Implementation: Mesh denoising via L0 minimization

from Lei He (Texas A&M University) and Scott Schaefer (Texas A&M University)

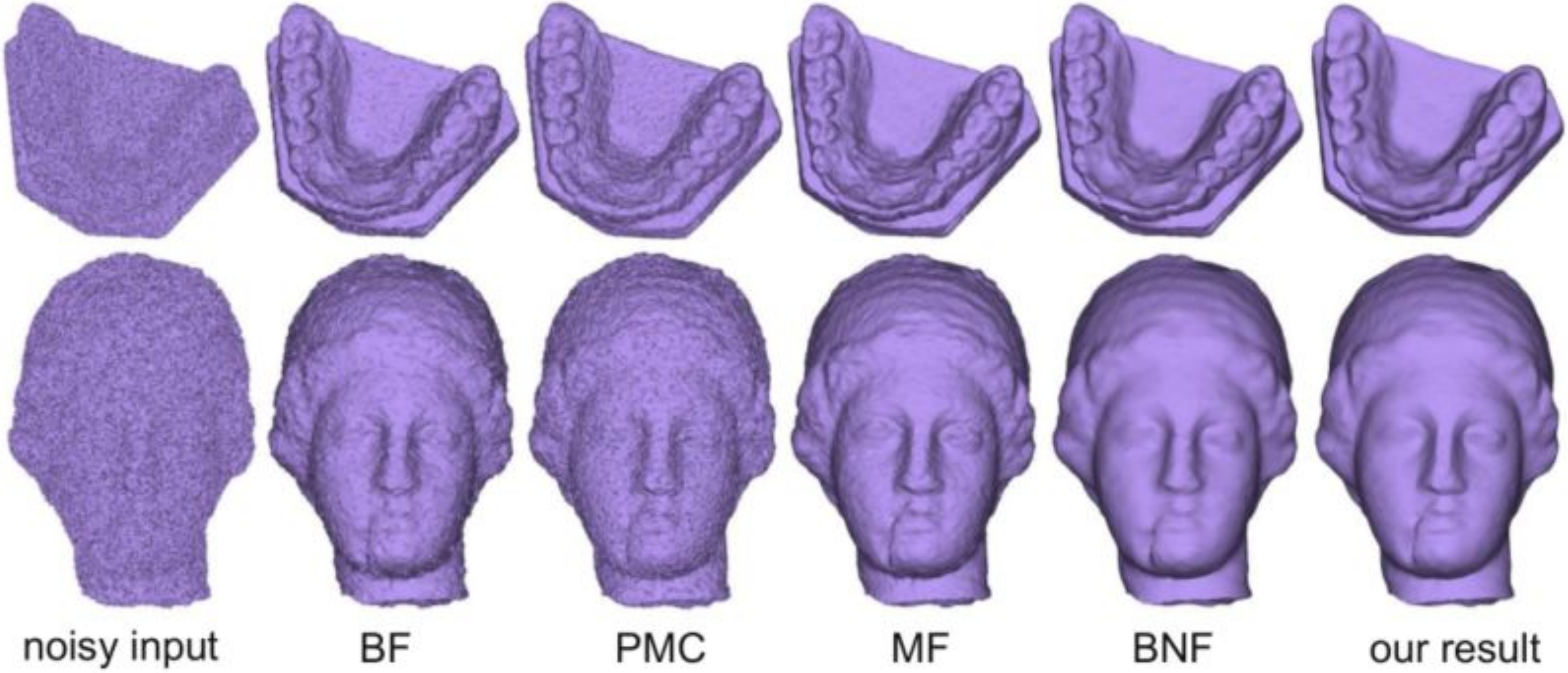
Paul Alonzo Quio Añamuro

paul.quio@ucsp.edu.pe

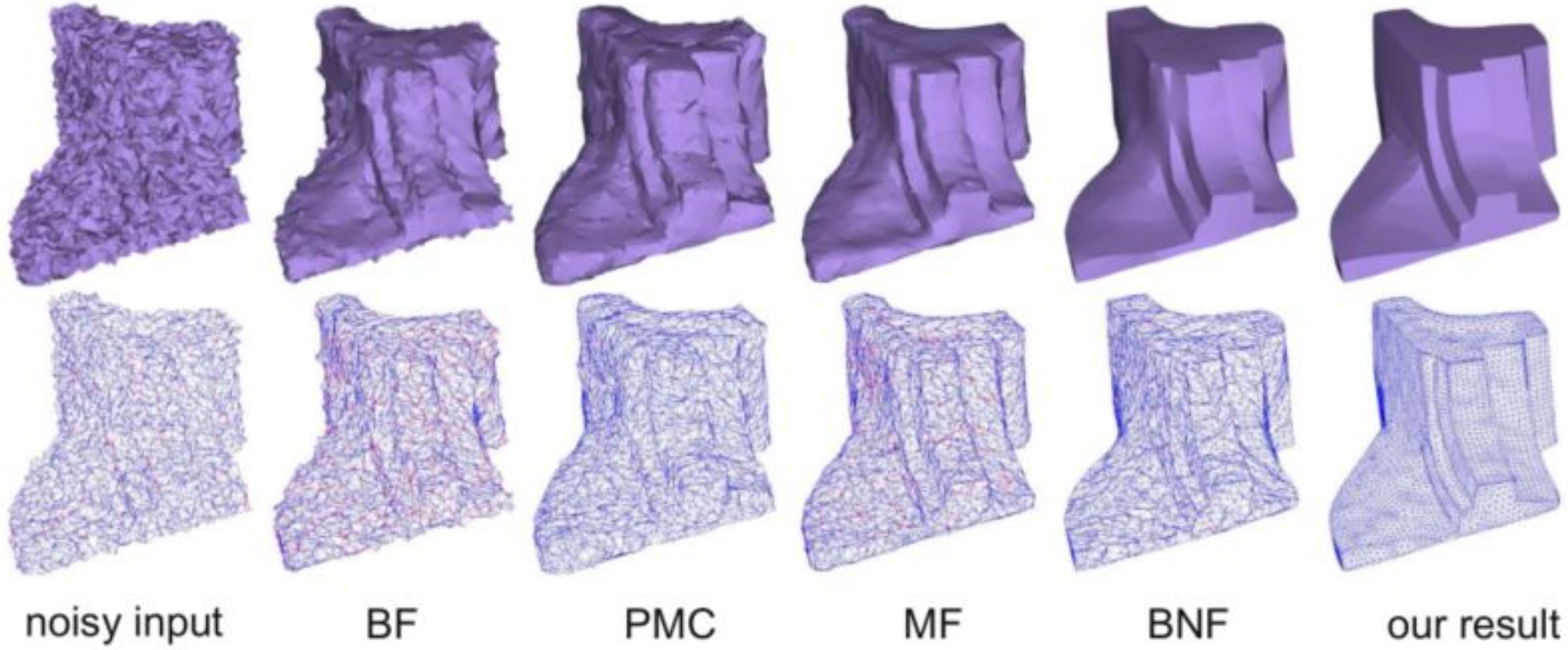
CONCYTEC
CONSEJO NACIONAL DE CIENCIA, TECNOLOGÍA E INNOVACIÓN TECNOLÓGICA

Universidad Católica
San Pablo

