and histograms. The PCL visualization module is based on the Visualization Toolkit (VTK).

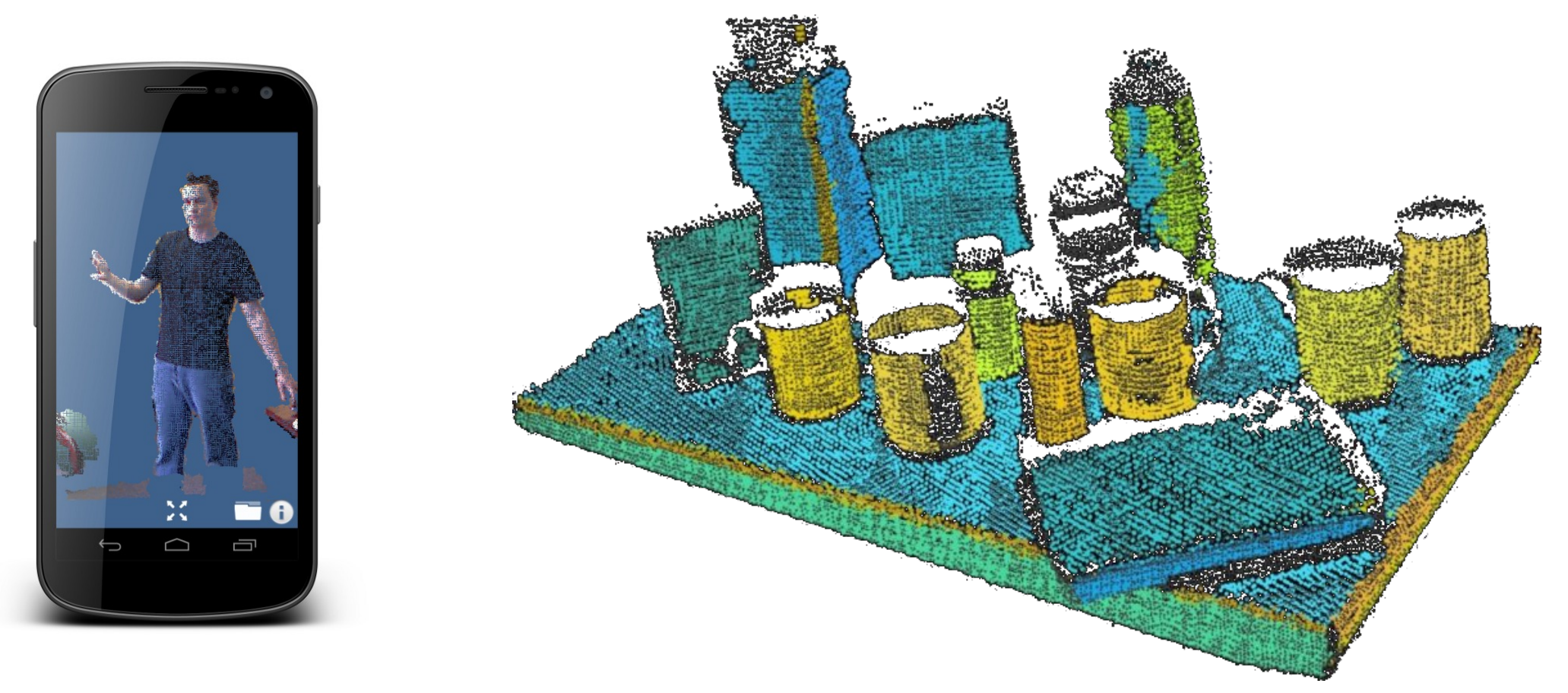

Figure 2: Left, PCL running on an Android mobile phone. Right, a point cloud scene segmentation using cylinder and plane sample consensus filters.

Visualization Toolkit and ParaView

The Visualization Toolkit (VTK) is an open-source, C++ toolkit supporting a wide variety of visualization algorithms including: scalar, vector, tensor, texture, and volumetric methods; and advanced modeling techniques such as: implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation.

