Dynamic fusion

大学共同利用機関法人 情報・システム研究機構
国立情報学研究所
National Institute of Informatics

Internship Week 14
Tracking Mesh and Fusion
25 May 2017

Inoë ANDRE

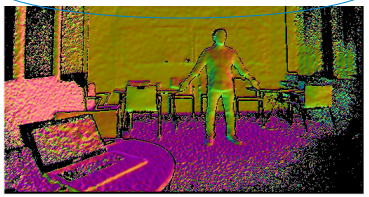
Advisors : Prof. A.Sugimoto
Dr. D.Thomas

Last meeting

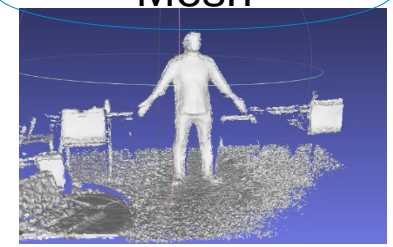
- Previously
  - Pipeline
  - Compute Marching Cubes Normals
  - Mesh Tracking
- Plan for today's meeting:
  - Tracking with Mesh
  - Fusion
  - Segmented Fusion

Mesh Tracking

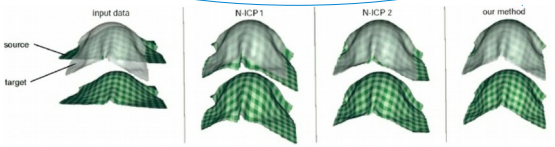
Vtx + Nmls



Mesh



rigid Alignment



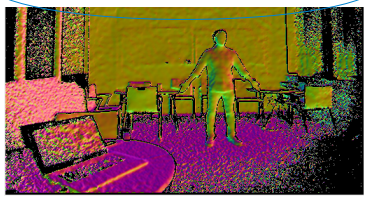
New Pose Estimation

Pipeline

Tracking

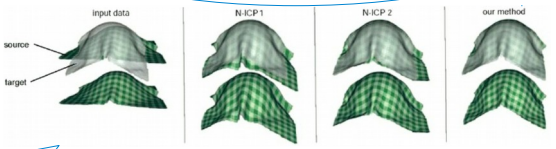
1

Vtx + Nmls + BBox



2

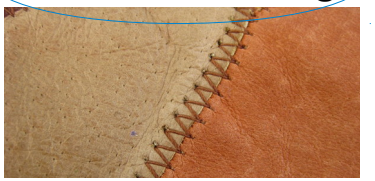
Rigid Alignment



stitching

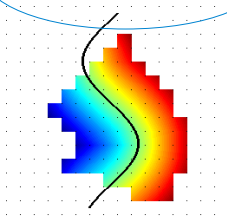
TSDF Stitching

3



4

TSDF Fusion



5

Mesh

6

Warp Field

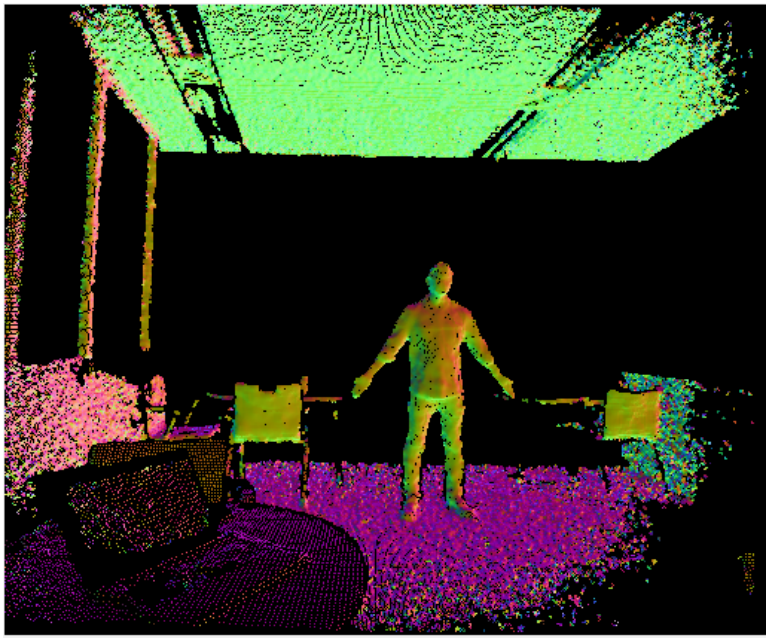
(e) Canonical model warped into its live frame

Progress

- Tracking with the Mesh
  1) Transform Mesh with current Pose
  2) Projection Vtx of Mesh in current depth frame
  3) Compare Vtx
  4) Compare Nmls
  5) Add correspondence to matrix
  6) ICP

Progress

- Code versionning management
  - Fusion of bad code : corrected
- Merge Vertex
  - GPU : not enough memory
    (even when type = short int)
  - CPU : list Vtx and Nmls
    differents size
    => cannot do the correspondence
    at the beginning



