Project Work in Image Processing and Computer Vision

Voxel Carving for 3D Reconstruction of Images

Riccardo Spolaor (riccardo.spolaor@studio.unibo.it)

Contents

1 Introduction

In the field of *Computer Vision*, a particularly interesting problem with a diverse range of practical applications is the process of generating the 3D representation of an object, given a set of 2D images.

Voxel Carving deals with this task by building a three-dimensional voxel grid that fully contains the object to be reconstructed. By using information from a picture of the object, we can easily decide what voxels aren't part of the object itself judging by their projections in the image and we can carve them out. Then, using information from subsequent pictures of the object taken from different perspectives, we can cut out more and more from the grid until we end up with a good approximation of the object in voxel space.

In this report, *Voxel Carving* is employed in order to reconstruct a 3D digital version of a toy dinosaur from a series of two-dimensional images of the object itself from different perspectives. The images of the dinosaur along with their respective *Perspective Projection Matrices* have been collected from the website of the *Department of Engineering Science of the University of Oxford* [1]. The images of the dinosaur can be observed in Figure 1.

Figure 1: The 37 images of the toy dinosaur in different perspectives.

Firstly, a binary mask that segments the dinosaur from the background in each image is obtained through a pre-processing step. Secondly, a voxel grid of 3D points that encloses the dinosaur is created and projected on each image through the use of the *Perspective Projection Matrices*. Next, the *occupancy* of each 3D voxel grid points is computed as the number of times that it is projected inside the binary mask of the dinosaur in each image. In other words, the number of times a voxel point can be "seen" inside the dinosaur shape is computed considering each available perspective. Finally, just the points that reach a certain occupancy threshold are kept and they are considered as points that define the 3D mesh of the dinosaur toy.

The technology stack used to develop the solution is composed of *Python* [2] as programming language and *OpenCV* [3] as a Computer Vision framework. Furthermore, the software *ParaView* [4] has been employed to inspect the results of the 3D reconstruction of the toy dinosaur. The source code of the implementation that is illustrated in the report is available in the following *Github* repository [5], where the solution is presented through a notebook.

2 Data

The available data are the images of the toy dinosaur in different perspectives and the relative *Perspective Projection Matrices*.

2.1 Toy Dinosaur Images

The toy dinosaur that has to be reconstructed in 3D is captured through 37 images that depict it in different perspectives. Each image is an *RGB* array with values between 0 and 255 for each pixel in each channel. The toy dinosaur has a bright orange color scheme that highly contrast with the blue background of the scene. It is interesting to observe how a black border is visible in the top and right corners of each image. In Figure 2 three sample images of the dinosaur from different angles are shown.

Figure 2: Sample images of the toy dinosaur from different angles.

2.2 Perspective Projection Matrices

Each *Perspective Projection Matrix* P contain both the intrinsic (A) and extrinsic camera parameters (R and t)

$$P = A\ [R \mid t]$$

Where $A \in \mathcal{R}^{3\times3}$ is the intrinsic camera parameters matrix, $R \in \mathcal{R}^{3\times3}$ is the rotation matrix and $t \in \mathcal{R}^{3\times1}$ is the translation vector.

Given a *3D World Reference Frame* point M in homoegeneous coordinates $M = [X,\ Y,\ Z,\ 1]$, we can approximately obtain through the Perspective Projection Matrix P the relative *2D pixel coordinates*, $m^* \approx [x,\ y,\ w]$ in the *Camera Reference Frame*. It can be then scaled by the third coordinate to obtain the *canonical pixel coordinates* of the point $m = [x/w,\ y/w]$, which is the projection of M on the image.

$$m^* = [x,\ y,\ w] \approx P\ M$$

$$m \approx [x/w,\ y/w]$$

3 Images Pre-Processing and Binary Segmentation

As an initial step, the images are pre-processed and segmented in order to obtain a binary mask that separates the silhouette of the dinosaur toy from the background. These binary mask are finally used in the *Voxel Carving* process to reconstruct the toy in 3D.

3.1 Images Pre-Processing

Firstly, the black border on the top and right part of each image is cropped. This operation is performed because these extra borders could interfere with the binary segmentation operation that has to be performed, this leading to worse results. An example of this operation is illustrated for the 11^{th} image of the toy dinosaur in Figure 3.

(a) Original image.

(b) Image with the cropped border.

Figure 3: Example of the border-cropping operation applied to the 11^{th} image of the toy dinosaur.

A *Gaussian Filter* is then applied on the images in order to smooth them and remove any source of Gaussian-like noise that could possibly impact the results of the binarization. An example of the operation applied on the 11^{th} image of the dinosaur is observed in Figure 4.

(a) Image with the cropped border.

(b) Image with the applied Gaussian filter.

Figure 4: Example of the Gaussian filtering operation applied to the 11^{th} image of the toy dinosaur.