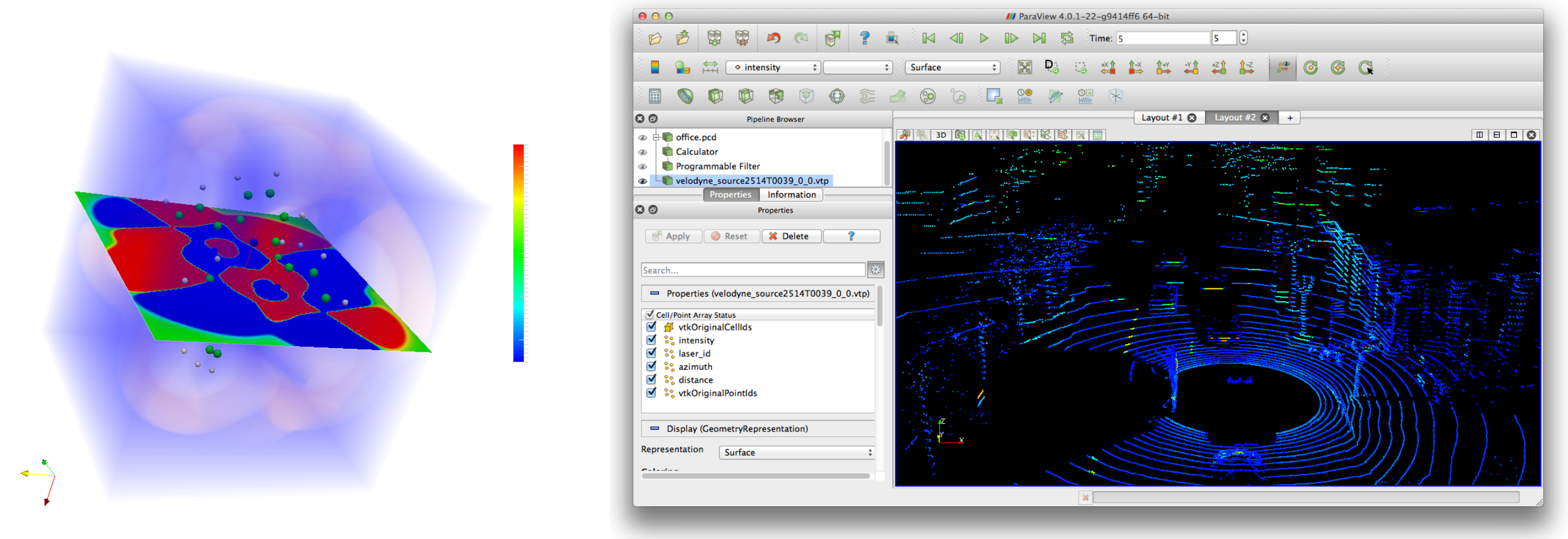

Figure 3: Volume rendered molecular orbital with sliced contour (left), and ParaView user interface with LiDAR point cloud data (right).

ParaView is an open-source, cross-platform data analysis and visualization application. It is one of the flagship open-source projects developed by Kitware, building on VTK and Qt to provide a client-server application that allows users to quickly build visualizations to analyze their data. ParaView was developed to analyze extremely large data sets using distributed memory computing resources. It can be used interactively with the cross-platform GUI or scripted from Python.

PCL Plugin for ParaView

The PCL plugin for ParaView allows users to access filters from PCL within ParaView. The plugin wraps PCL algorithms as VTK filters. The plugin also provides Python bindings for the filters using VTK's python wrapping, thus enabling fast prototyping and integration with NumPy and SciPy. With point cloud data loaded in ParaView, users can interactively apply PCL algorithms, color the point clouds by different attributes, and quickly compose complex processing pipelines to explore the point cloud data.

