User manual: Polyhedral splines

JÖRG PETERS[*], University of Florida, USA
KYLE SHIH-HUANG LO[*], University of Florida, USA
KĘSTUTIS KARČIAUSKAS, Vilnius University, Lithuania

ABSTRACT

This user and developer guide provides usage and testing examples of polyhedral spline algorithms, accompanying the article

> Jörg Peters, Kyle Shih-Huang Lo, Kęstutis Karčiauskas. Algorithm xxx: Polyhedral splines. *ACM Trans. Math. Softw.* xx, x, Article xx (202x), xx pages.

The package is implemented in the C++ programming language and can be compiled and executed either in the Linux, macOS or Windows operating systems.

1 INSTALLATION

Environment settings: Succesfully tested versions are listed in parentheses.

- Operating system: Linux (Ubuntu 20.04 LTS) or macOS (Catalina 10.15) or Windows 10
- Dependencies: OpenMesh (8.1), CMake (3.16.3)
- Compiler: g++ (9.3.0) or Apple clang++ (11.0) or Visual Studio 2017

Note:

- CMake creates a makefile that will automatically download and install OpenMesh into `/Source/External` using the source with commit hash pointing to the tested version. Note that, depending on internet connectivity, the download of external libraries may allow for a coffee break: OpenMesh ~60 MB and the remainder, including the submission code, ~10 MB.
- The implementation is expected to be compatible with equivalent or higher versions than those listed in parentheses, with little or no modification.

Compilation:
For UNIX-based system:

```
$ unzip polyhedral_spline.zip
$ cd ./polyhedral_spline
$ mkdir build
$ cd ./build
$ cmake ../Source
$ make
```

Note: macOS users need to make sure `$PATH` includes path to qt5 bin folder

Authors' addresses: Jörg Peters, University of Florida, Gainesville, USA, jorg.peters@gmail.com; Kyle Shih-Huang Lo, University of Florida, Gainesville, USA, kyles; Kęstutis Karčiauskas, kestutis.karciauskas@mif.vu.lt, Vilnius University, Lithuania.

For Windows:

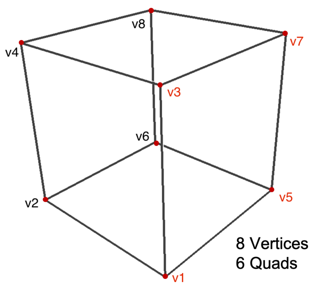
1. Launch `x86 Native Tools Command Prompt for VS 2017`
2. Run commands for UNIX-based system except for `make`
3. Launch `PolyhedralSplines.sln` with Visual Studio 2017
4. Set configuration to `Release` mode and switch platform to `Win32`
5. Build solution

Note: Windows users need to make sure environment variables include the path to the QT bin folder

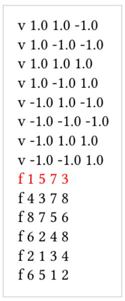
2 USAGE

Fig. 1 juxtaposes the visual and the file representation of input and output of a cube.

- Input: mesh (with semi-structured layout) in `.obj` file format
- Output: BB-coefficients in `.bv` file format. The BView file format (.bv) is described at (https://www.cise.ufl.edu/research/SurfLab/bview/#file-format)

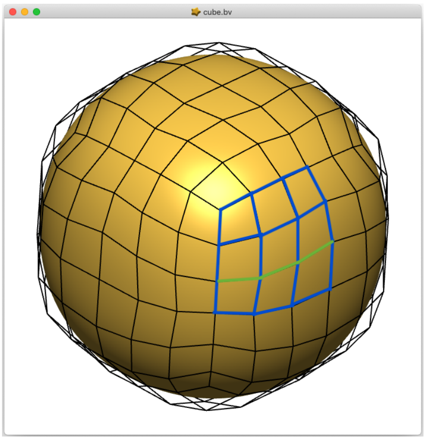


(a) cube with the four vertices of one face in red.

