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

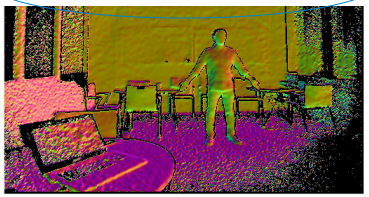
Internship Week 15
Tracking Mesh and Fusion (III)
31 May 2017

Last meeting

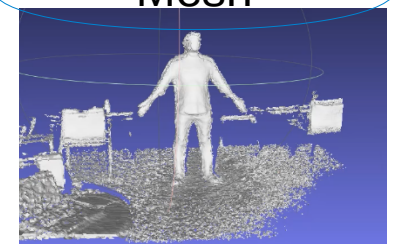
- Previously
  - Correcting Normals
  - Mesh Tracking
  - Fusion : Pose, inverse, normals
- Plan for today's meeting:
  - Identify source
  - Tracking with Mesh
  - Fusion

Mesh Tracking

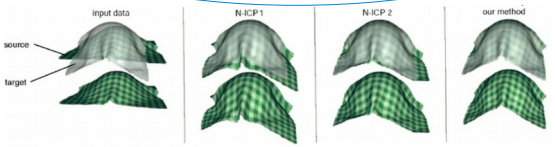
Vtx + Nmls



Mesh



rigid Alignment



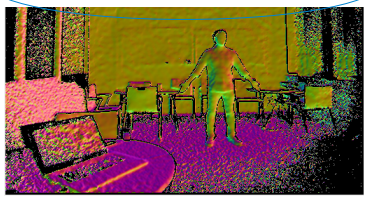
New Pose Estimation

Pipeline

Tracking

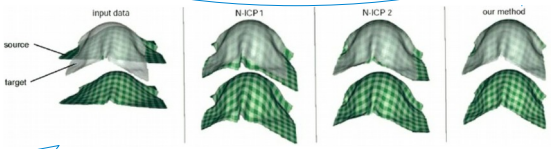
1

Vtx + Nmls + BBox



2

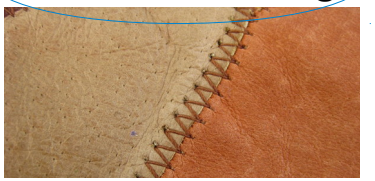
Rigid Alignment



stitching

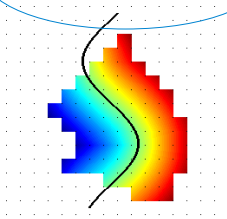
TSDF Stitching

3



4

TSDF Fusion



5

Mesh

6

Warp Field

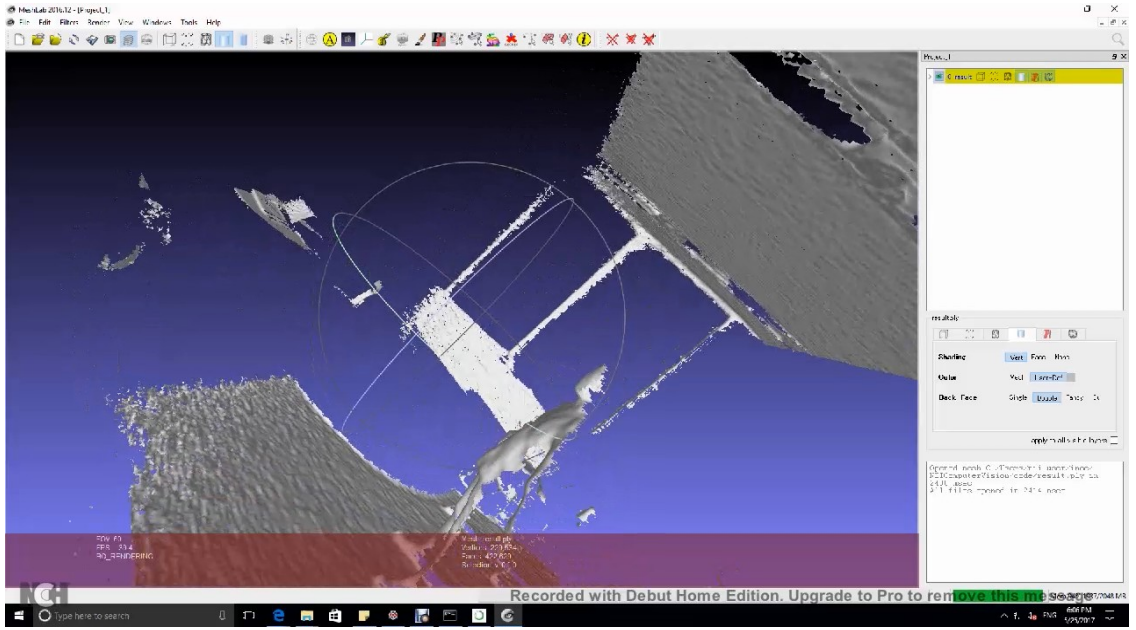
(e) Canonical model warped into its live frame

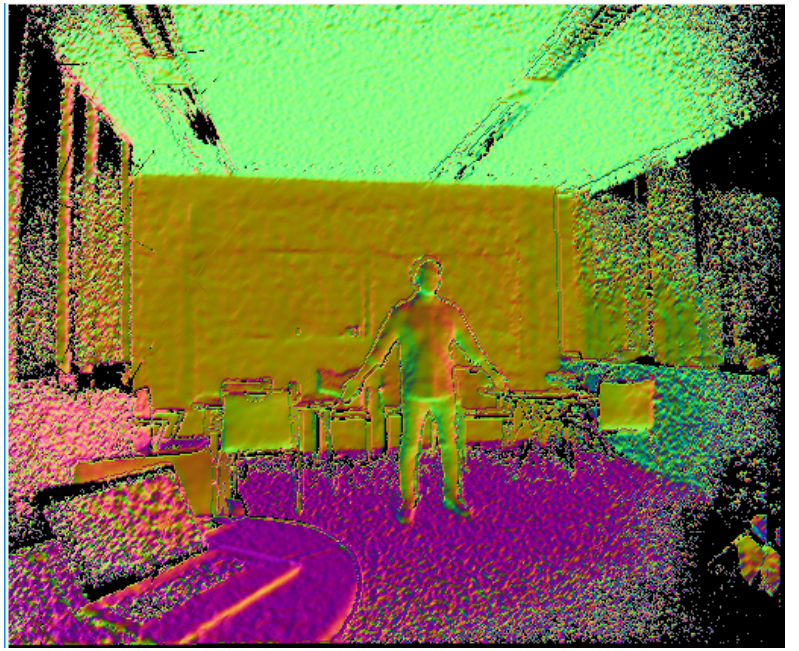
Progress

- 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

- Correct Inverse transform
  - Inv[0:3,0:3] = Pose[0:3,0:3]^−1
  - Inv[:,3] = -Inv[0:3,0:3].Pose[:,3]