Motivation

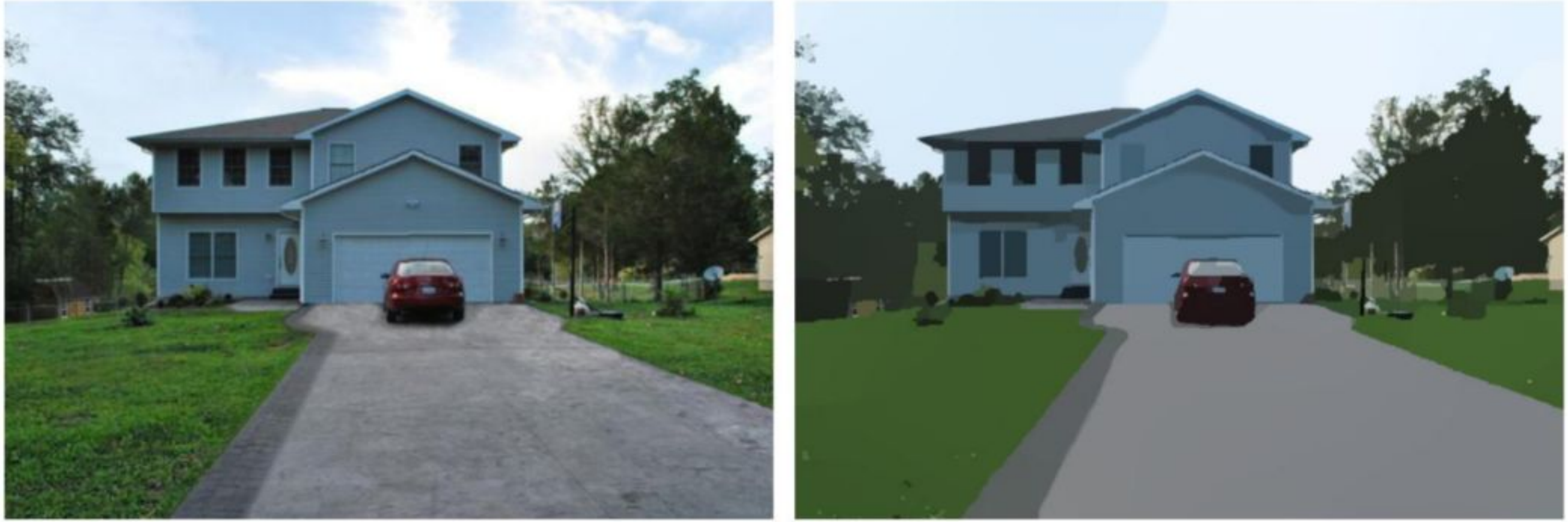


Motivation

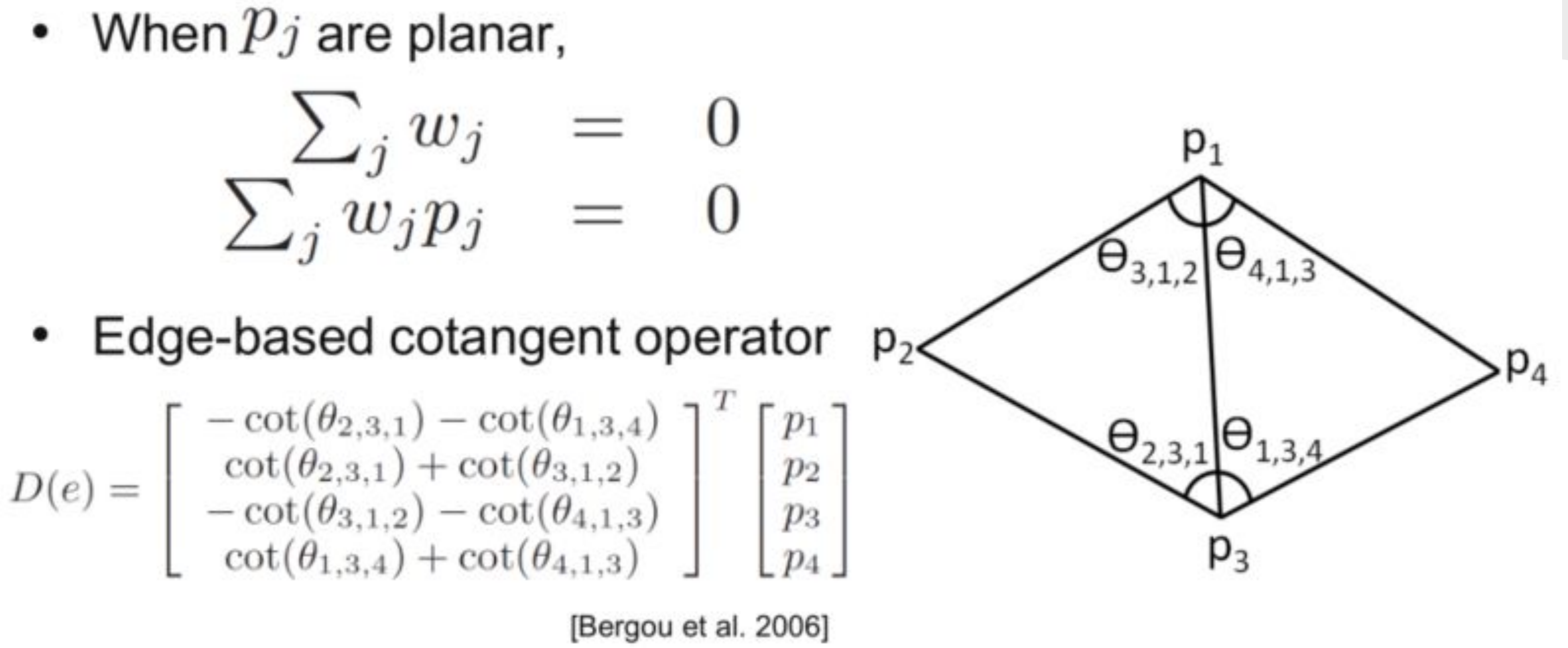


Lo minimization for images

$$\min_{c} |c - c^*|^2 + \lambda |\nabla c|_0$$



Edge-based cotangent operator



- When p_j are planar,

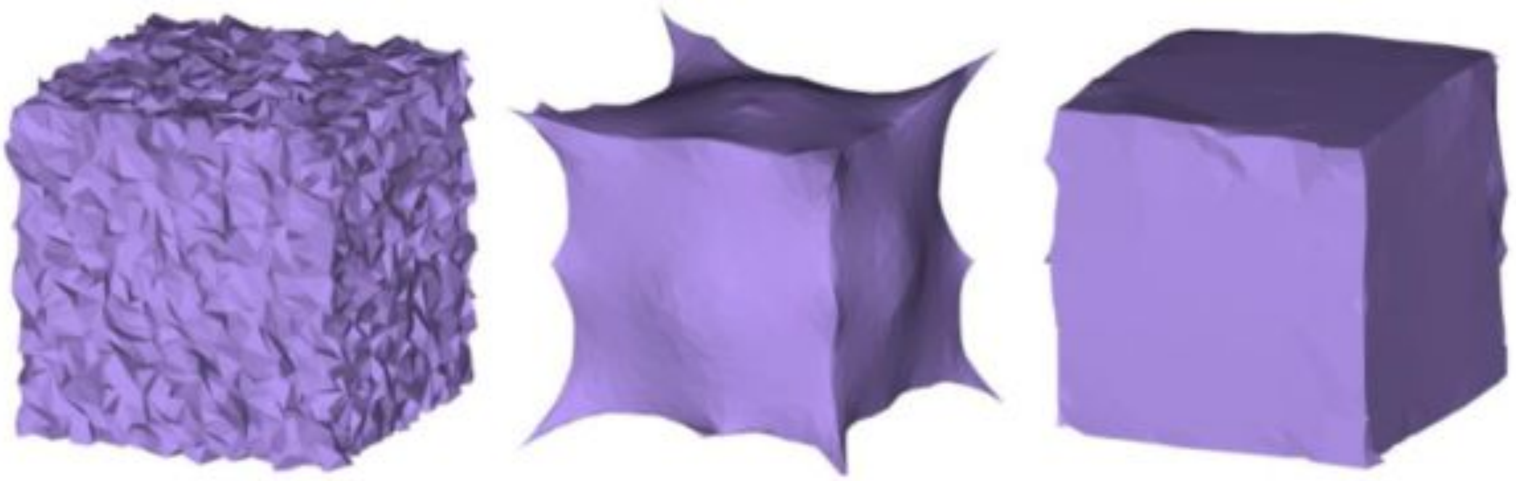
$$\begin{aligned} \textstyle\sum_j w_j &= 0 \\ \textstyle\sum_j w_j p_j &= 0 \end{aligned}$$

- Edge-based cotangent operator

$$D(e) = \begin{bmatrix} -\cot(\theta_{2,3,1}) - \cot(\theta_{1,3,4}) \\ \cot(\theta_{2,3,1}) + \cot(\theta_{3,1,2}) \\ -\cot(\theta_{3,1,2}) - \cot(\theta_{4,1,3}) \\ \cot(\theta_{1,3,4}) + \cot(\theta_{4,1,3}) \end{bmatrix}^T \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

[Bergou et al. 2006]