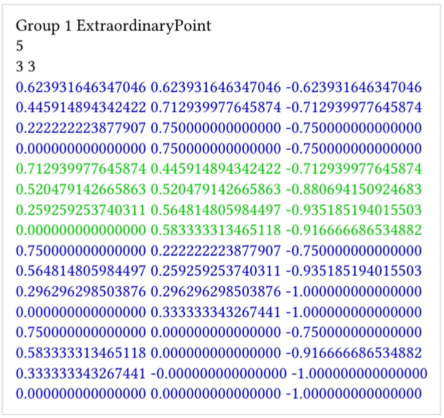


```
v 1.0 1.0 -1.0
v 1.0 -1.0 -1.0
v 1.0 1.0 1.0
v 1.0 -1.0 1.0
v -1.0 1.0 -1.0
v -1.0 -1.0 -1.0
v -1.0 1.0 1.0
v -1.0 -1.0 1.0
f 1 5 7 3
f 4 3 7 8
f 8 7 5 6
f 6 2 4 8
f 2 1 3 4
f 6 5 1 2
```

(b) Input: cube.obj with vertex indices of one face marked red.



(c) A bi-3 BB-net of 4 × 4 BB-coefficients of one of the 24 patches is marked in blue except one row in green.



```
Group 1 ExtraordinaryPoint
5
3 3
0.623931646347046 0.623931646347046 -0.623931646347046
0.445914894342422 0.712939977645874 -0.712939977645874
0.222222223877907 0.750000000000000 -0.750000000000000
0.000000000000000 0.750000000000000 -0.750000000000000
0.712939977645874 0.445914894342422 -0.712939977645874
0.520479142665863 0.520479142665863 -0.880694150924683
0.259259253740311 0.564814805984497 -0.935185194015503
0.000000000000000 0.583333313465118 -0.916666686534882
0.750000000000000 0.222222223877907 -0.750000000000000
0.564814805984497 0.259259253740311 -0.935185194015503
0.296296298503876 0.296296298503876 -1.000000000000000
0.000000000000000 0.333333343267441 -1.000000000000000
0.750000000000000 0.000000000000000 -0.750000000000000
0.583333313465118 0.000000000000000 -0.916666686534882
0.333333343267441 -0.000000000000000 -1.000000000000000
0.000000000000000 0.000000000000000 -1.000000000000000
```

(d) Output file cube.bv listing 4 × 4 rows of x, y, z colored corresponding to (c).

Fig. 1. Example showing geometry *left* and file format *right*. Run `./PolyhedralSplines ../testfile/cube.obj` on input file (b) illustrated in (a) to generate cube.bv. (d) Load cube.bv into BView to display the BB-nets in (c). The blue and green entries in (d) correspond to the highlighted 4 × 4 BB-net in (c).

Execution:
For UNIX-based systems:

```
$ ./PolyhedralSplines /path/to/filename.obj
```

Example: ./PolyhedralSplines ../testfile/cube.obj
Note: test .obj files are in `/testfile`.

To raise the degree of all patches to a uniform degree 3×3 use the option `-d` or `-DEGREE_RAISE`:

```
$ ./PolyhedralSplines -d /path/to/filename.obj
```

For Windows:

```
$ PolyhedralSplines.exe /path/to/filename.obj
```

```
$ PolyhedralSplines.exe -d /path/to/filename.obj
```

View the output .bv file:
Users can view the surfaces defined by .bv files using either the online or the desktop version of BView. BView provides surface inspection tools such as patch coloring, Gaussian curvature, highlight lines and more.
(https://www.cise.ufl.edu/research/SurfLab/bview/)

3 PROGRAM STRUCTURE

Fig. 2 shows the coding diagram of the implementation of polyhedral splines. The solid arrows on top define the data flow. The dashed lines indicate dependency on the structure pointed to. The outside package `OpenMesh` provides graph (half-edge) data structures to traverse the input mesh. The other key data structures are `Matrix` and `Patch`. Each instance of `Patch` contains the components of a bivariate patch in BB-form. `Pool` initializes and keeps the pointers to instances (algorithms) defined in `PatchConstructor`.