Progress

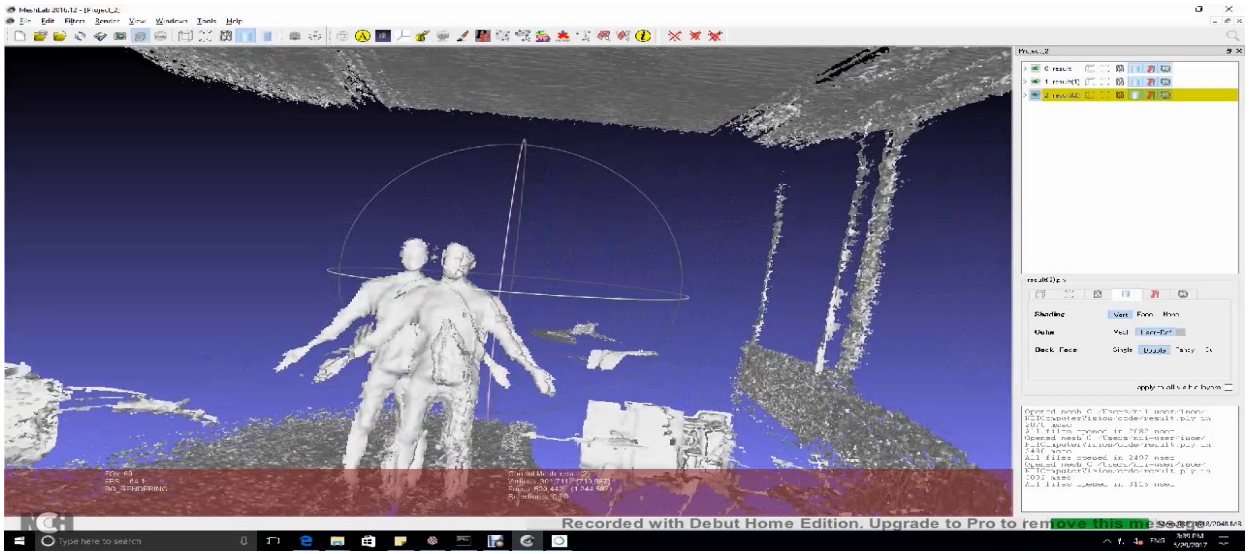
- Identify the source of the problem:
  - In frame to frame fusion:
    - Nb of vertices constantly rise : 220508 to 229534
    - Residual : tend to 0
  - In mesh to frame fusion
    - Nb of vertices decrease
    - Residual : do not tend to 0
  - Mesh tracking without fusion : error
  - Test with known transfo
    - Mesh tracking : Give another matrix
    - Frame to Frame tracking : Give almost same matrix!
- Conclusion : pb in Mesh Tracking

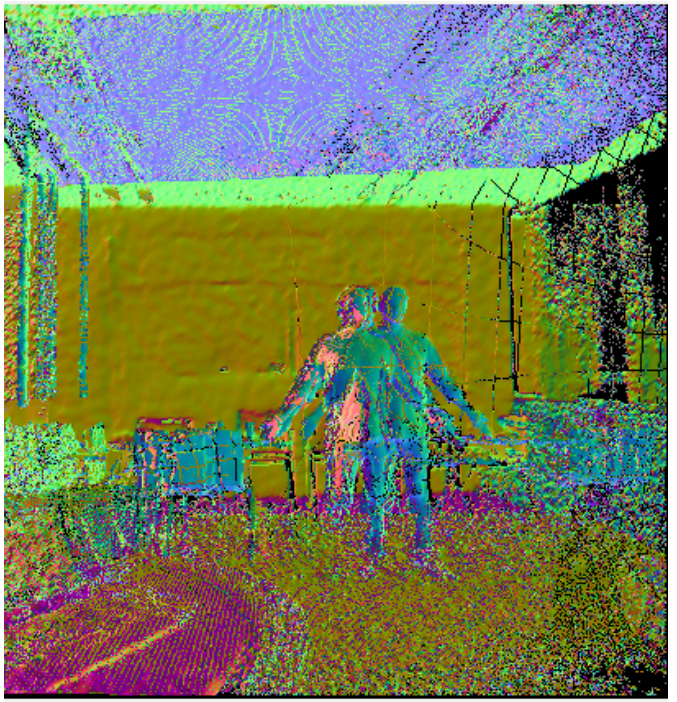
Progress

- Identify the source of the problem in mesh tracking
  - In frame to frame fusion:
    - Transform list in mat (M) with draw function
    - Tracking used in Frame to frame fusion
    - Permutation normal indices
    - Change delta transfo by its inverse
    - Compare M with its initial form => no correspondence