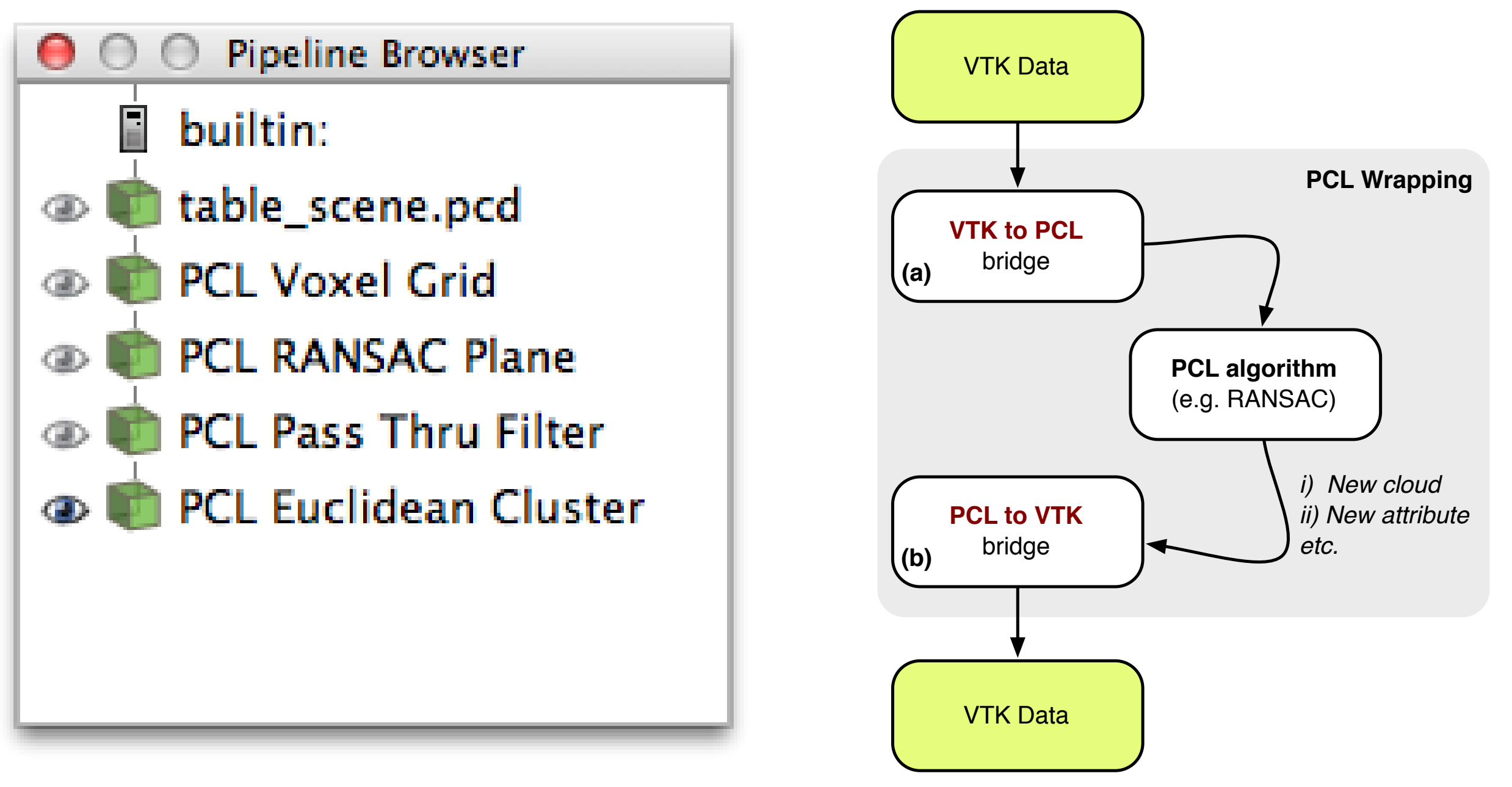

Figure 4: Left, the ParaView pipeline browser showing a data filtering pipeline consisting of PCL filters. Right, the VTK-PCL bridge is used to wrap PCL filters in VTK.