A developer can replace the Mesh reader by internal routines that send a mesh to `ProcessMesh`. Developers can write their own `PatchConsumer` routine 'Other' to work with the polyhedral splines output, e.g. to compute the derivative of the patch in the u-direction. (PatchConsumer.hpp contains a commented code snippet for computing the derivative of a BB-patch in the u direction – by scaling, by the degree in u, the difference of BB-coefficients $[i][j]$ and $[i-1][j]$.) The distribution comes with two instances of `PatchConsumer` that convert the patches, to Bezierview `.bv` and IGES 128 `.igs` format, respectively, with `.bv` the default.

4 TEST RESULTS

Table 1 shows some of the provided models. Tables 2-5 list all supported test configurations. The figures of the input mesh and of the BB-net are independently scaled.

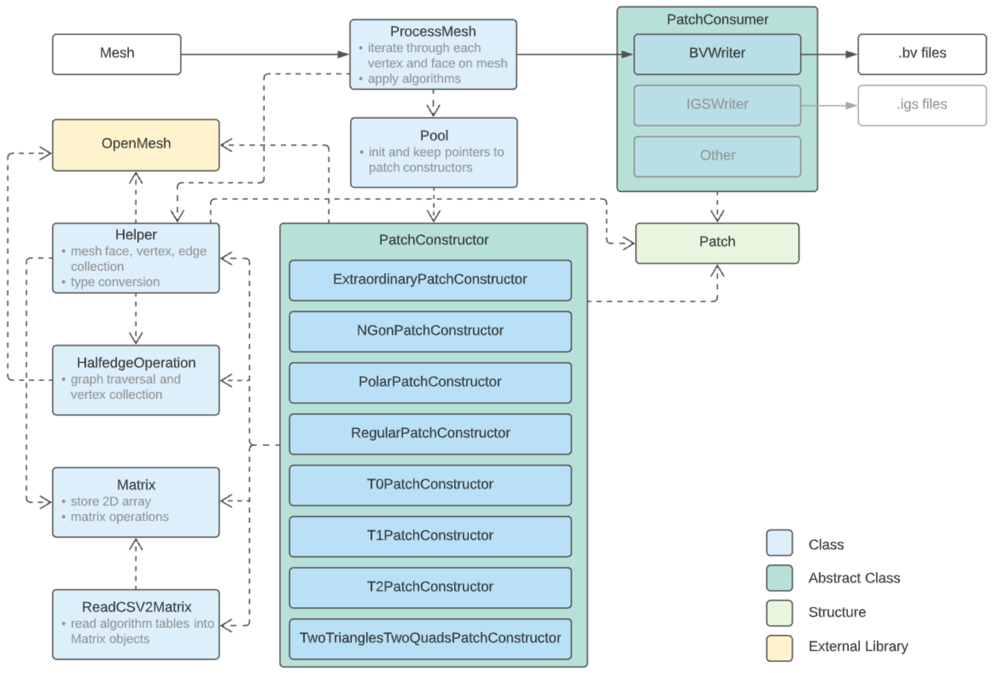


Fig. 2. The coding diagram of Polyhedral splines. Solid arrows represent data flow, dashed arrows dependence.

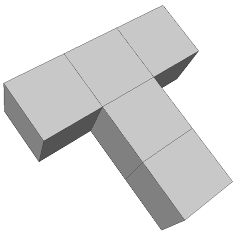
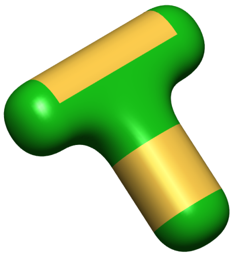
5 LICENCE

LPGL. The intent is

- research use: free and cite this article.
- commercial use: contact the first author.