2D rendering from MC's Vtx

Progress

- Mesh tracking

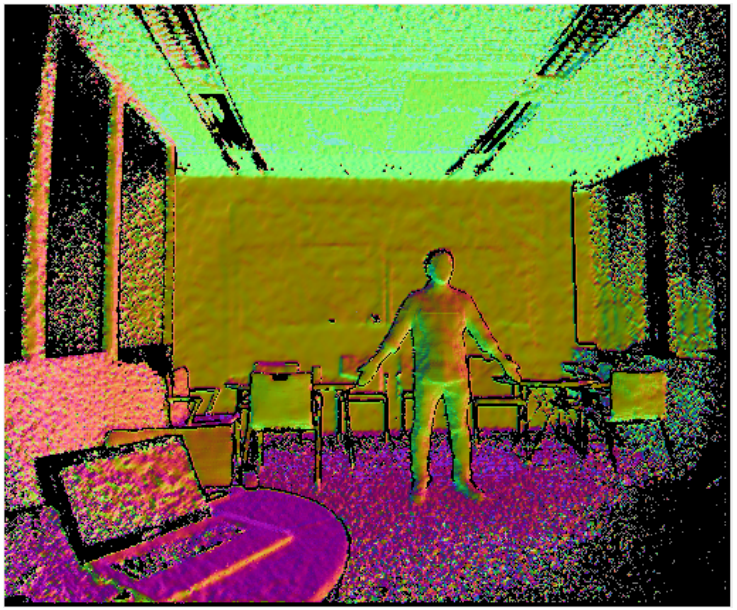
Input: 3D Mesh = {Vtx, Nmles (list)}
RGBD image {Vtx_in and Nmles_in images(424*512)}
current Pose & intrinsic.
Output : new Pose

```
Res = Pose
For iter in range(max_iter):
    For i in range(Vtx.size):
        v_curr = Vex[i]
        n_curr = Nmles[I]
        v_curr = Pose*v_curr
        N_curr = Rot(Pose)*n_curr

        pix = Intrinsic*v_curr
        if (pix in [424,512]):
            v_match = Vtx_in[pix]
            n_match = Nmles_in[pix]

            if (dist(n_curr, n_match) < thresh_n && dist(v_curr, v_match) < thresh_v):
                # Compute jacobian that corresponds to n_curr(v_curr-v_match)
        ...
        # resolve equation to get Res update
return Res
```

Progress

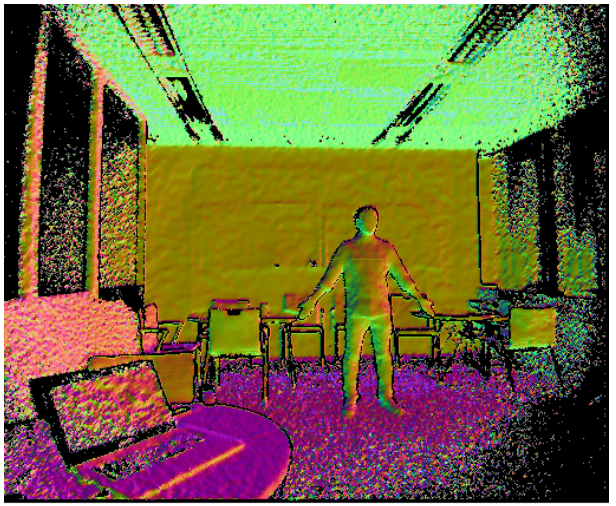
- Mesh tracking

Align Mesh and current depth Image



Overlay current Depth Map with former Mesh (use of transform)

Align Mesh with its own depth Map



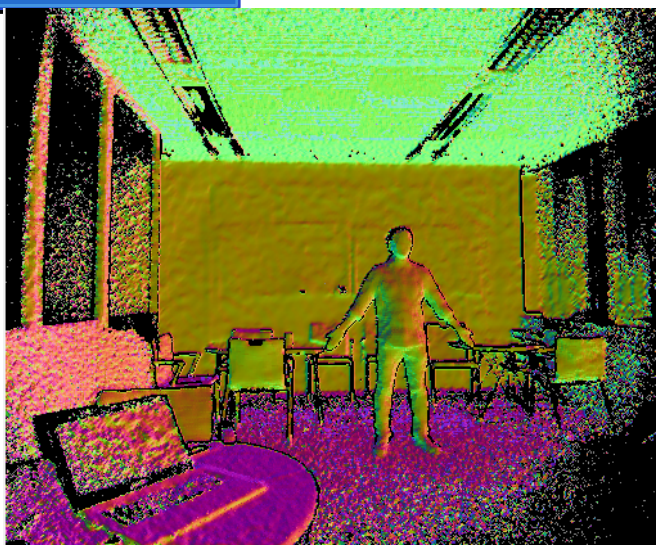
Overlay Depth Map with its Mesh (project in 2D space)