Progress

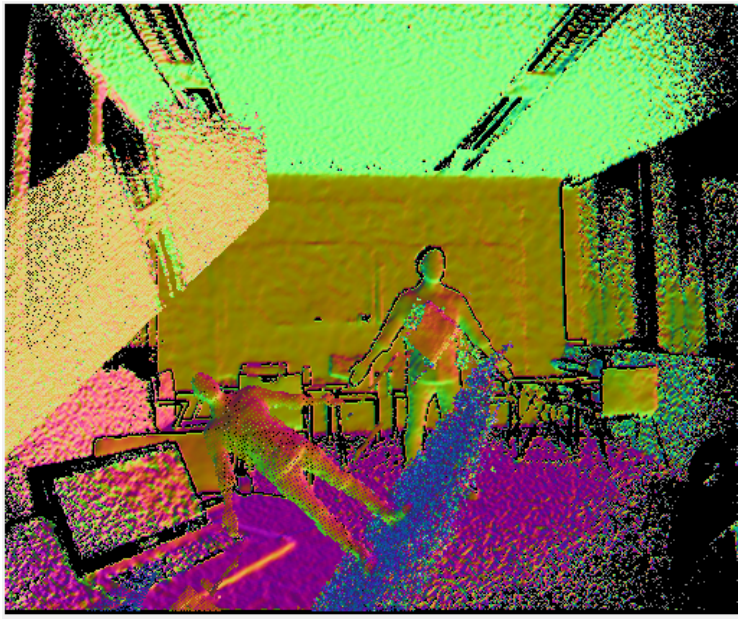
- Using frame to frame fusion:
  - Permutation normals indices