PCL Python Bindings: pcl-python

The *pcl-python* project provides a set of python bindings to PCL. It is implemented using Cython to wrap the heavily templated API of PCL. The pcl-python API tries to follow the PCL C++ API, and also provides helper functions for interacting with NumPy. The following example code uses *pcl-python* to perform smoothing:

```
import pcl

p = pcl.PointCloud()
p.from_file("table_scene_lms400.pcd")

fil = p.make_statistical_outlier_filter()
fil.set_mean_k(50)
fil.set_std_dev_mul_thresh(1.0)
fil.filter().to_file("inliers.pcd")
```

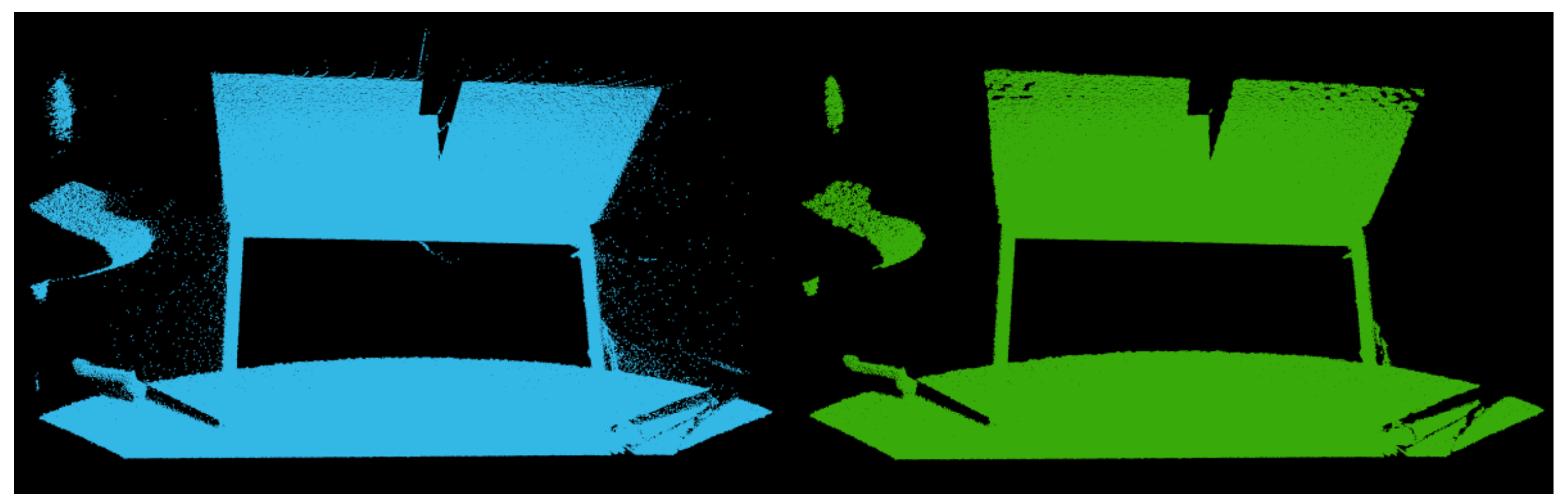

Figure 5: Left, the input point cloud. Right, the filtered result of the above Python example.

The *pcl-python* project is supported by, and currently in production use at Strawlab. The following features of PCL, using PointXYZ point clouds, are available:

- I/O and integration; saving and loading PCD files
- segmentation
- sample consensus model fitting (RANSAC + others)
- smoothing (median least squares)
- filtering (voxel grid downsampling, passthrough, statistical outlier removal)

Live Sensor Data Acquisition

The *pcl_io* module contains classes and functions for reading and writing point cloud data (PCD) files, as well as capturing point clouds from a variety of sensing devices. PCL is agnostic with respect to the data sources that are used to generate 3D point clouds. The *pcl::Grabber* interface provides the base class for device driver implementations.

