Practical report : CVPR

Subject : 3D shape modeling

Written by : Arthur Vivière, Dakri Abdelmouttaleb
The 24 février 2022

1 Part 1 : Voxel carving

Given a set of 12 images taken of the `Al.off` mesh model from different perspectives (with the respective camera projective matrices), we should be able to reconstruct a 3D model. It is sufficient to project a uniform grid on each of the images using the associated camera matrices on the images and only keep those who fall inside the contours of the 3D form ie.

$$\forall\ x\ \in [X,Y,Z] \text{ if } p = proj_{cami}(x)/p[2] == 0 \ then \text{ Occupency} = 0$$

. The projection of the uniform grid $[X,Y,Z]$ on one of the cameras is presented in Figure 1. Running the 12 images give us the results in Figure2.

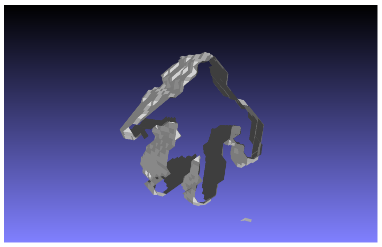


FIGURE 1 – One camera reconstruction

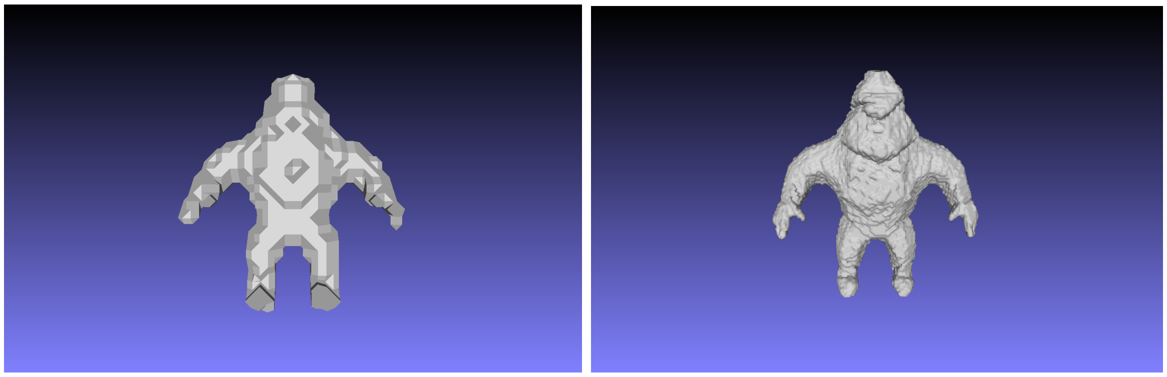


FIGURE 2 – Results using a resolution of 30 and 120 respectively

N.B : The fact that the inside of the representation is void is due to the marching cube reconstruction algorithm : a one camera projection being casted to infinity inside the convex hull, the triangles are not rendered and therefore not represented inside Meshlab.

2 Part 2 : Neural implicit modeling

2.1 Programming strategy

Q1. The MLP neural network described in the program is composed of four linear layers with different non-linear activation functions (`tanh, ReLU`), the output being a probability of being inside or outside the 3D form (which is the result of the implicit form learning) being the objective of the training, a sigmoid function can be used (but will not be used given **Q3.**).

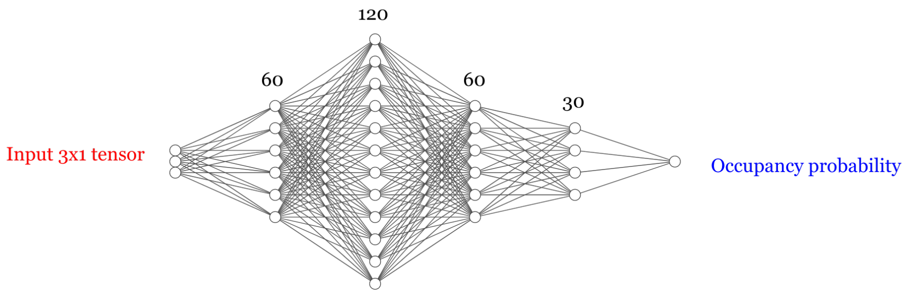


Figure 3 – MLP network architecture.

Q3. The X,Y,Z are indexed so that each pixel has its coordinates in the same array, accessing the occupancy grid and the X,Y,Z can be done through the same indices. Occupancy is now a single list and each index corresponds to a pixel. As MLP network inputs, The arrays were flattened.

Q3. The `BCEWithLogitsLoss` loss function combines a Sigmoid layer and the BCELoss in one single class. According to the Pytorch documentation, this version is more numerically stable than using a plain Sigmoid followed by a BCELoss as, by combining the operations into one layer, we take advantage of the log-sum-exp trick for numerical stability.