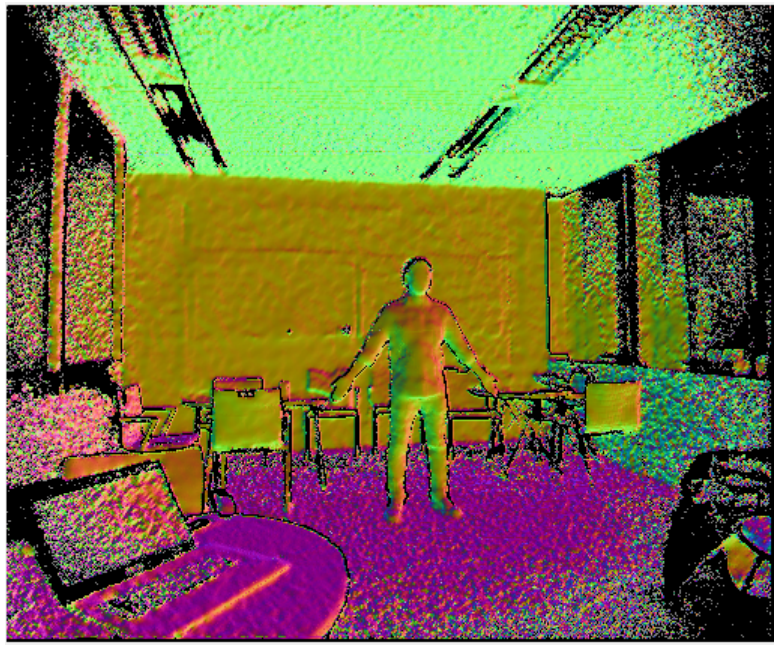
Progress

- Using in frame to frame fusion:
  - Change delta transfo by its inverse

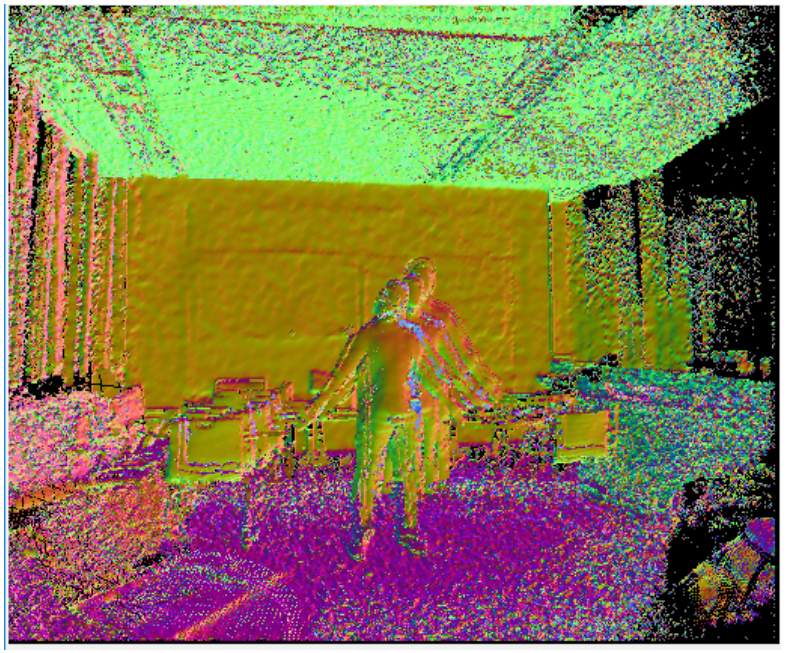


Progress

- Using frame to frame fusion:
  - Transform list in mat with draw function



Fusion

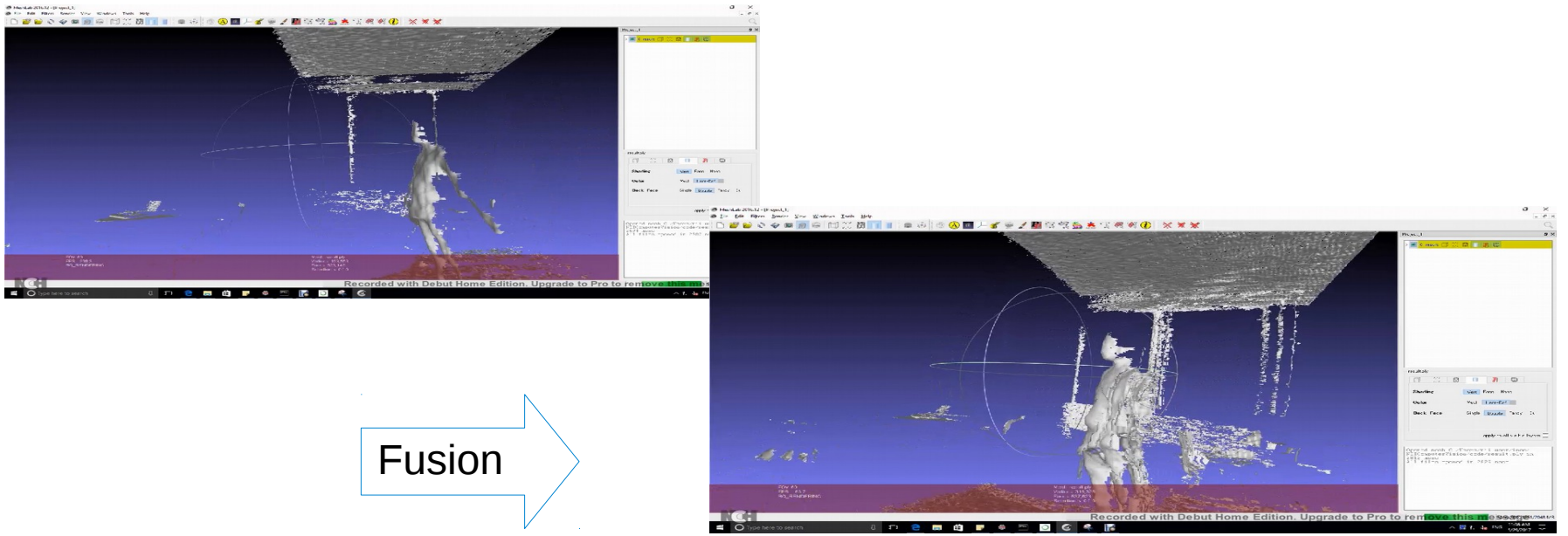


Conclusion : data from Mesh are correct

Testing it with fusion

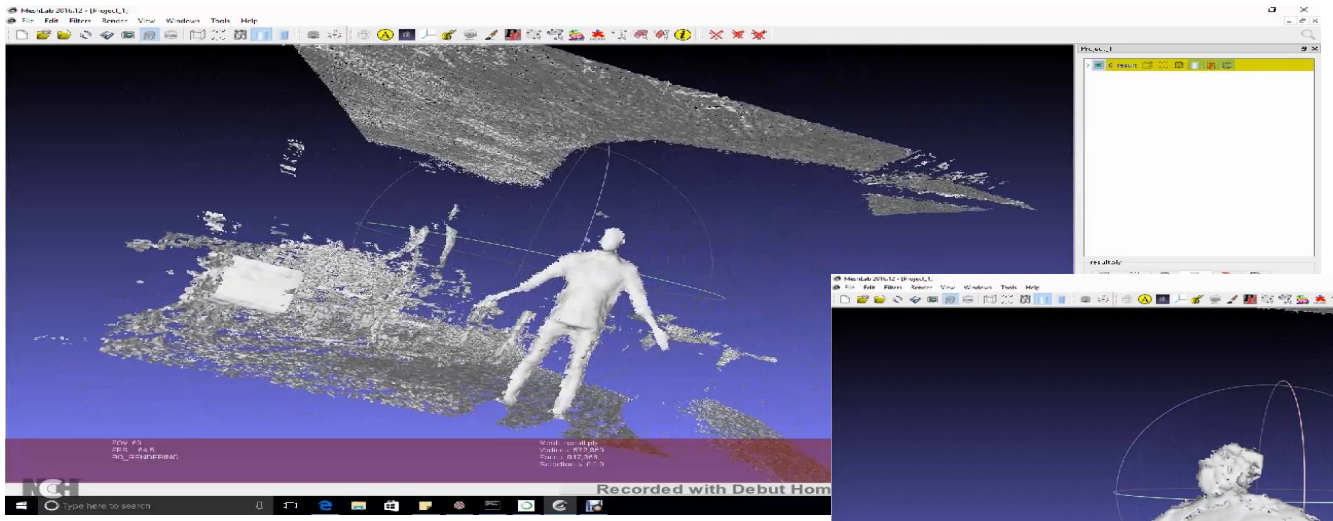
Progress