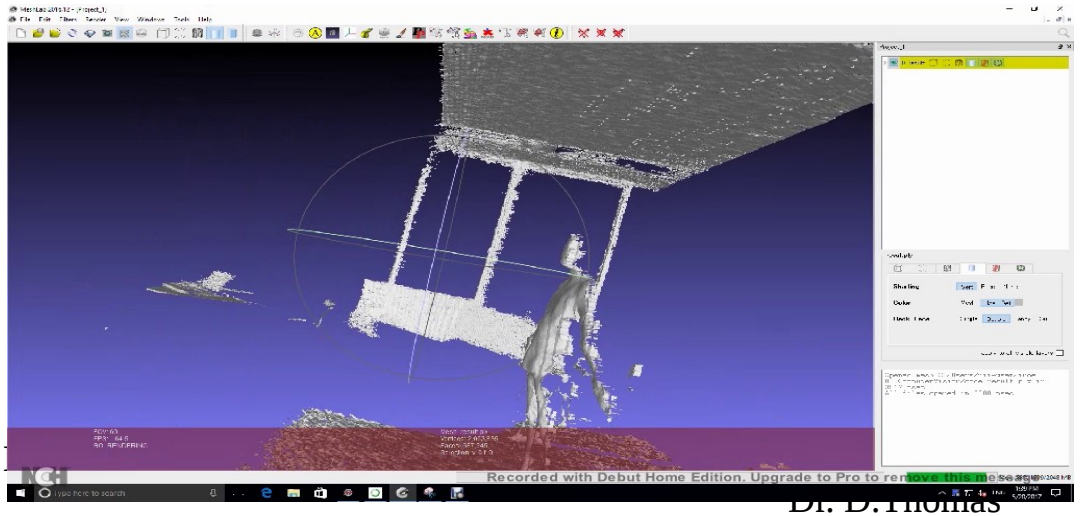
self.Pose

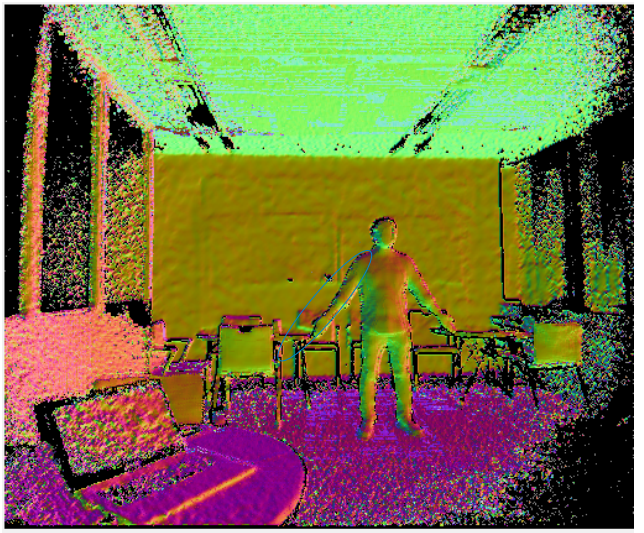
```
[[ 1.00000000e+00 -1.00410589e-05 -5.77200990e-05  2.79055297e-04]
 [ 1.00391580e-05  1.00000000e+00 -3.29403410e-05  2.35130239e-04]
 [ 5.77204264e-05  3.29397626e-05  1.00000000e+00 -2.30166825e-05]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

Progress

- Mesh Tracking
  - Non optimize : point by point correspondence
  - Fusion 3 images: 1h30

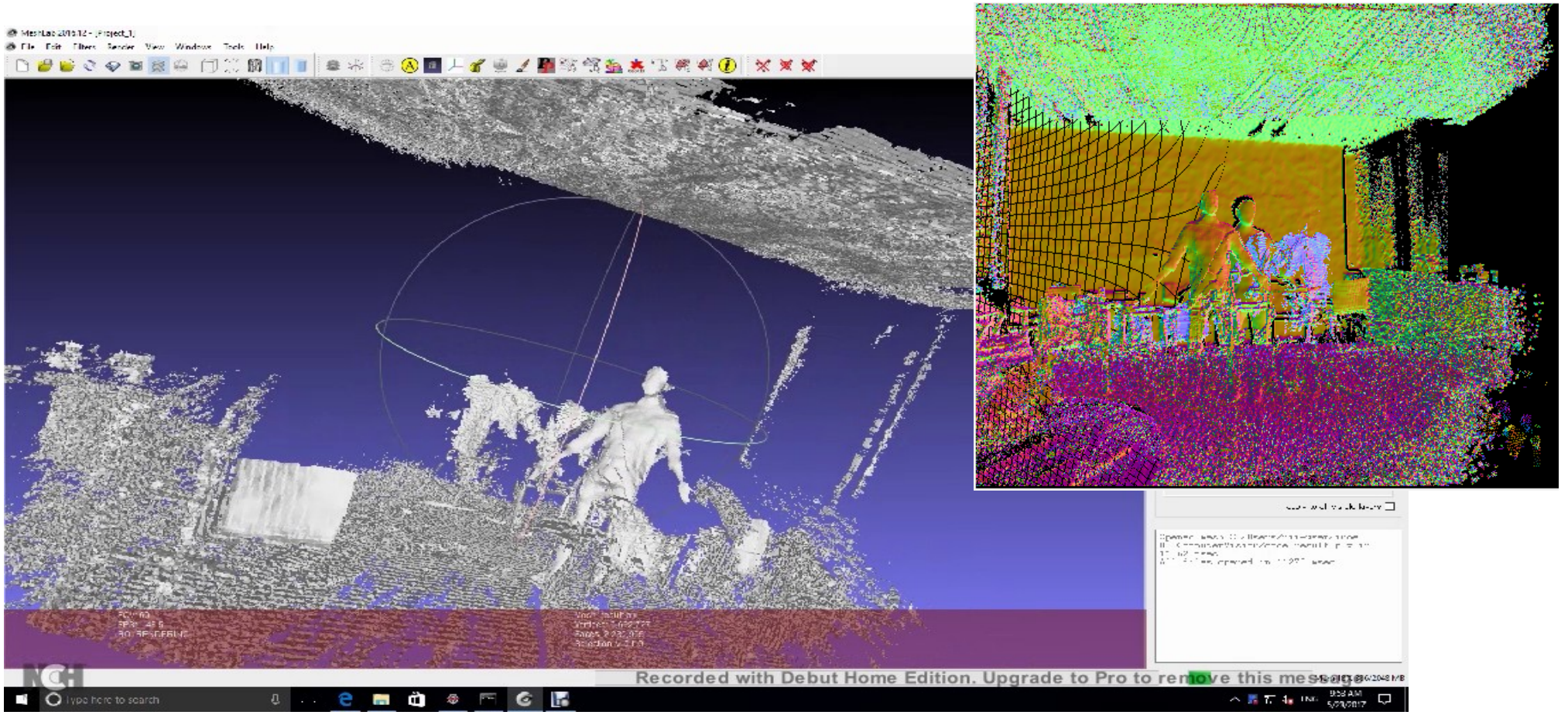






Progress

- Fusion 15 images



Progress

- Mesh tracking: First try by reshaping list