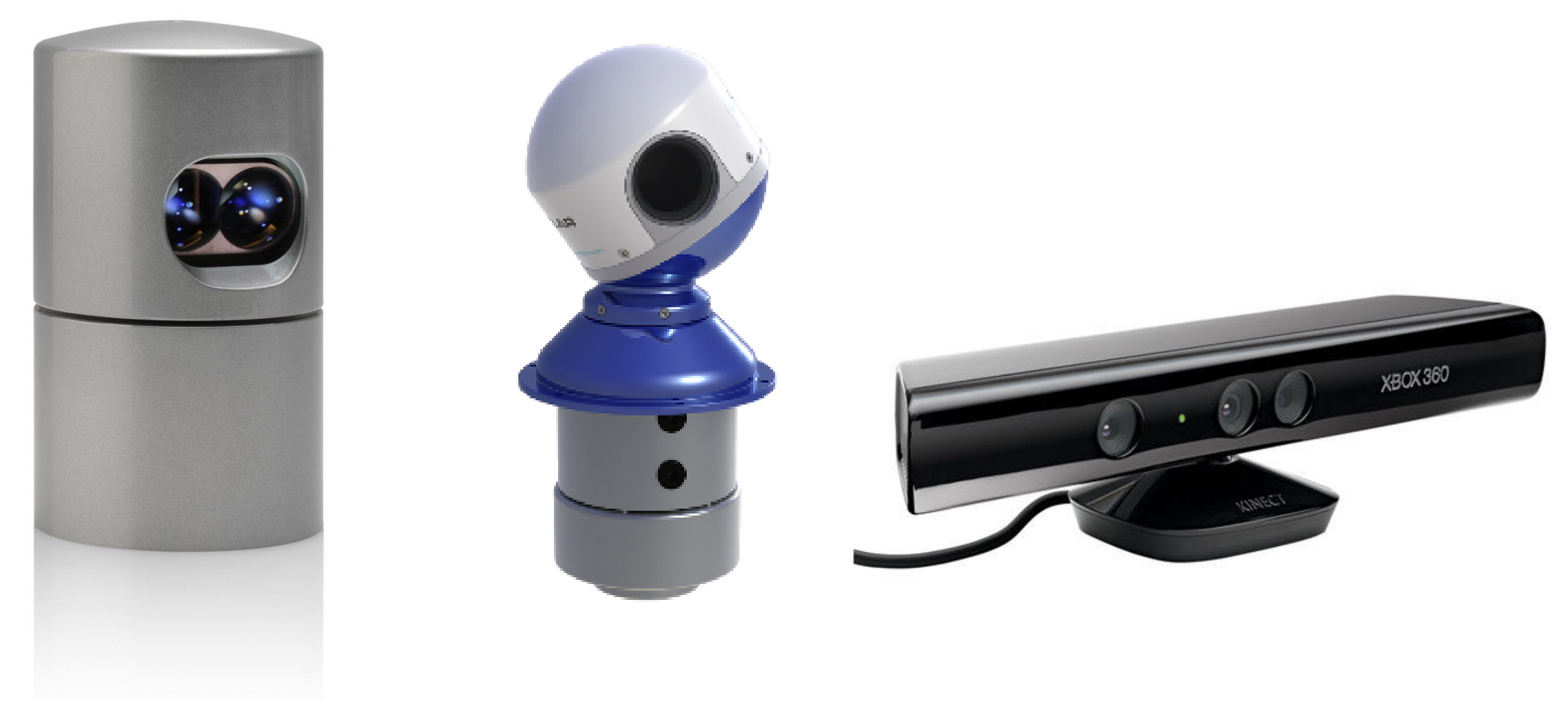

Figure 6: Velodyne HDL-32e (left), Ocular Robotic RobotEye (middle), Microsoft Kinect (right).

Point Cloud Processing with PCL and NumPy/SciPy

Both the PCL Plugin for ParaView and the pcl-python bindings provide conversions between PCL point cloud data structures and NumPy ndarray objects. The VTK Python bindings used by the PCL Plugin for ParaView support zero-copy array access of the point data and related attributes, but require a copy when converting PCL to VTK as in Figure 4.

```
from paraview.simple import *

OpenData('office.pcd')
ProgrammableFilter(Script=readFile('scipy_cluster_demo.py'))

# save the result to pcd and save a screenshot
WriteData('clustered.pcd')
Show(ColorArrayName='labels')
WriteImage('screenshot.png')

########################
# scipy_cluster_demo.py
from scipy.cluster import vq
rgb = inputs[0].PointData['rgb_colors']
centroids, labels = vq.kmeans2(rgb, k=4)
output.PointData.append(labels, 'labels')
```

Example 1: Point cloud processing pipeline implemented using the paraview.simple module.