- Using frame to frame fusion:
  - Transform list in mat with draw function

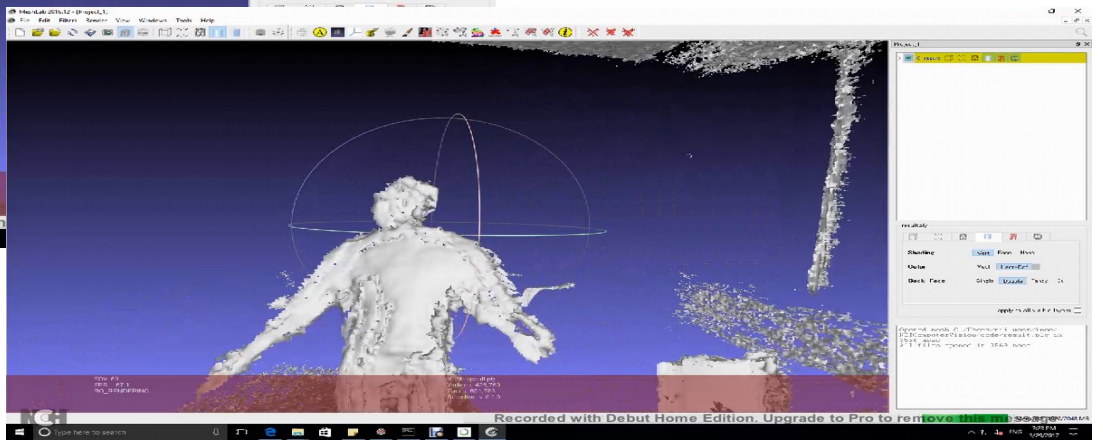


Progress

- Realizing that I had Pose²

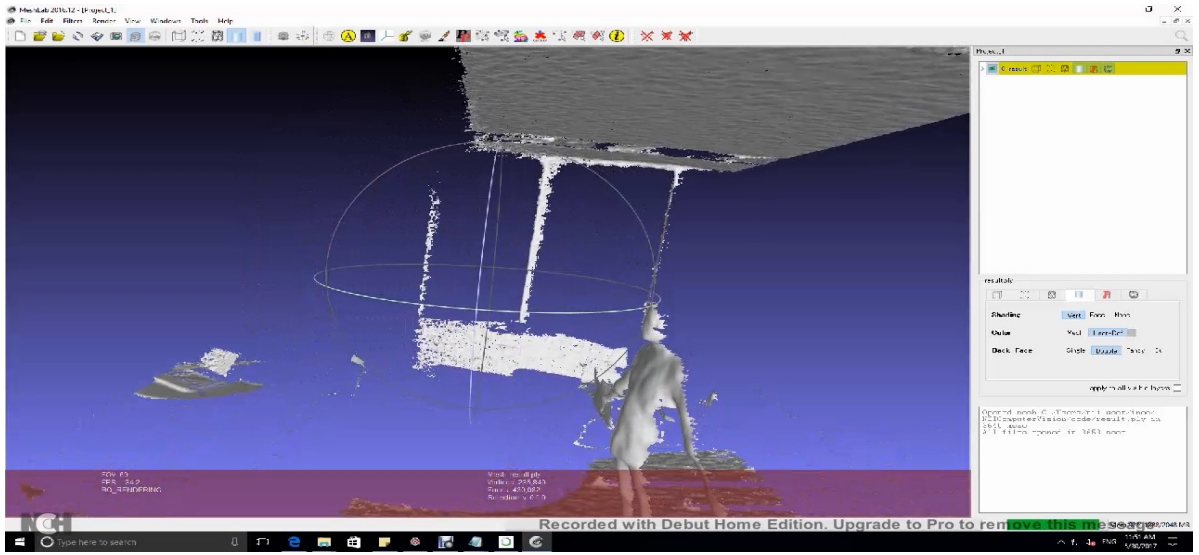


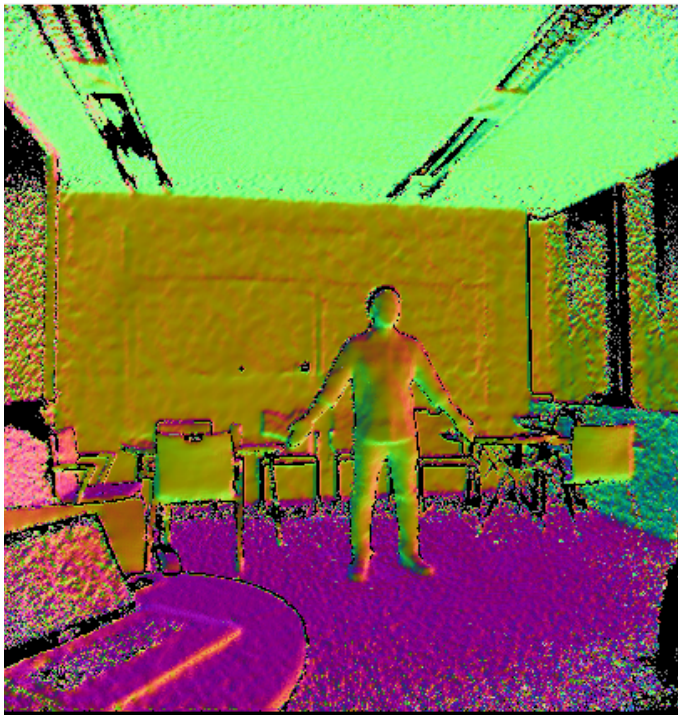
Pose no square



Progress

- Using frame to frame fusion:
  - Compare Matrix generated by MC's lists with its initial form => no correspondence → Align Mesh with the image generating Mesh