```
Input:    3D Mesh = {Vtx, Nmles (list)}
          RGBD image {Vtx_in and Nmles_in images(424*512)}
          current Pose & intrinsic.
Output : new Pose


Res = Pose
For iter in range(max_iter):
      For i in range(Vtx_in.size[0]):
          For j in range(Vtx_in.size[1])
              v_curr = Vex_in[i,j]
              n_curr = Nmles_in[i,j]
              v_curr = Pose*v_curr
              N_curr = Rot(Pose)*n_curr


              pix = Intrinsic*v_curr
              Reshape Vtx & Nmls in [424,512]
              if (pix in [424,512]):
                    v_match = Vtx[pix]
                    n_match = Nmles[pix]


                    if (dist(n_curr, n_match) < thresh_n && dist(v_curr, v_match) < thresh_v):
                          # Compute jacobian that corresponds to n_curr(v_curr-v_match)
          ...
          # resolve equation to get Res update
return Res
```

Reshaping idea:
Reduce number of element
with panda or Sparse matrix
Indexes in shape of Vtx_in

Progress

- Mesh tracking: In tracking
- Buffer[Indexes_ref[:][:]] = np.dstack((w*mask[:,:]*nmle[:, :,0], \

  w*mask[:,:]*nmle[:, :,1], \

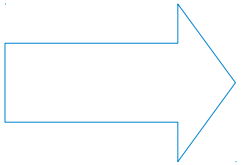
  w*mask[:,:]*nmle[:, :,2], \

  w*mask[:,:]*(-Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,2]*nmle[:,:,1] + Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,1]*nmle[:,:,2]), \

  w*mask[:,:]*(Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,2]*nmle[:,:,0] - Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,0]*nmle[:,:,2]), \