Edge-based cotangent operator

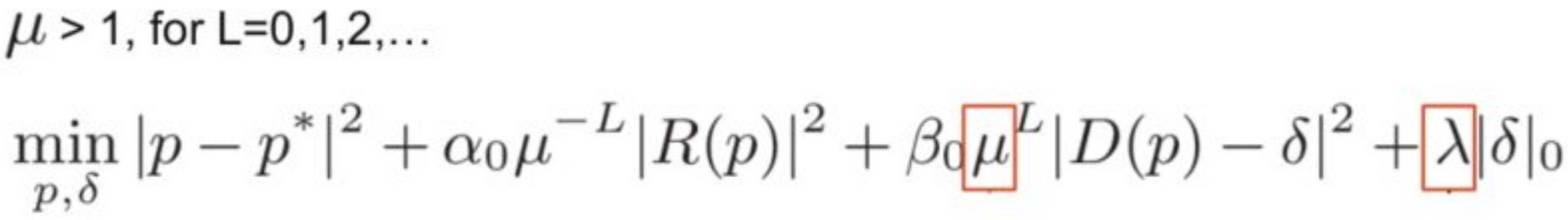


input surface

vertex-based cotangent operator

cotangent edge operator

Final equation

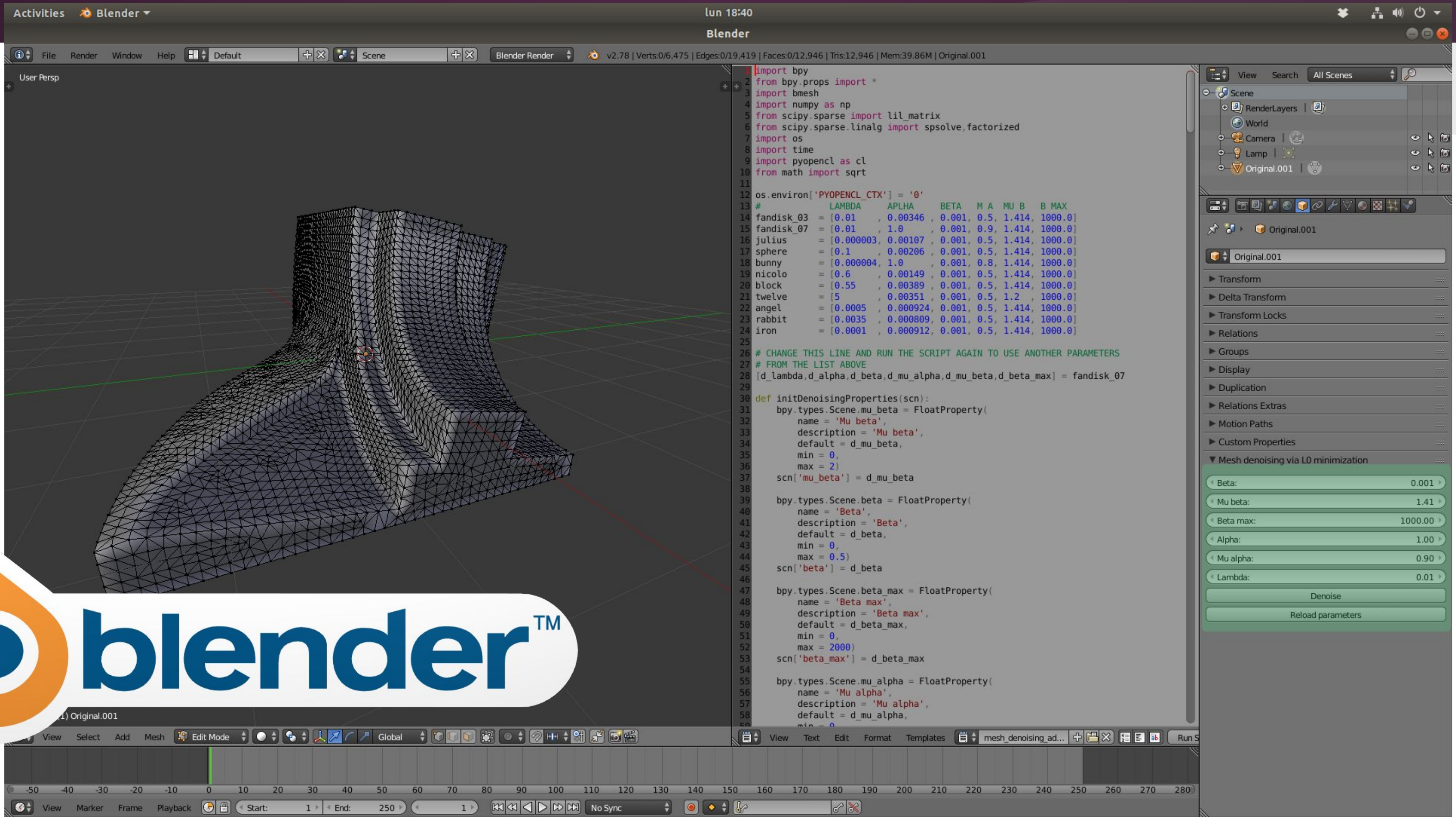


μ > 1, for L=0,1,2,…