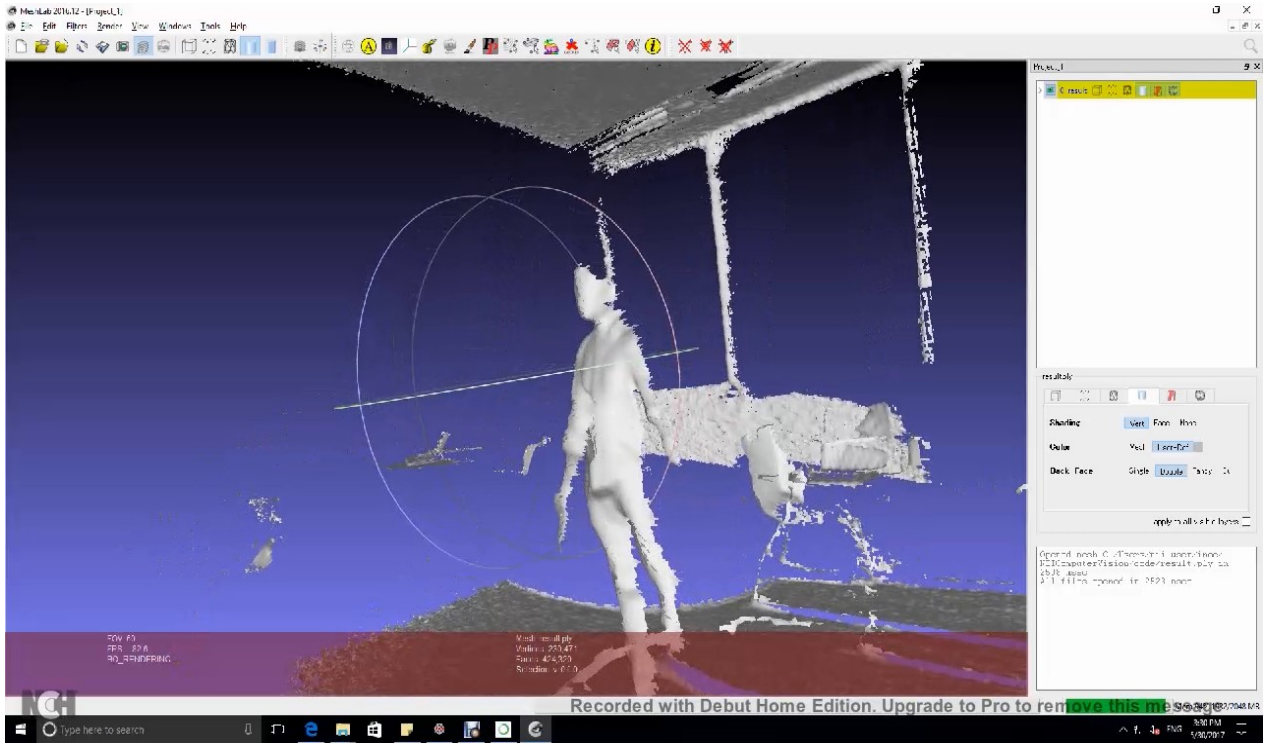
Progress

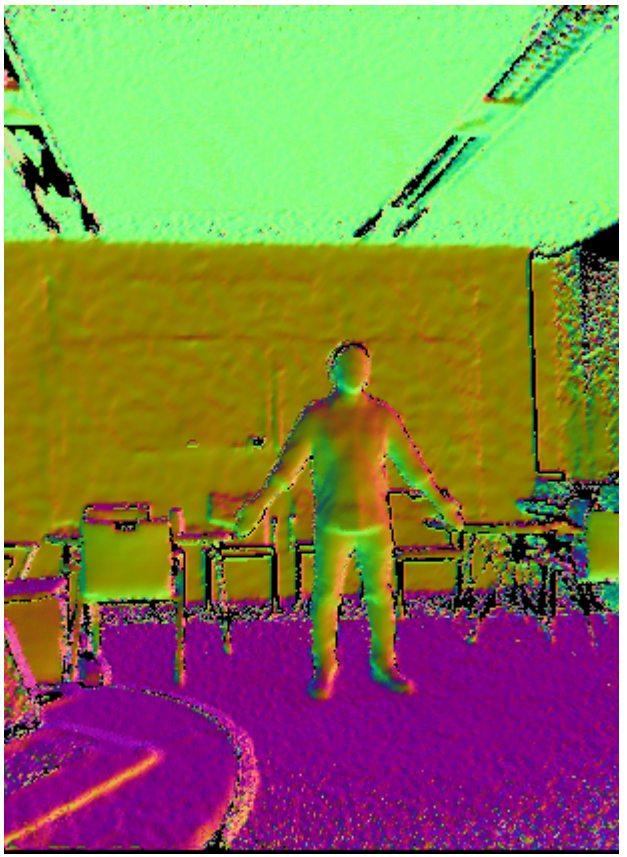
- Algo

1) Get Image Img1
2) TSDF + Marching Cubes (giving lists)
3) Convert list to matrix M1 (as in draw function)
4) Get new Image Img2
5) Align Img2 & M1 (frame to frame)
6) Go to 2 (with Img2 becoming Img1)