  w*mask[:,:]*(-Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,1]*nmle[:,:,0] + Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,0]*nmle[:,:,1])))

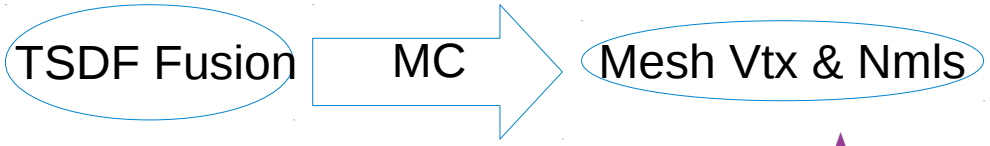
Image2.Vtx → pt
nmls → NewImage.Nmls



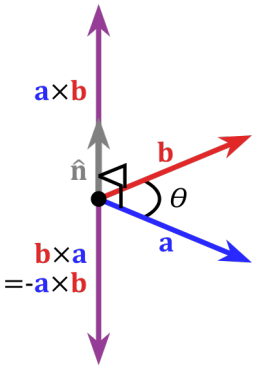
Determinant null

Progress

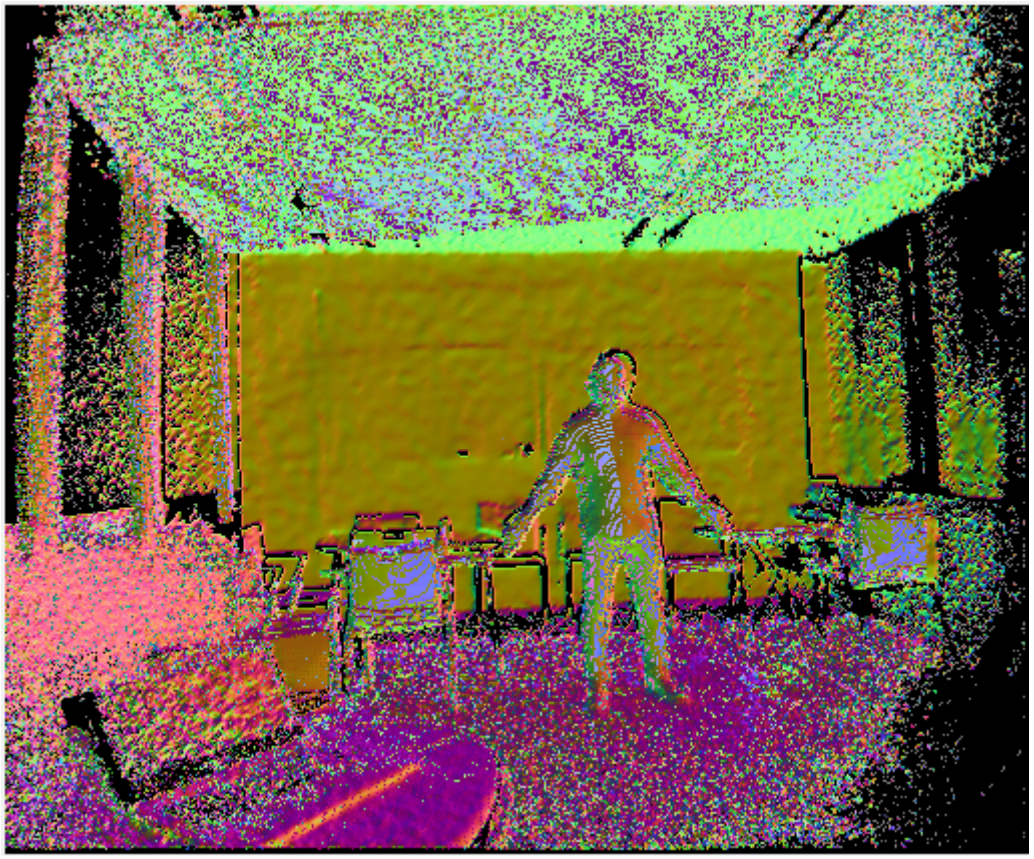
- Mesh tracking: Change Normals orientation



When computing normals from Mesh:
Input : Mesh Vtx, Mesh Faces
Output: Mesh Normals

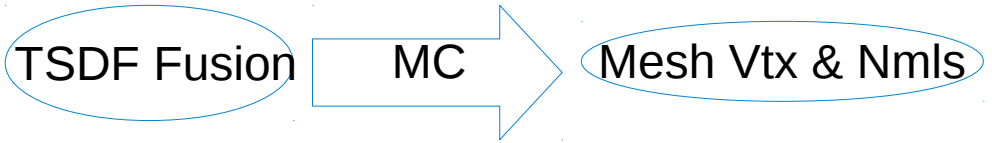


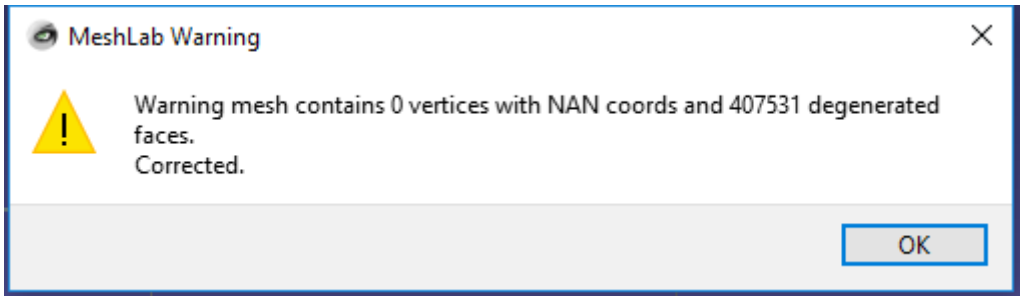
Algo: run through faces{
a ,b two vectors as edges of faces
If angle between a & b > 0 : n= a^b
Else n = b^a
Then put n in vtx of faces
}
Normalize every n.



Progress

- Merge Vertices

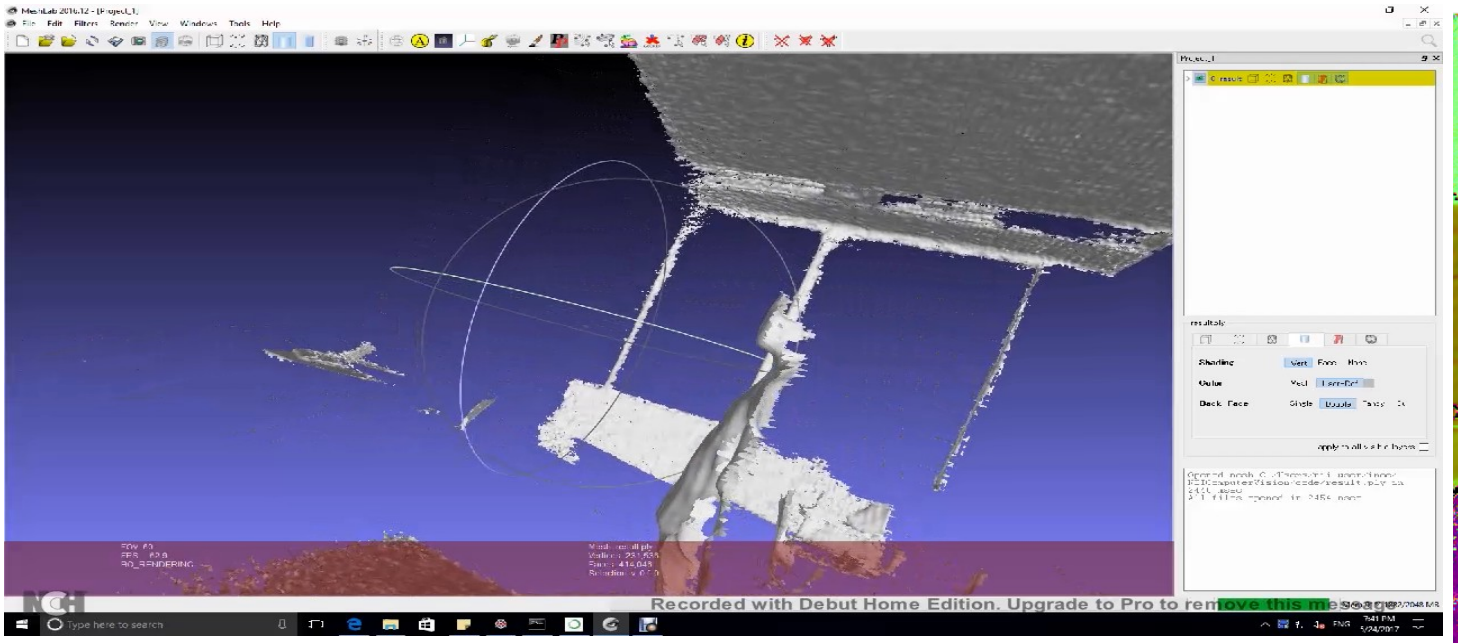


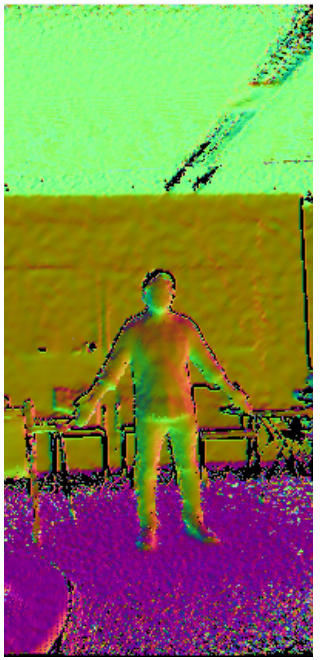


Progress