$$\min_{p,\delta} |p - p^*|^2 + \alpha_0 \mu^{-L} |R(p)|^2 + \beta_0 \boxed{\mu}^L |D(p) - \delta|^2 + \boxed{\lambda} |\delta|_0$$

Proposal

Addon for Blender



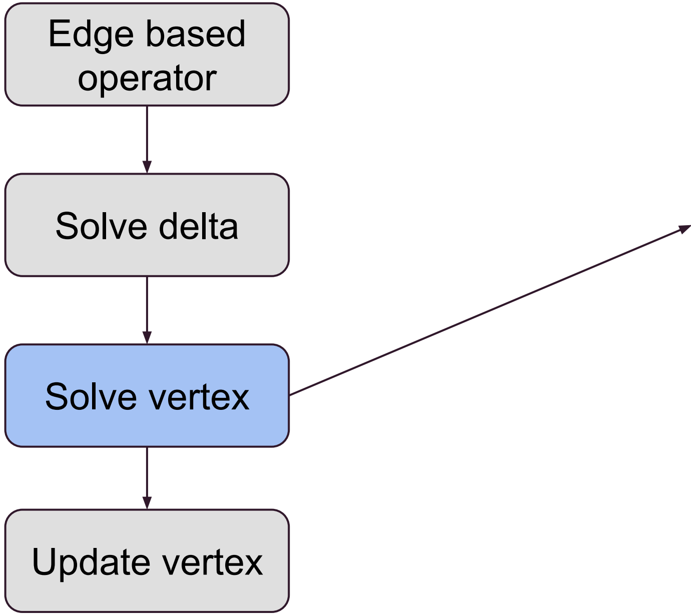
v2.78 | Verts:0/6,475 | Edges:0/19,419 | Faces:0/12,946 | Tris:12,946 | Mem:39.86M | Original.001