Table 1. Some of the test models: control net pattern and its semi-structured spline.

Config. Type	Input (.obj)	Output (.bv)

Tee
 	
gold=Regular
green=*n*-valent
		

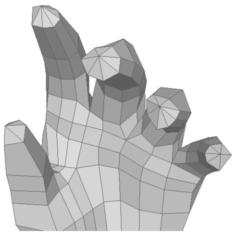
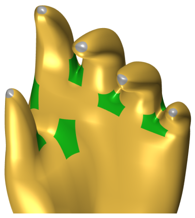
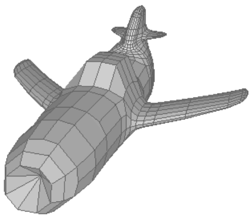
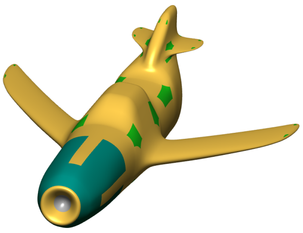
Hand silver=polar finger tips		
Airplane Cyan=T1		

Table 2

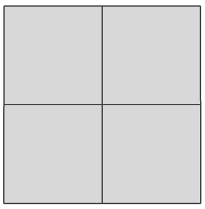
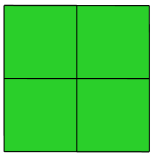
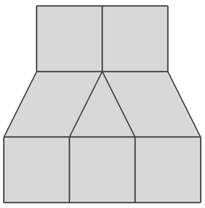
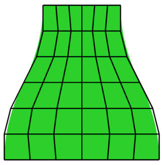
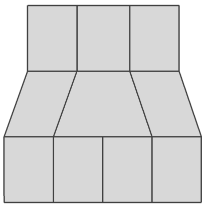
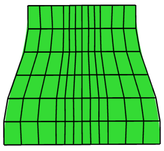
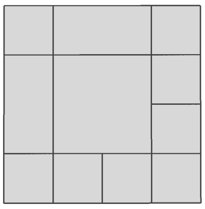
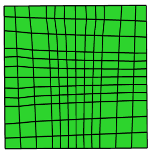
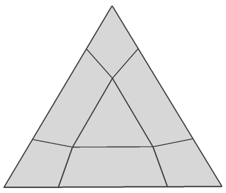
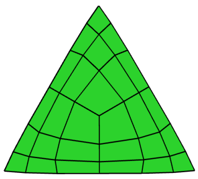
Config. Type	Input (.obj)	Output (.bv)
Regular (plane2x2.obj)		
T0 (T0.obj)		
T1 (T1.obj)		
T2 (T2.obj)		
3-gon (ngon3.obj)		

Table 3

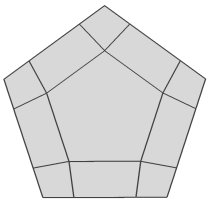
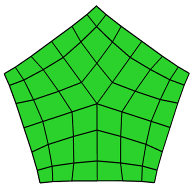
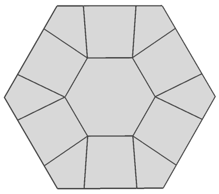
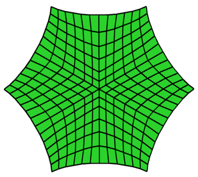
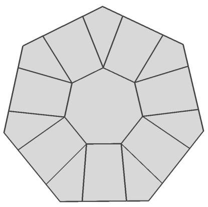
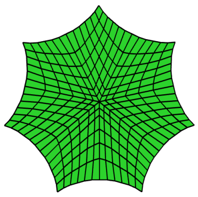
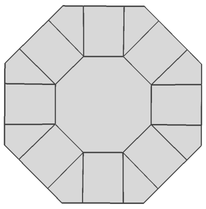
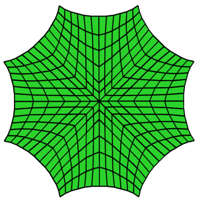
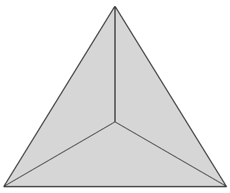
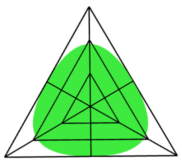
Config. Type	Input (.obj)	Output (.bv)
5-gon (ngon5.obj)		
6-gon (ngon6.obj)		
7-gon (ngon7.obj)		
8-gon (ngon8.obj)		
Polar n=3 (polar3sct.obj)		