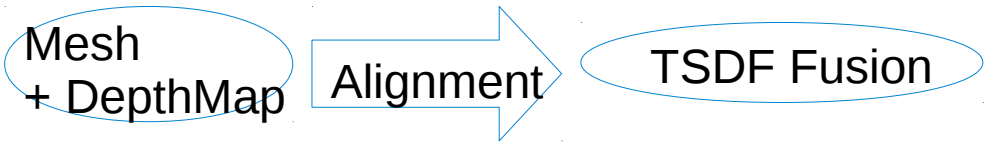
Compute index from list Vtx and Nmls
Transform in TSDF = Pose
MergeVtx()

- Mesh tracking: Pose instead of Inv(Pose) in TSDF





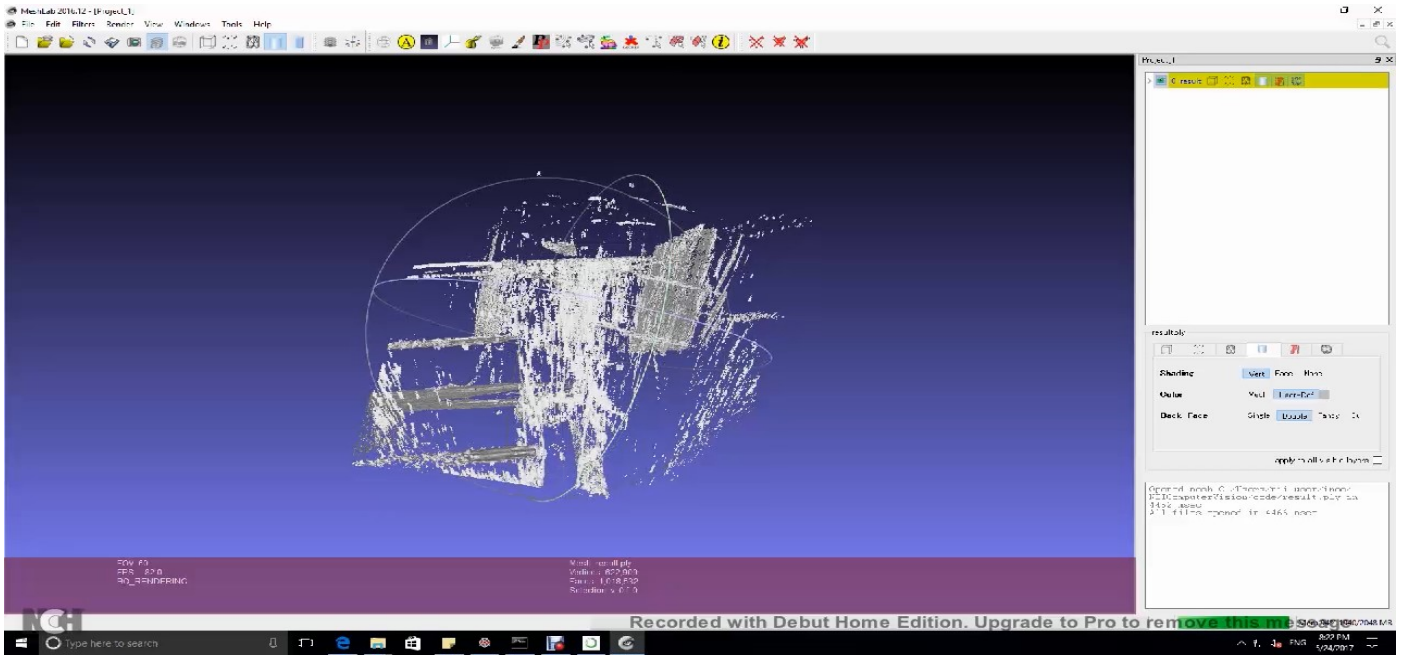
Fusion from 0 to 4 images (600sec)

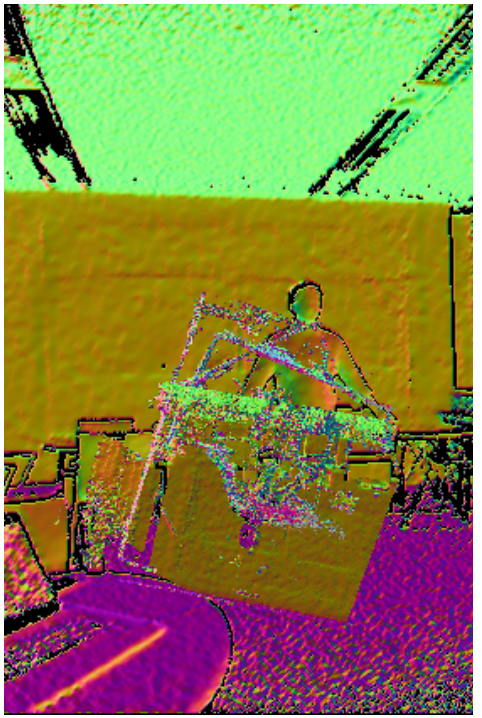


Progress

Compute index from list Vtx and Nmls
Transform in TSDF = Pose
MergeVtx()

- Mesh tracking: Pose instead of Inv(Pose) in TSDF



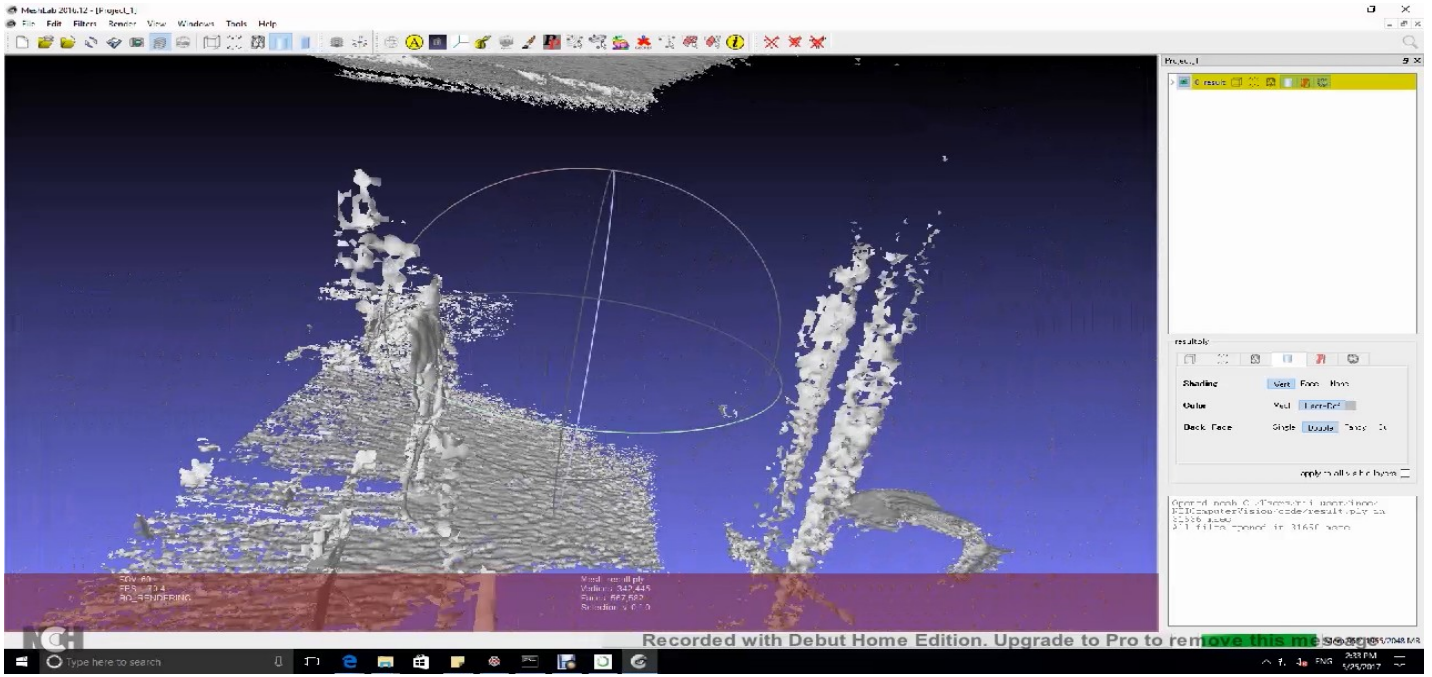


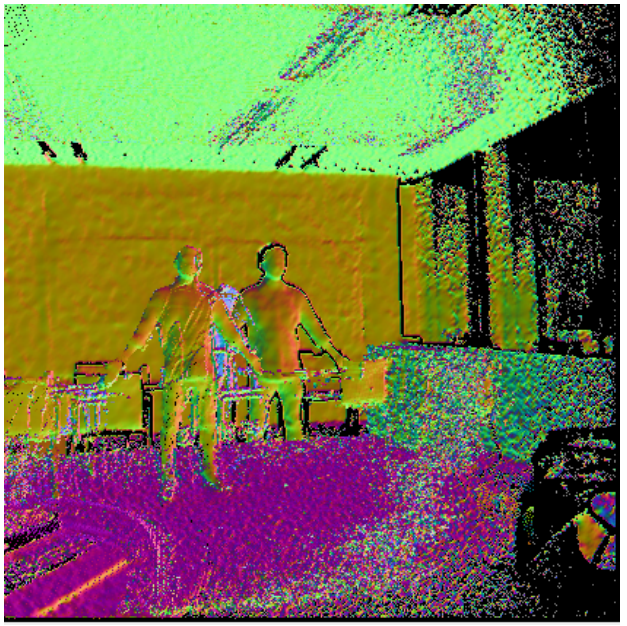
Tracking Model to Frame 15 images fusion

Progress

Compute ind from list Vtx and Nmls
Transform in TSDF = Pose
MergeVtx()

- Mesh tracking: Pose instead of Inv(Pose) in TSDF





Fusion from 9 to 14 images
This looks good because 5 first images are the same!

Progress

