Kinetic Shape Reconstruction – User manual

This document provides instructions for running the executables KSR-GUI.exe and KSR-CL.exe, which provide implementations of the paper “Kinetic Shape Reconstruction” (TOG) by Jean-Philippe Bauchet and Florent Lafarge.

1. KSR-GUI

KSR-GUI is the user interface version of the paper “Kinetic Shape Reconstruction”.

1.1. Input point cloud

KSR.exe opens a user interface. In the top right corner, the user can notice a button “Load point cloud”. A file dialog pops up and invites the user to open an oriented point cloud. Supported formats are PLY and VG (acronym for Vertex Group: point cloud and clusters of points). The latter was introduced by Nan et al in “Polyfit: Polygonal Surface Reconstruction from Point Clouds” (ICCV 2017), we refer the reader to the project’s website for a complete description.

1.2. Shape detection

The first step of our reconstruction pipeline consists in detecting a set of planar primitives in the point cloud.

If the input is a point cloud without precomputed clusters of points (PLY file), the user should set a minimal number of inliers per plane, a tolerance distance (epsilon), and then press the button “Detect primitives”.

A regularization algorithm (“Regularize primitives”) may be applied to the primitives. It slightly modifies plane equations in order to favor parallelism, orthogonality and coplanarity relationships between extracted planes. It can be relevant to run this procedure when the input point cloud represents a man-made structure. The regularization mostly relies on the CGAL::regularize_planes function. More information is available in the CGAL documentation.

If the input is a VG file, the user should still click on the button “Detect primitives” to compute a set of planar primitives based on the provided clusters of points.

1.3. Kinetic partition

Once planes are extracted and possibly regularized, the user clicks on “Compute partition” to generate a kinetic partition of a space. The maximal number of intersections has to be set before, using the spin box above. Alternatively, the user might open an existing kinetic partition, saved in the KGRAPH file format (see section 3 below) computed from this set of primitives using the button “Load partition”.

1.4. Surface extraction

Finally, we invite the user to run the surface extraction module of our reconstruction pipeline. As explained in our paper, this method is based on a visibility criterion and min-cut/max-flow optimization. The user sets the value of the balancing term lambda, and then clicks on “Compute surface”. The input point cloud is approximated as a concise polygonal mesh.

1.5. Output

The button “Save data” offers the user the possibility to keep trace of his or her work if it is satisfying enough. The user is first requested to select a folder for saving the different outputs, which will be themselves stored in a generated subfolder (by default named after the point cloud and the current date). Three types of products can generated by this executable, related to the planar shapes, the kinetic partition or the final surface.

First, there exists different ways to save the detected planar shapes. One option is to save the point cloud and the detected clusters of points in a single VG file. Detected alpha-shapes, bounding rectangles and convex hulls can be saved in three separate PLY files.

Second, the kinetic partition can be saved in a PLY file or a KGRAPH file.

Finally, the approximation of the input cloud as a concise, polygonal mesh can be saved into a PLY file. It is also possible to save facets’ contours in a separate PLY file.

1.6. Advanced parameters

The button “Load advanced parameters” opens a dialog with parameters related to the different steps of the algorithm, and classified by categories.

1.6.1. Shape detection

The planar shape detection is based on a region growing algorithm: a plane hypothesis is propagated from a vertex in the point cloud to its closest neighbors. A vertex is matched to the current plane if its normal makes an angle with the plane that is smaller than a certain value: in our implementation, we perform a comparison between the cosine of this angle and a certain threshold (the closer to 1, the smaller the angle). Two spin boxes let the user set the size of the neighborhood, and the value of the aforementioned threshold.

The two next parameters are the regularization parameters: they control the maximal angle variation and displacement that can be applied to each plane by the CGAL::regularize_planes function, that optimizes parallelism, orthogonality and coplanarity relationships between all the detected planes.

Finally, it is important to note that orientations of the planes are discretized on the unit sphere. A parameter allows the user to control the precision of this operation.

1.6.2. Kinetic partition

The next section lists advanced parameters for the kinetic partitioning algorithm. Three spin boxes let the user define a subdivision scheme for propagating the planar polygons (more information in the paper).

Two checkboxes allow the user to apply a reorientation of the primitives in the 2D or 3D space if needed. The algorithm will simply search for a bounding box of minimal volume enclosing all the polygons and perform a kinetic propagation in that bounding box.

The next checkbox is for computing a kinetic partition of the 3D space using the best bounding rectangles of each extracted alpha-shape, instead of convex hulls, as it is the case by default.

1.6.3. Surface extraction

Depending on the scenes that are processed by the interface, it may be useful to force one of the sides of the bounding box to be part of the object to reconstruct. A typical example is an outdoor scene, when it is often preferable to include the ground in the output surface. A set of buttons allows the user to compare both approaches, with or without forcing some bounding box facets to be included in the reconstruction. Finally, one last checkbox lets the user decide whether the facets of the surface that also belong to the bounding box should be displayed on screen.

2. KSR-CL.exe

KSR-CL.exe is the command-line version of the implementation of the paper "Kinetic Shape Reconstruction". It performs most of the operations performed in KSR-GUI.exe. We refer the reader to the previous section and the aforementioned paper for details.