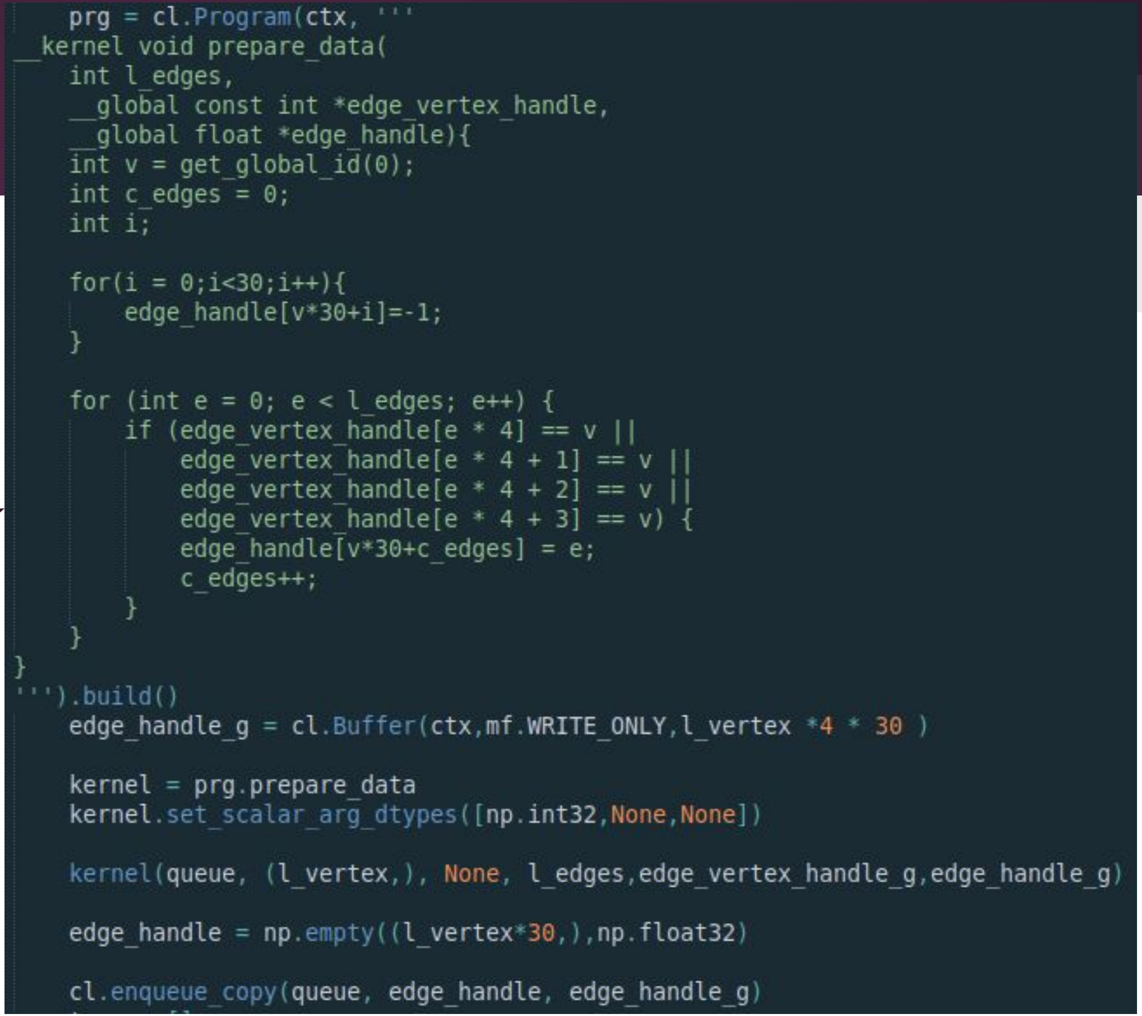
```
import bpy
from bpy.props import *
import bmesh
import numpy as np
from scipy.sparse import lil_matrix
from scipy.sparse.linalg import spsolve,factorized
import os
import time
import pyopencl as cl
from math import sqrt

os.environ['PYOPENCL_CTX'] = '0'
#               LAMBDA      APLHA      BETA   M A  MU B   B MAX
fandisk_03  = [0.01     , 0.00346 , 0.001, 0.5, 1.414, 1000.0]
fandisk_07  = [0.01     , 1.0     , 0.001, 0.9, 1.414, 1000.0]
julius      = [0.000003, 0.00107 , 0.001, 0.5, 1.414, 1000.0]
sphere      = [0.1      , 0.00206 , 0.001, 0.5, 1.414, 1000.0]
bunny       = [0.000004, 1.0     , 0.001, 0.8, 1.414, 1000.0]
nicolo      = [0.6      , 0.00149 , 0.001, 0.5, 1.414, 1000.0]
block       = [0.55     , 0.00389 , 0.001, 0.5, 1.414, 1000.0]
twelve      = [5        , 0.00351 , 0.001, 0.5, 1.2  , 1000.0]
angel       = [0.0005   , 0.000924, 0.001, 0.5, 1.414, 1000.0]
rabbit      = [0.0035   , 0.000809, 0.001, 0.5, 1.414, 1000.0]
iron        = [0.0001   , 0.000912, 0.001, 0.5, 1.414, 1000.0]

# CHANGE THIS LINE AND RUN THE SCRIPT AGAIN TO USE ANOTHER PARAMETERS
# FROM THE LIST ABOVE
[d_lambda,d_alpha,d_beta,d_mu_alpha,d_mu_beta,d_beta_max] = fandisk_07

def initDenoisingProperties(scn):
    bpy.types.Scene.mu_beta = FloatProperty(
        name = 'Mu beta',
        description = 'Mu beta',
        default = d_mu_beta,
        min = 0,
        max = 2)
    scn['mu_beta'] = d_mu_beta

    bpy.types.Scene.beta = FloatProperty(
        name = 'Beta',
        description = 'Beta',
        default = d_beta,
        min = 0,
        max = 0.5)
    scn['beta'] = d_beta

    bpy.types.Scene.beta_max = FloatProperty(
        name = 'Beta max',
        description = 'Beta max',
        default = d_beta_max,
        min = 0,
        max = 2000)
    scn['beta_max'] = d_beta_max

    bpy.types.Scene.mu_alpha = FloatProperty(
        name = 'Mu alpha',
        description = 'Mu alpha',
        default = d_mu_alpha,
```