Progress

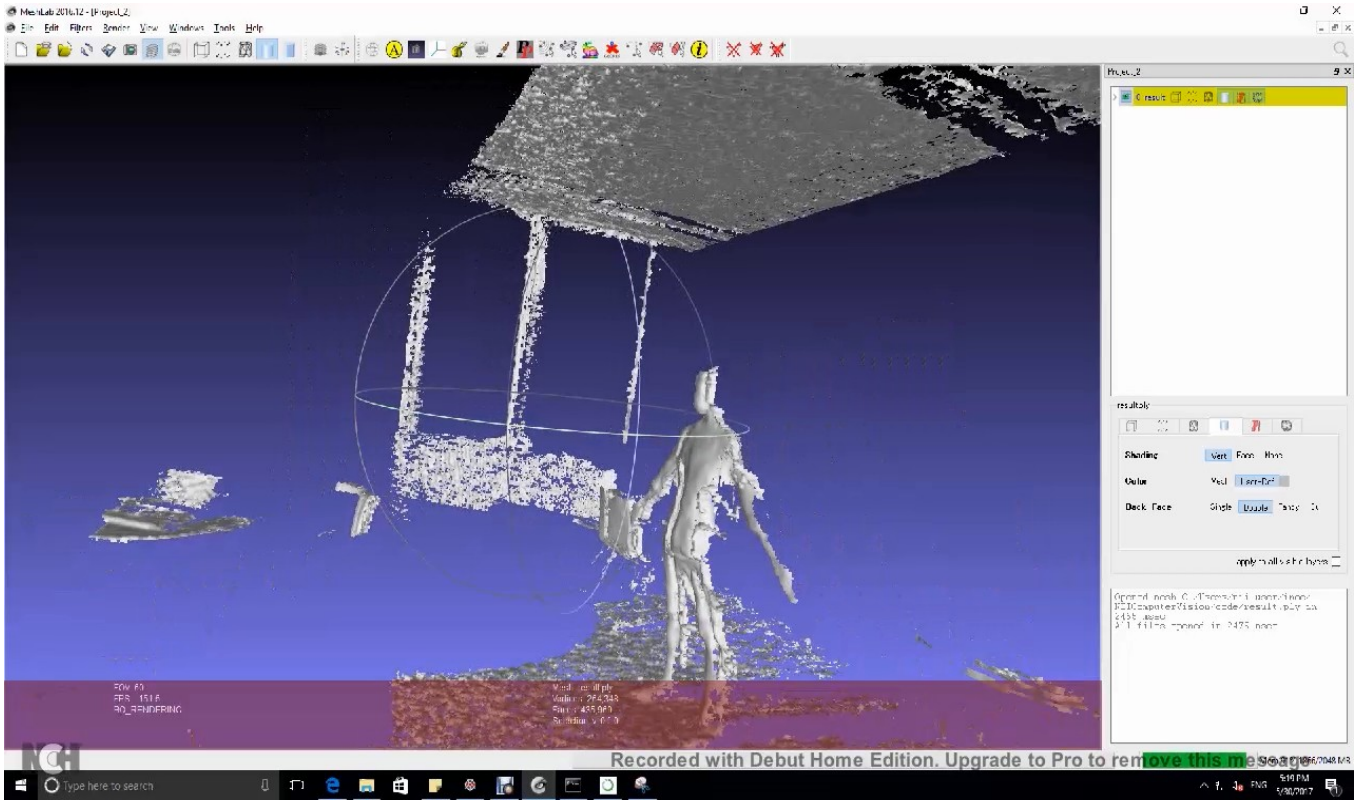
For 10 images





Progress

- Identify the source of the problem in mesh tracking

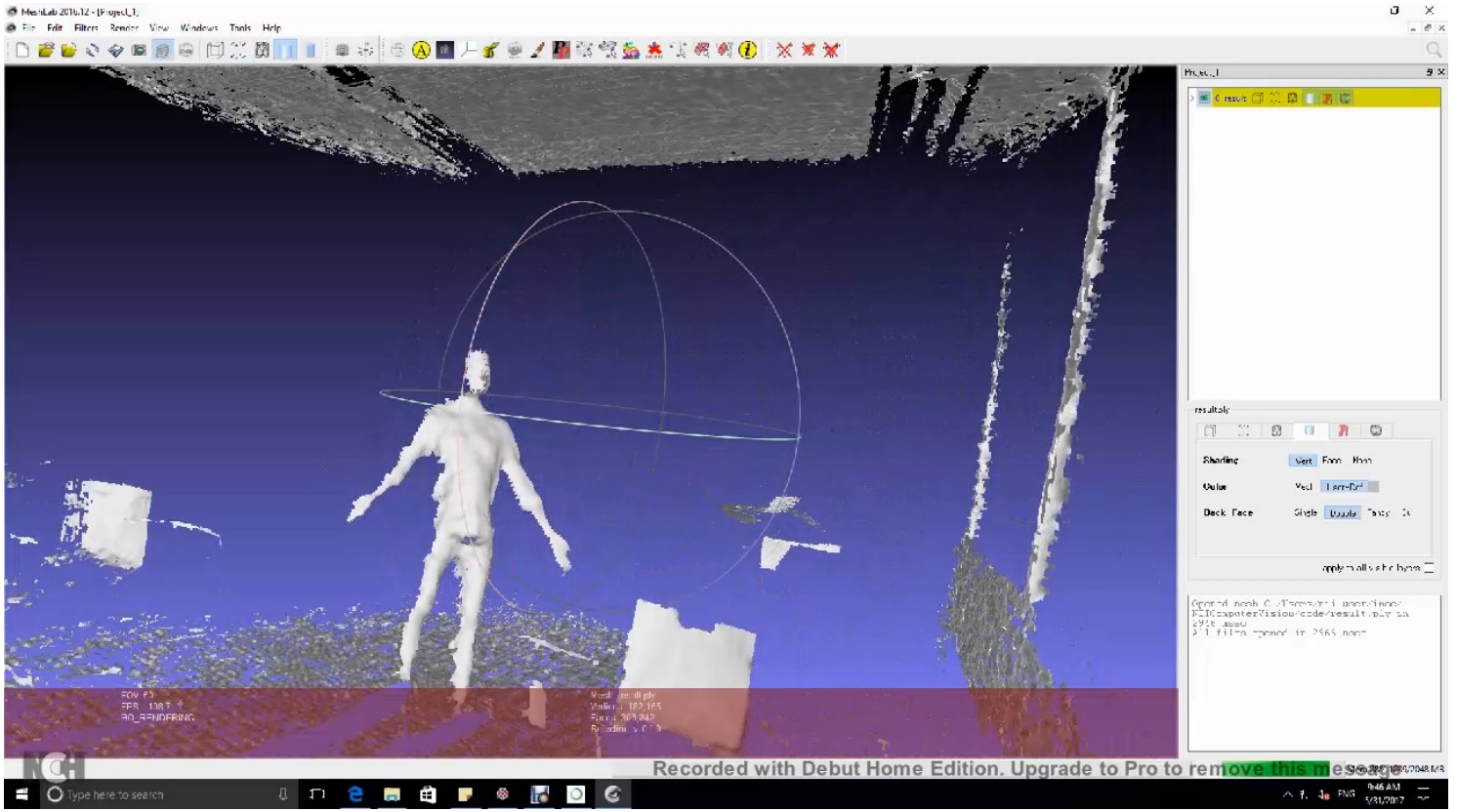


Just changing tracking functions From frame to frame TO mesh to frame

1) Get Image Img1
2) TSDF + Marching Cubes (giving lists)
3) Convert list to matrix M1 (as in draw function)
4) Align M1 and Img1 (F2F)
5) Get new Image Img2
6) Align Img2 & Mesh (M2F)
7) Go to 2 (with Img2 becoming Img1)

Progress

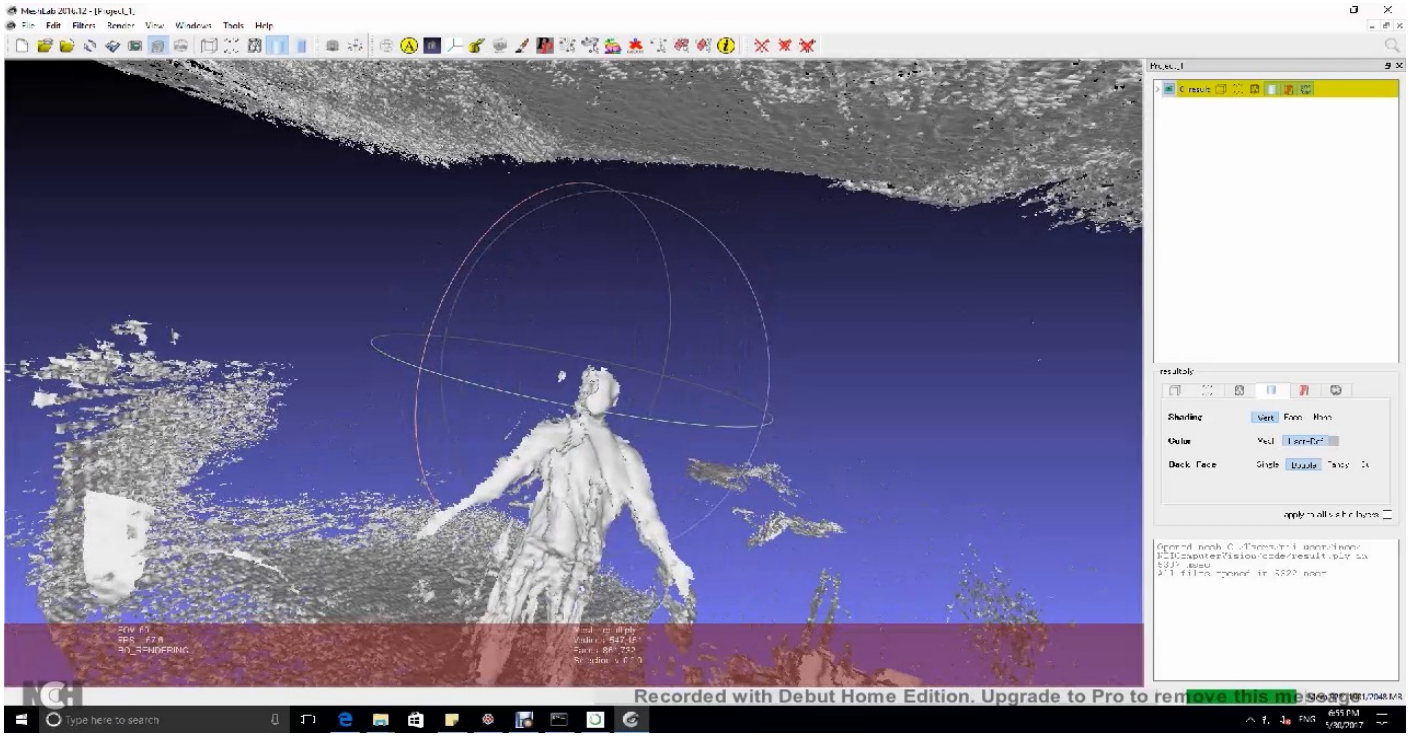
- Identify the source of the problem in mesh tracking