2.1. Command-line arguments

Here is a list of expected command-line arguments:

`--input [path]`: provides an input point cloud to the algorithm, PLY and VG formats accepted

`--output [name]`: name of output reconstructed surface, .ply suffix expected

`--nmin [value]`, `--epsilon [value]`: the two main parameters of the primitive detection step

`--K [value]`: the K parameter of the kinetic partitioning step

`--grid [x] [y] [z]`: optional subdivision scheme, minimal values are (1, 1, 1)

`--lambda [value]`: balancing term in the surface extraction step.

`--quiet`: turns on the silent mode, no traces will be displayed on screen. Disabled by default.

`--verbose`: turns off the silent mode. Disables `--quiet` and vice-versa.

`--config [path]`: provides a configuration file to the algorithm, with extra options.

2.2. Options of configuration file

A configuration file is a TXT file format with a list of options `parameter_name [value1] [value2] ...` at every new line. Comments are not allowed. Here is a list of expected parameters:

2.2.1. Input/output

As mentioned above, options `input [path]` and `output [name]` respectively provide an input point cloud to the algorithm and the name of the reconstructed surface. In addition to them, options `save_alpha_shapes`, `save_convex_hulls`, `save_partition_ply`, `save_partition_kgraph` write the detected alpha-shapes, convex hulls, and computed partitions in the PLY format, except for the last

command where the KGRAPH file format is used. The names of such files are automatically generated with appropriate suffixes, deduced from the name of the output surface.

2.2.2. Primitive detection

Accepted options are `nmin [value]`, `epsilon [value]` but also `normal_threshold [value]` that sets a cosine value as threshold. This value is used for grouping an oriented point from the point cloud to the current cluster of points, which is summarized by a plane equation. We assume the distance between the point and the plane to be lower than epsilon. Now, if the dot product between both normals is greater than this threshold, then the point is added to the cluster, otherwise it is rejected. Default value for that threshold is 0.85.

2.2.3. Primitive regularization and discretization

Adding the option `regularize` enables the regularization algorithm, via a call the CGAL::regularize_planes function. We offer the offer the possibility to tune the sensitivity of the regularizer using options `tolerance_angle [value]` and `tolerance_coplanarity [value]`.

Moreover, it is important to note that orientations of the detected (or regularized) planes are discretized before running the kinetic partition algorithm. We provide an advanced option for controlling the precision of discretized orientations: `discretization_step [value]`. Like in the GUI version, accepted values are 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 3 and 5 (in degrees). It is also possible to merge parallel planes whose distance from each other is less than the value read by option `discretization_distance_ratio [value]` (interpreted as a percentage of the bounding box diagonal).

2.2.4. Kinetic partition

Proposed options are the same as described in the previous paragraph: `K [value]` and `grid [x] [y] [z]`.

2.2.5. Surface extraction

Similarly to before, we define the balancing term using the keyword `lambda [value]`. In addition, we define options `gc-free-label` (the default policy followed by the pipeline), `gc-imposed-unique-label [label]`, `gc-imposed-labels [xmin] [xmax] [ymin] [ymax] [zmin] [zmax]` and `display_facets_on_bbox` hat implement the functionalities described in paragraph 1.6.3. All read labels should be `0` (outside) or `1` (inside).

2.2.6. Others

Other accepted parameters are `quiet` and `verbose` that control the level of verbosity on screen.

3. Details on the KGRAPH file format

The KGRAPH file format describes a connected graph resulting from the propagation and intersection of multiple planar primitives in a kinetic framework. It is organized as follows:

```
[P, number of primitives] [H, number of planes] [V, number of vertices]
[E, number of edges] [F, number of facets] [C, number of polyhedra] //
six integers in total
primitives
```

```
0 [plane index for primitive 0]
1 [plane index for primitive 1]
…
P [plane index for primitive P]
planes
0 a_0 b_0 c_0 d_0
1 a_1 b_1 c_1 d_1
…
H a_H b_H c_H d_H
```

Where each quadruplet (a_i, b_i, c_i, d_i) represents the equation of plane i. Each coefficient is written as a quotient of arbitrary long integer numbers. Then, here comes a definition of connected 3D graph:

```
vertices
i [number of planes intersecting in vertex i] [indices of these planes]
// ... V lines in total
Edges
i [index of first vertex of edge i] [index of second vertex of edge i]
// ... E lines in total
facets
i [index of supporting plane for facet f_i] [number of halfedges delimiting
f_i] [the list of halfedges]
// ... F lines in total
```

A list of halfedges in a sequence of pairs `[e_j orientation]`. Given an edge e_j = (v1 v2), a value of 1 (resp. 0) indicates that consider the halfedge (v1 -> v2) (resp. v2->v1). Once all facets are written, we list polyhedrons of the kinetic partition.

```
polyhedrons
i [number of sides of polyhedron p_i] [list of sides]
// … C lines in total
```

We formulate a side of the polyhedron as a pair `[f_j orientation]`. The facet f_j refers to one of the facets listed before, and orientation is a boolean. A value of 1 (resp. 0) indicates that the barycenter of the polyhedron belongs to the positive (resp. negative) half-space delimited by the equation $a_k x + b_k y + c_k z + d_k = 0$ where k is the index of the plane supporting the facet f_j.