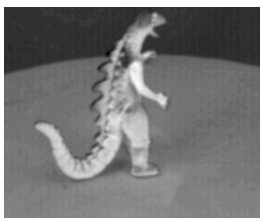
The background of the images is of a distinct blue colour with respect to the dinosaur, hence colour segmentation through *Mahalanobis Distance* from the background could have been an appropriate technique. Nonetheless, this approach requires an additional pre-processing step that consists in estimating the background color by computing the mean and covariance matrix from some samples representing it.

It was observed that the images in the B channel of the LAB colour space present a marked distinction between the background and the toy. Hence a transformation of the filtered images to this specific channel has been employed as it could lead to a less costly segmentation approach that exploits its characteristic as described in section 3.2.

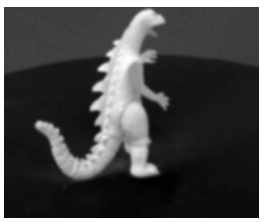
Figure 5 shows the transformation of the first Gaussian-filtered image of the toy dinosaur in the LAB colour space along with each of its single channels.

(a) LAB image.

(b) Channel L.



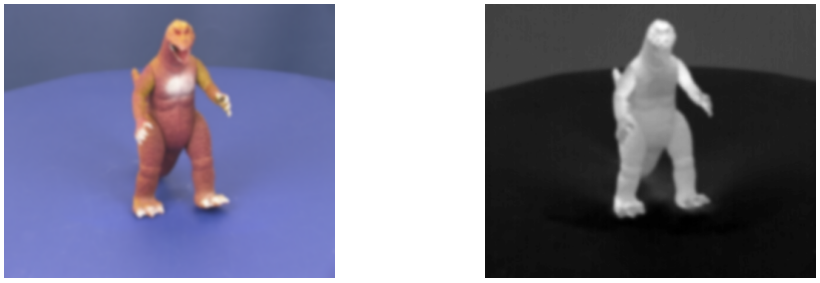
(c) Channel A.



(d) Channel B.

Figure 5: Transformation of the first Gaussian-filtered images of the toy dinosaur in the LAB colour space.

Figure 6 shows the conversion of the 11^{th} Gaussian-filtered image of the toy dinosaur to the B channel of the LAB colour space.



(a) Image with the applied Gaussian filter.

(b) Image converted to the B channel of the LAB colour space.

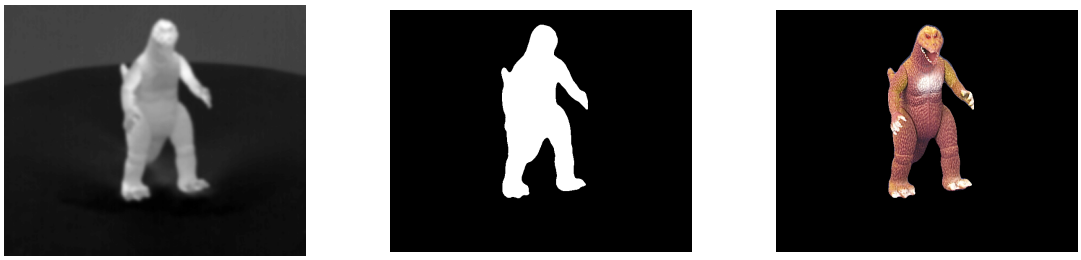
Figure 6: Example of the conversion to the B channel of the LAB colour space applied to the 11^{th} image of the toy dinosaur.

3.2 Binary Segmentation

To obtain the binary masks of the toy dinosaur, *Otsu's Algorithm* has been applied. It computes for each image a different global threshold t that divides two maximally homogeneous regions.

The method analyses an image gray-level histogram, detecting the regions in a way such that their within-group variance (σ_W^2) is minimal. In other words, the sum of the weighted variance of the two maximally homogeneous regions divided by t is minimized. The two variances are weighted according to the number of pixels in each region. The algorithm uses an equivalent and faster approach by maximizing the so called between-group variance (σ_B^2) of the two regions. This value indicates how well they are separated. The segmentation of an image is then performed according to the obtained threshold t as in the previous case.

Otsu's Algorithm is designed to be robust to small changes of intensity in inherently binary images as the ones of the toy dinosaur in the B channel of the LAB colour space. The results of this operation are satisfactory as it can be observed for the example in Figure 7.



(a) Image in the B channel of the LAB colour space.

(b) Binary mask obtained through *Otsu's Algorithm*.

(c) Image of the toy dinosaur segmented by its binary mask.

Figure 7: Example of the masking process applied to the 11^{th} image of the toy dinosaur.

The pixels of the borders that were removed at the beginning of the pre-processing step are finally added back as *background pixels*. This operation is applied in order to guarantee that the binary masks are back to the original size of the toy dinosaur images, properly matching them in order to avoid any mismatching between 3D world points and pixel points during their projection in the Voxel Carving process described in section 4.

The results are clean, hence no *closing* operations are needed to resolve imperfections in the masks.