Beta: 0.001
Mu beta: 1.41
Beta max: 1000.00
Alpha: 1.00
Mu alpha: 0.90
Lambda: 0.01

Start: 1
End: 250
1

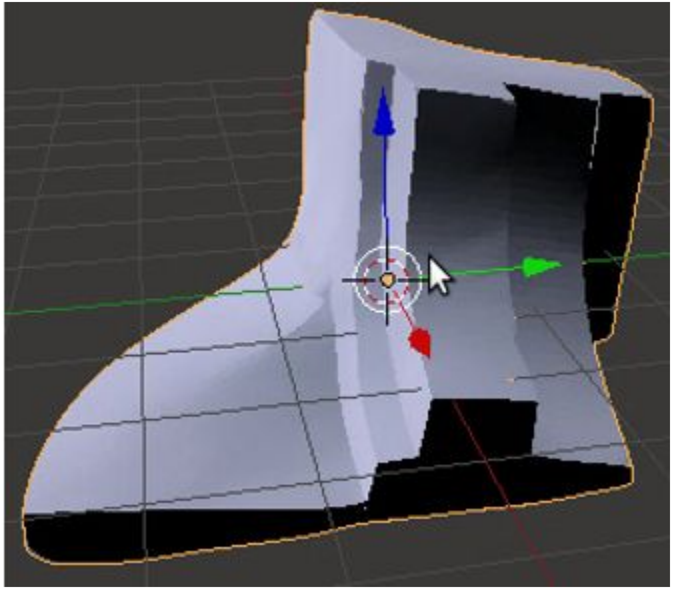
OpenCl



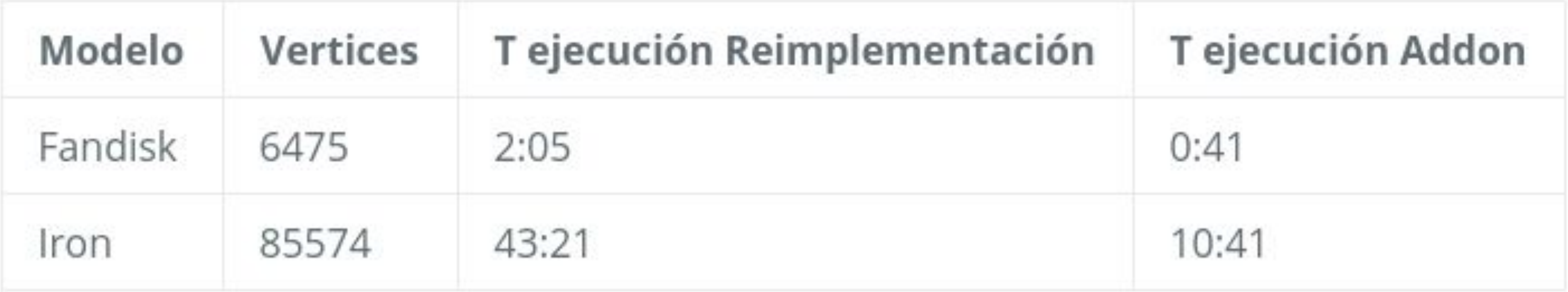


```
    prg = cl.Program(ctx, '''
__kernel void prepare_data(
    int l_edges,
    __global const int *edge_vertex_handle,
    __global float *edge_handle){
    int v = get_global_id(0);
    int c_edges = 0;
    int i;

    for(i = 0;i<30;i++){
        edge_handle[v*30+i]=-1;
    }

    for (int e = 0; e < l_edges; e++) {
        if (edge_vertex_handle[e * 4] == v ||
            edge_vertex_handle[e * 4 + 1] == v ||
            edge_vertex_handle[e * 4 + 2] == v ||
            edge_vertex_handle[e * 4 + 3] == v) {
            edge_handle[v*30+c_edges] = e;
            c_edges++;
        }
    }
}
''').build()
    edge_handle_g = cl.Buffer(ctx,mf.WRITE_ONLY,l_vertex *4 * 30 )

    kernel = prg.prepare_data
    kernel.set_scalar_arg_dtypes([np.int32,None,None])

    kernel(queue, (l_vertex,), None, l_edges,edge_vertex_handle_g,edge_handle_g)

    edge_handle = np.empty((l_vertex*30,),np.float32)

    cl.enqueue_copy(queue, edge_handle, edge_handle_g)
```