Table 4

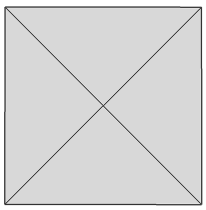
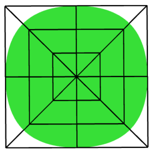
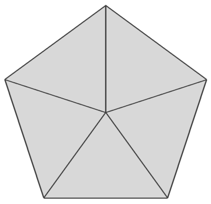
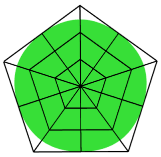
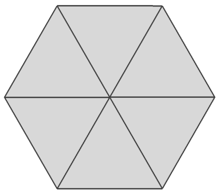
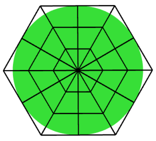
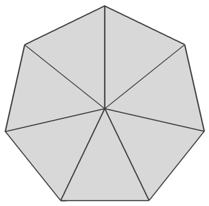
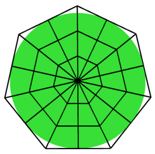
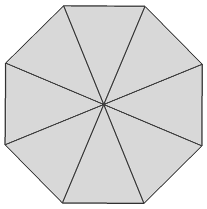
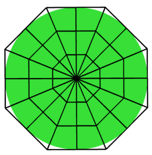
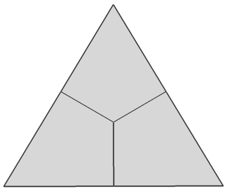
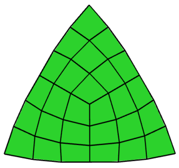
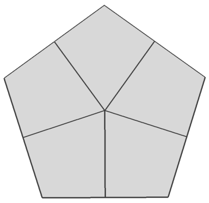
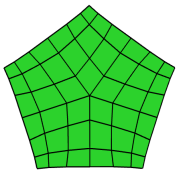
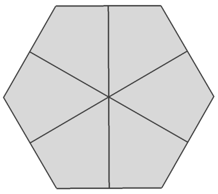
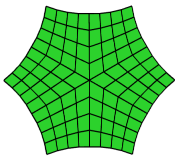
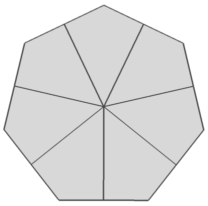
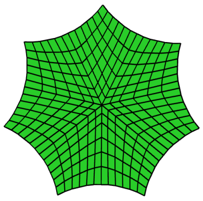
Config. Type	Input (.obj)	Output (.bv)
Polar n=4 (polar4sct.obj)		
Polar n=5 (polar5sct.obj)		
Polar n=6 (polar6sct.obj)		
Polar n=7 (polar7sct.obj)		
Polar n=8 (polar8sct.obj)		

Table 5

Config. Type	Input (.obj)	Output (.bv)
EOP. n=3 (eop3sct.obj)		
EOP. n=5 (eop5sct.obj)		
EOP. n=6 (eop6sct.obj)		
EOP. n=7 (eop7sct.obj)		
EOP. n=8 (eop8sct.obj)