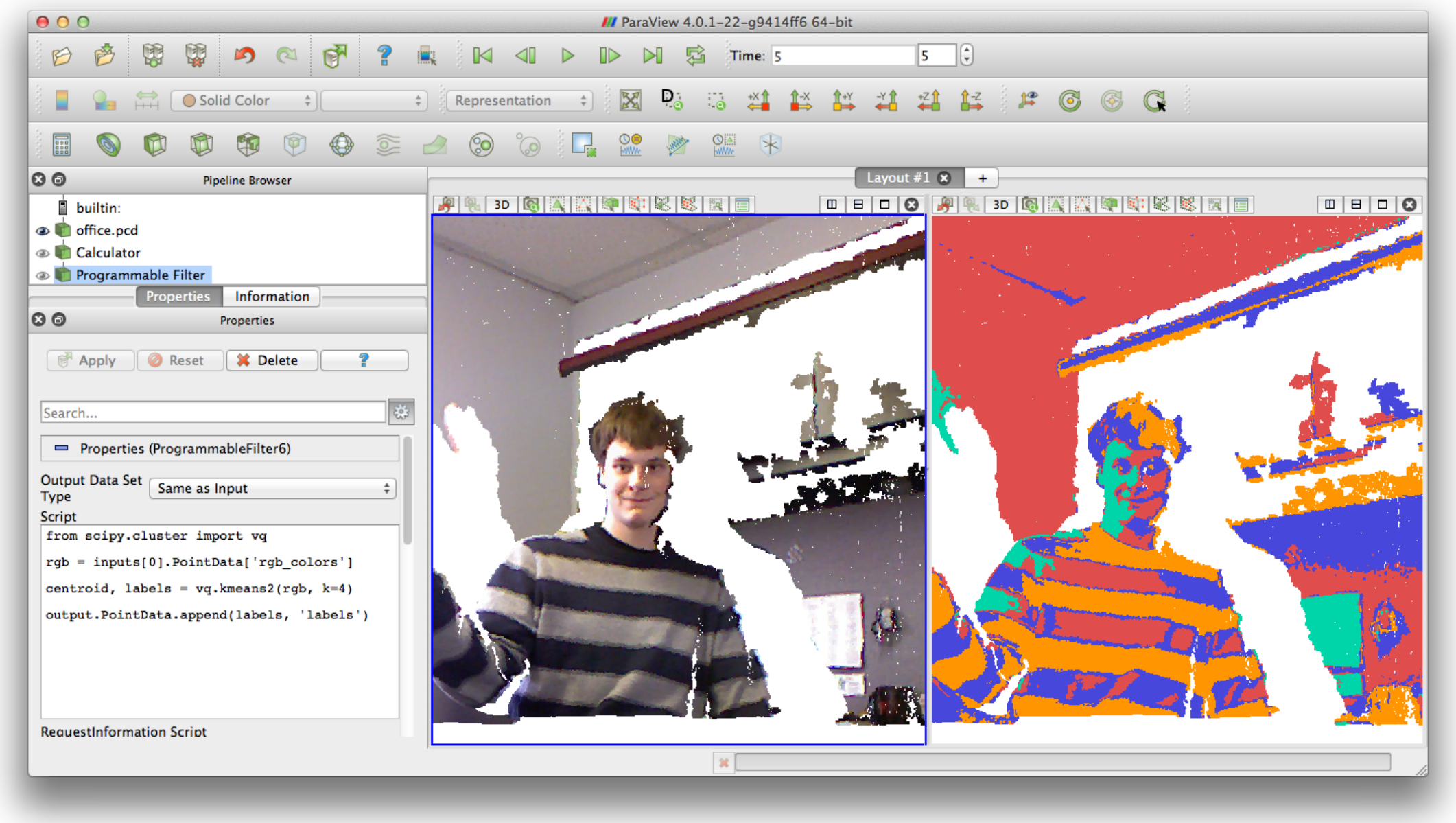

Figure 7: ParaView showing the result of the color segmentation demo. The original point cloud (left) and the point cloud with RGB labels (right).