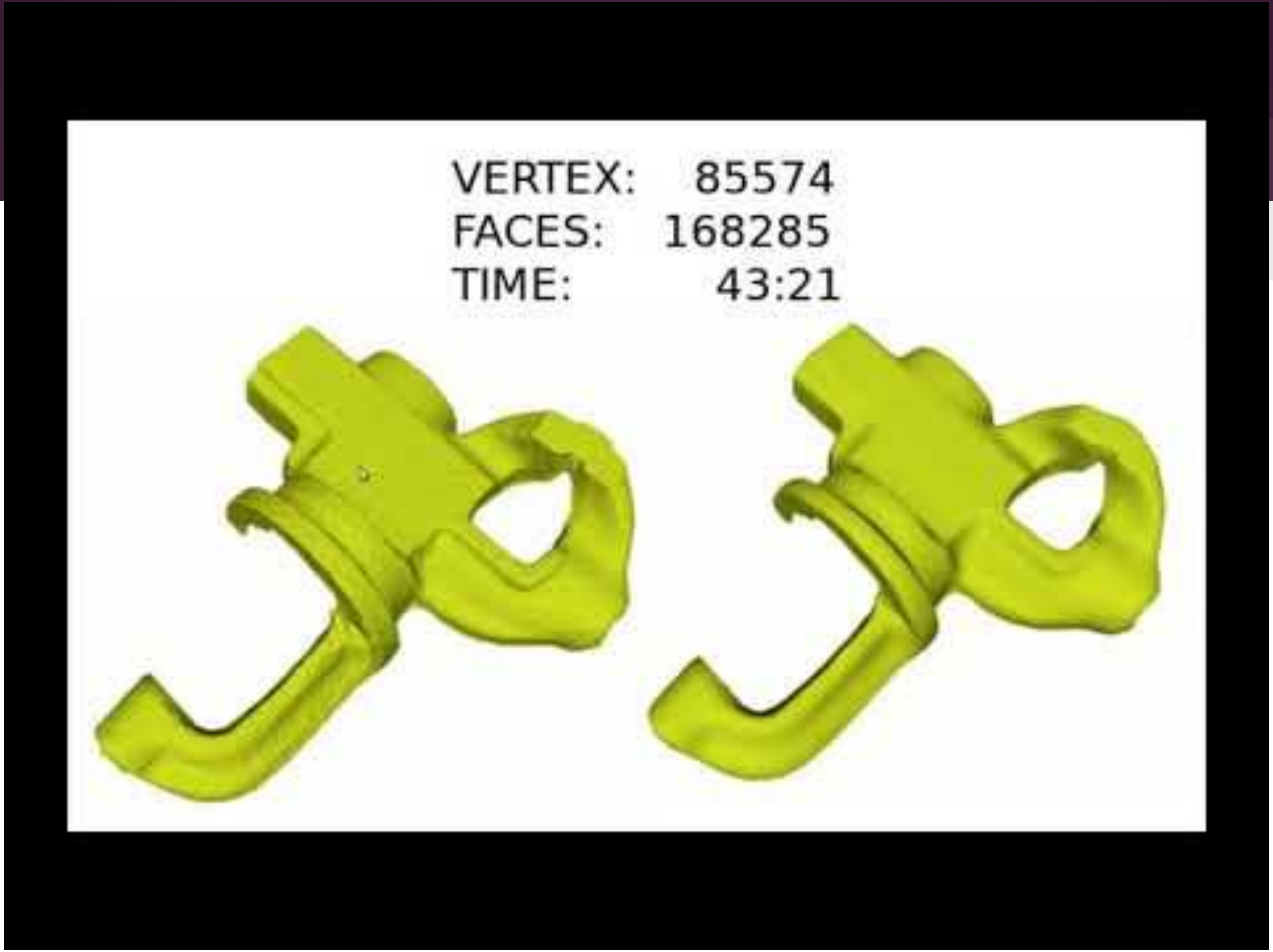
Comparison



Comparison



Modelo	Vertices	T ejecución Reimplementación	T ejecución Addon
Fandisk	6475	2:05	0:41
Iron	85574	43:21	10:41



```
VERTEX:   85574
FACES:   168285
TIME:     43:21
```

Thanks

CONCYTEC
CONSEJO NACIONAL DE CIENCIA,
TECNOLOGÍA E INNOVACIÓN TECNOLÓGICA

Universidad Católica
San Pablo