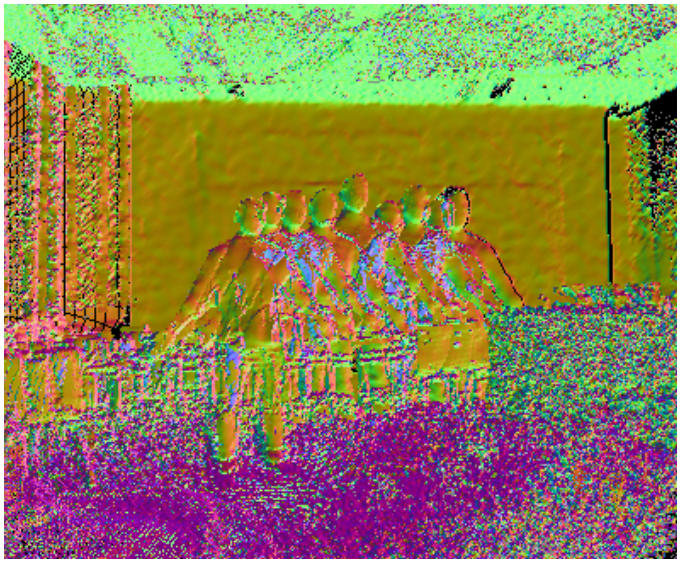
The first step was missing the alignment of the first image with its own mesh.

Progress

- Identify the source of the problem in mesh tracking



Expanding fusion to 10 images



Progress

- Identify the source of the problem in mesh tracking
  - Lists conversion into matrix just to test data from mesh
  - Match MeshTrack:
    - Normals : 9 232
    - Vertices : 131 268
    - Final : 0
    - Inverting normals and vertices did not change number of normals match but vertices =0
  - Match frame to frame and list conversion but no updated mesh to frame alignement:
    - Normals : about 40 000
    - Vertices : about 130 000
    - Final : 38 000
  - Match frame to frame tracking and list conversion:
    - Normals : about 100 000
    - Vertices : about 200 000
    - Final : about 60 000

Progress

- Research papers:
  - Breen, D. E., & Whitaker, R. T. (2001). A level-set approach for the metamorphosis of solid models. IEEE Transactions on Visualization and Computer Graphics, 7(2), 173-192. DOI: 10.1109/2945.928169

Action plan

- Mesh tracking
- Papers research : find idea!
- Fusion for each segmented body part separately:
  - Coordinates change one by one
  - Fuse one by one

Q&A

- Jacobian matrix?

```
In RegisterRGBD :
        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]) ))

         Buffer_B[Indexes_ref[:][:]] = np.dstack(w*mask[:,:]*(nmle[:,:,0]*(Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,0] - pt[:,:,0])\
                                              + nmle[:,:,1]*(Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,1] - pt[:,:,1])\
                                              + nmle[:,:,2]*(Image2.Vtx[line_index[:][:], column_index[:][:]][:,:,2] - pt[:,:,2])) ).transpose()

Nmle and vtx = Current Image, Image2 = former Image

In RegisterrGBDMesh:
        Buffer[Indexes_ref[:]] = np.stack((w*mask[:]*nmle[ :,0], \
            w*mask[:]*nmle[ :, 1], \
            w*mask[:]*nmle[ :, 2], \
            w*mask[:]*(-NewImage.Vtx[line_index[:], column_index[:]][:,2]*nmle[:,1] + NewImage.Vtx[line_index[:], column_index[:]][:,1]*nmle[:,2]), \
            w*mask[:]*(NewImage.Vtx[line_index[:], column_index[:]][:,2]*nmle[:,0] - NewImage.Vtx[line_index[:], column_index[:]][:,0]*nmle[:,2]), \
            w*mask[:]*(-NewImage.Vtx[line_index[:], column_index[:]][:,1]*nmle[:,0] + NewImage.Vtx[line_index[:], column_index[:]][:,0]*nmle[:,1]) ) , axis = 1)

         Buffer_B[Indexes_ref[:]] = ((w*mask[:]*(nmle[:,0]*(NewImage.Vtx[line_index[:], column_index[:]][:,0] - pt[:,0])\
                                  + nmle[:,1]*(NewImage.Vtx[line_index[:], column_index[:]][:,1] - pt[:,1])\
                                  + nmle[:,2]*(NewImage.Vtx[line_index[:], column_index[:]][:,2] - pt[:,2])) ).transpose()).reshape(Buffer_B[Indexes_ref[:]].shape)

Nmle and vtx = Mesh (former Image), NewImage = current Image
```

Q&A

- Test with simpler data?