Q4. p_weigh corresponds to the percentage of outside points in the occupancy grid. As such, the loss function associated with the positive points is weighted down compared to that of the negatives at each epoch.

Q5. To create the batches, we take a permutation of the indices for the pixels and the occupancy and we take a slice of size *batch_size*.

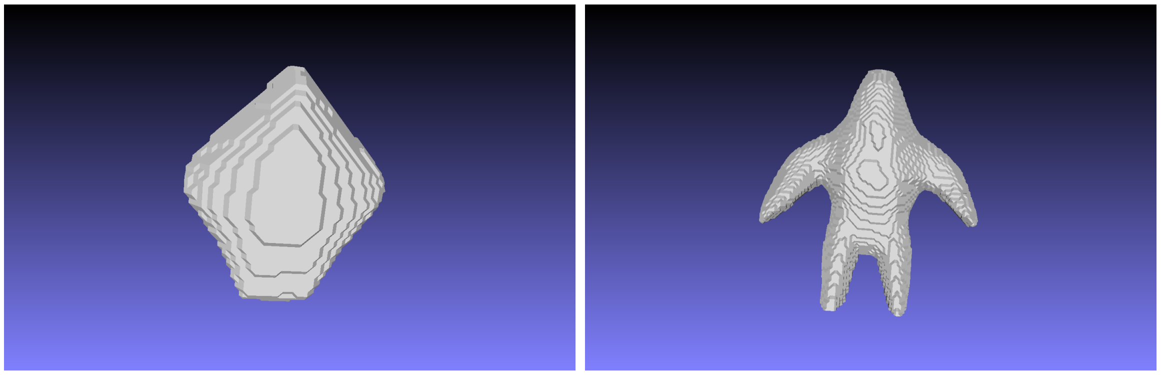
Q6. The accuracy function `binary_acc` evaluates the percentage of correct pixels after passing through the neural network. And it's not used in training. We will use it however to measure the performance of the neural network during training.

Q7. The Sigmoid output calculates probabilities for each pixel to determine if it is 0 or 1 (as in inside or outside the 3D form) then regenerates an occupancy matrix, we use the marching cubes algorithm similarly to part 1 with the new occupancy in order to generate a mesh from the discrete point cloud.

Q8. Resolution being given by a number $n \in \mathcal{N}$. The parameter numbers of the MLP are $3 \times 60 + 60 \times 120 + 120 \times 60 + 60 \times 30 + 30 \times 1 = 16410$ weights and having 270 perceptrons in the hidden layers and one in the output layer it computes 271 biases. In terms of physical memory mounted on the GPU, a batch is 135,000 times less than the uniform voxel grid and 10800 times less than the original image set and calibration.

2.2 Program editing

Q1. We add the perspective projection program we used in the first part to generate an initial coarse occupancy grid for training. The following results respectively correspond to a resolution of 50 and 100 on 10 epochs. On the same training lapse, the resolution (as in the initial training data) seems to be very impactful on the learning process.



The following result is that of a training on a 50 resolution 3D model (the previous low resolution) but this time on a 100 epochs. The training process helps to converge to a better 3D model.

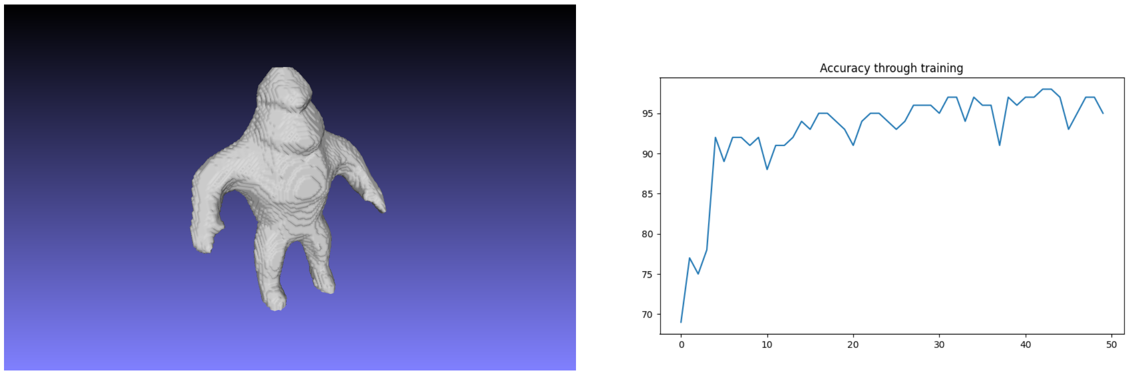


FIGURE 4 – Training on an initial 50 resolution model for 50 epochs and the associated accuracy graph.