Compute ind from list Vtx and Nmls
Transform in TSDF = Pose
MergeVtx()

- Mesh tracking: Pose instead of Inv(Transfo) in TSDF

  - Transfo = Inv(Pose)

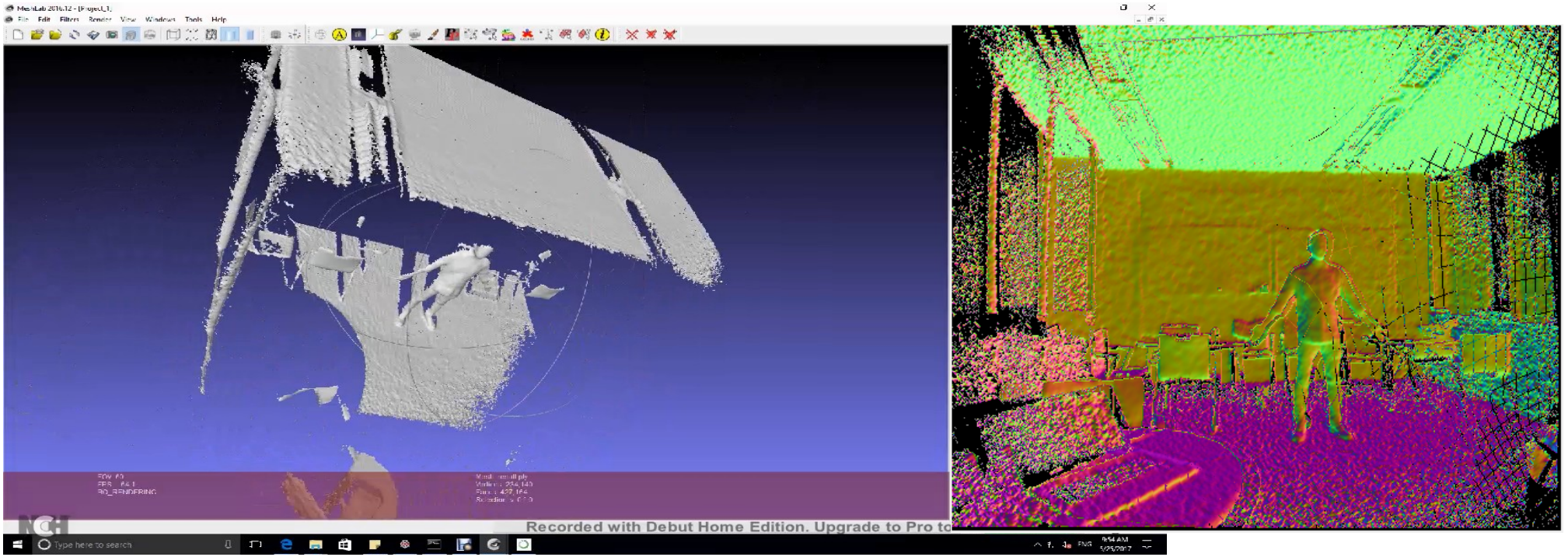
Transfo[0:3,0:3] = Inv(Pose[0:3,0:3])
Transfo[:,2] = -Pose[:,2])

  - Incremental Pose instead of Pose of all the image

Make sure to have the good transformations : exactly same result

Progress

- Mesh tracking: Pose in TSDF



Tracking Frame to Frame 15 images fusion

Why?

Progress

- Reading research papers : Mesh Fusion
  - Cut and paste fusion
  - Smooth fusion : Hermite interpolation
  - Seamless and natural fusion
  - Small topological genus restriction

Action plan

- Debug Fusion
- Then Fusion for each segmented body part separately:
  - Coordinates change one by one
  - Fuse one by one