# Computer Vision AI – Final Assignment
# 3D Mesh Generation and Texturing

Tuesday 10th May, 2016

Students are supposed to work on this assignment for two labs. Some additions and changes might be done during this period. Students will be informed for these changes via blackboard. The analysis and the conclusions must be included in a report. A final report and source code should be zipped and sent to computervision1516@gmail.com 31-05-2016, 23:59 (Amsterdam time). Subject of the email should be [CV2 Final Assignment].

## 1 Depth-based and Texture-based 3D Reconstruction Comparison

In the first three weeks of the computer vision you have implemented a depth based 3D reconstruction method (i.e. ICP). The following three weeks you have implemented a texture based 3D reconstruction method (i.e. SFM). Please briefly explain what the advantages and disadvantages of these methods are. Do you think these two methods could be used together to get better 3D reconstruction (Please comment on this question)?

## 2 3D Meshing and Watertighting

In both previous assignments, you have obtained point clouds as 3D models. In this assignment, you are expected to generate 3D mesh from registered point clouds. You will work with the provided depth and color images, and camera poses. The sample data is in ".3ds" format, you will use the provided C++ source code to read this data. To merge point clouds into a single one, you should follow the steps as described by algorithm 1:

To get the final 3D mesh, you need to pass *model_point_cloud* to a mesh generation method (i.e. poisson, marching cube). You will get a 3D mesh. This 3D model will have holes on the part where there is no camera view. You are expected to fill the holes by applying watertight.

**Algorithm 1** Merging

```
procedure MergingPointclouds(3DFrames)
    model_point_cloud ← emptyPointCloud()
    for frame in 3DFrames do
        depth_image ← frame.depth_image
        focal_length ← frame.focal_length
        camera_pose ← frame.camera_pose
        point_cloud ← depthToPointCloud(depth_image, focal_length)
        point_cloud_with_normals ← computeNormals(point_cloud)
        point_cloud_with_normals ← transformPointCloud(point_cloud_with_normals, camera_pose)
        model_point_cloud ← concatPointClouds(model_point_cloud, point_cloud_with_normals)
    end for
    return model_point_cloud          ▷ Contains al points XYZ and normals
end procedure
```

# 3 Coloring 3D Model

The 3D model that you have generated in section 2 is not colored yet. To this end, you should follow the steps as described by algorithm 2:

**Algorithm 2** Texturing

```
procedure Texture(mesh, 3DFrames)
    polygons ← mesh.polygons
    point_cloud ← mesh.point_cloud
    for frame in 3DFrames do
        depth_image ← frame.depth_image
        focal_length ← frame.focal_length
        camera_pose ← frame.camera_pose
        transformed_point_cloud ← transformPointCloud(point_cloud, camera_pose.inverse())
        for polygon in polygons do
            if polygon visible to this camera then
                uv_coordinates ← getUVCoordinates(polygon, transformed_point_cloud)
                assign uv_coordinates of this camera to the polygon
            end if
        end for
    end for
end procedure
```

For this assignment, you can make use of the existing libraries (e.g. http://pointclouds.org/), you are not expected to implement these functions on your own. However, you are expected to verify your choices of algorithms. For instance, you should explain why do you select poisson algorithm over marching cube.