Q2. We use random vectors $X_{rand}, Y_{rand}, Z_{rand}$ to fill the MLP input. The visualisation is still done through the uniform grid. We get the following result for training for 100 epochs on a 50 resolution. The result is not too bad, the MLP is still able to reconstruct the 3D model (with reference to the projection constructed model) even through randomly sampled Voxel values inside the grid.

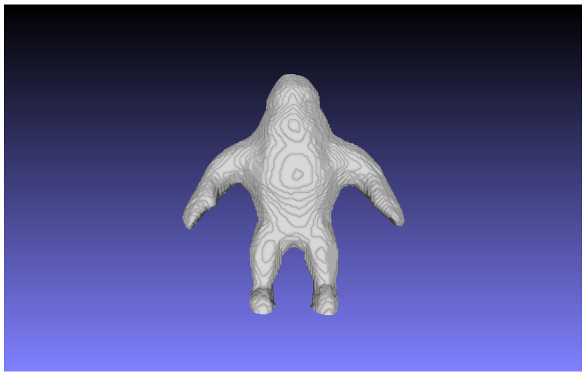


FIGURE 5 – Results for random values in a $[-1, 1] \times [-1, 1] \times [-0.5, 0.5]$ grid.

Q3. An optimisation method for the MLP training will be to replace the weighting of the input points (the loss associated to the inside pointsis down-weighted) with the reduction of the outside points set before the training. We tried to add offsets on higher resolution and inspect the binary accuracy and the shape reconstruction quality. There exists a middle point in which the MLP's accuracy is maximal (before it, the accuracy keeps climbing up and after it the model reconstructed becomes truncated). So the strategy we implemented was to compute the bounding box of the generated occupancy grid in order to reduce in a uniform manner the size of the outside points set. The image below shows the result of the MLP training for 50 epochs on a resolution of 50, compared to the previous optimisation scheme, this method seems to be faster and produces somewhat better results under the same training conditions :

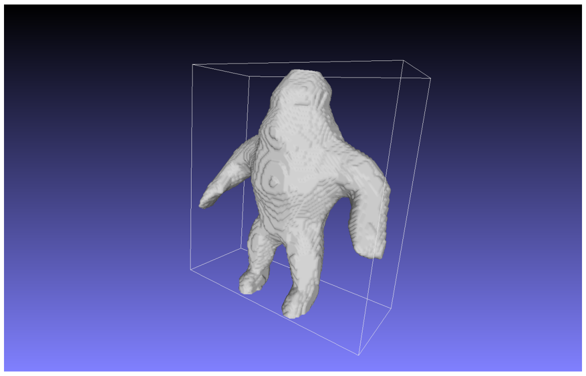


FIGURE 6 – Results using occupancy bounding box reduction.

Q4. We try two other configuration for the MLP network : (a) we add a hidden layer of 180 perceptrons between the 60 and 120 layers to deepen and enlarge the neural network, (b) we remove the 120 perceptron layers and adapt the other layers accordingly.

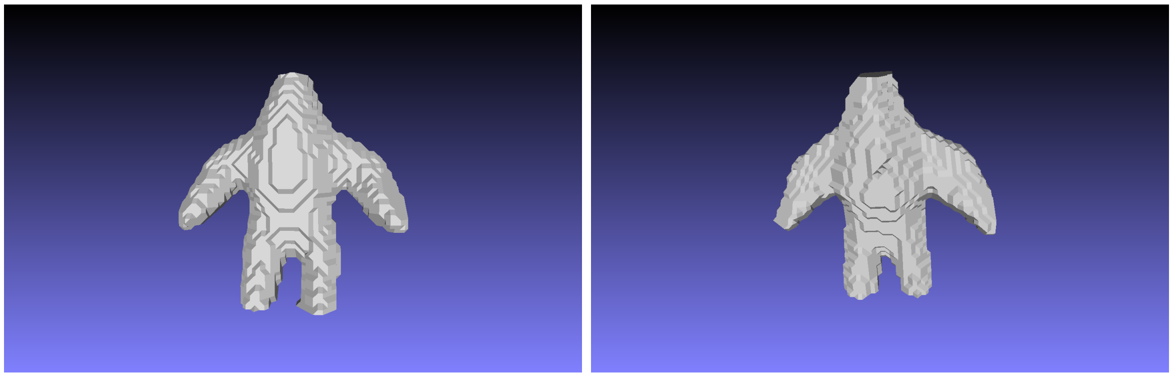


FIGURE 7 – Results for deeper and less deep MLP networks respectively for the same resolution and training time.