4 Voxel Carving

In this last section, a 3D voxel grid is built in a way that the projection of its *3D World Reference Frame* points to the *2D pixel points* of the image manages to completely enclose the silhouette of the toy dinosaur.

Finally, for each voxel point it is computed its *occupancy*, which is how many times its projection falls inside the binary mask of the dinosaur considering all the different perspectives. Just the points that manage to overcome a certain *occupancy threshold* are considered as being points that constitute the 3D mesh of the dinosaur toy.

4.1 Voxel grid definition

Initially, a 3D voxel grid is created. An equal number of points is considered for each axis of the grid (120), thus composing a final 3D grid of $120 \times 120 \times 120 = 1,728,000$ points.

The coordinates of each 3D point $M = [X,\ Y,\ Z,\ 1]$ are considered in their *homgeneous* form in the *3D World Reference Frame*, to guarantee they can be projected to their *2D pixel coordinates* m by the use of each respective *Perspective Projection Matrix* P as explained in section 2.2.

$$m^* = [x,\ y,\ w] \approx P\ M$$

$$m \approx [x/w,\ y/w]$$

The homogeneous coordinates of the grid in the *3D World Reference Frame* are manually adjusted in order to enclose the dinosaur toy.

The coordinates of the grid are computed through the use of the *NumPy* [6] function `meshgrid` and for each axis they originally range as integer numbers between the values 0 and 199. To guarantee they enclose the dinosaur toy, each coordinate X, Y and Z of a point M is scaled to X', Y' and Z' through *Min-Max scaling*:

$$X' = x_{min} + \frac{(X - 0)(x_{max} - x_{min})}{199 - 0}$$

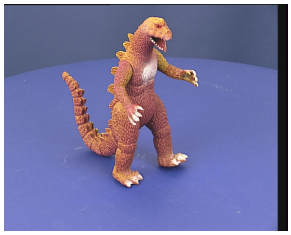
$$Y' = y_{min} + \frac{(Y - 0)(y_{max} - y_{min})}{199 - 0}$$

$$Z' = z_{min} + \frac{(Z - 0)(z_{max} - z_{min})}{199 - 0}$$

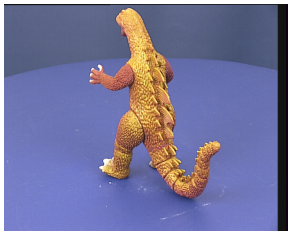
In particular:

- The minimum x-axis coordinate x_{min} is -0.05.
- The maximum x-axis coordinate x_{max} is 0.05.
- The minimum y-axis coordinate y_{min} is -0.1.
- The maximum y-axis coordinate y_{max} is 0.04.
- The minimum z-axis coordinate z_{min} is -0.75.
- The maximum z-axis coordinate z_{max} is -0.5.

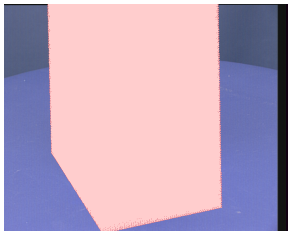
Figure 8 shows the projection of the grid on two selected images of the toy dinosaur at different angles.



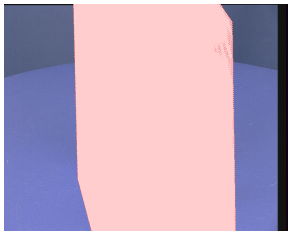
(a) 8^{th} image.



(b) 25^{th} image.



(c) Voxel grid over the 8^{th} image.



(d) Voxel grid over the 25^{th} image.

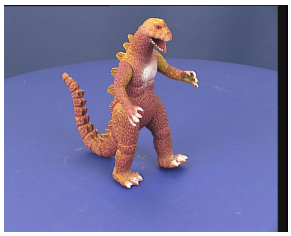
Figure 8: Projection of the voxel grid on two sample images of the toy dinosaur from different angles.

4.2 Occupancy computation

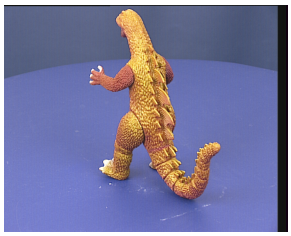
The count of the *occupancy* of each pixel point p is computed by summing how many times each falls in the mask of the toy dinosaur considering the different perspectives available. This operation is completed in a time of ≈ 5.56 seconds.

$$\text{occupancy}(p) = \sum_{\forall\ M\ \in\ \text{masks}} \mathbb{1}_M(p)$$

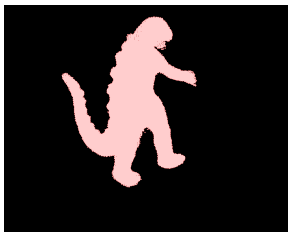
$$\mathbb{1}_M(p) = \begin{cases} 1, & \text{if } p \in M, \\ 0, & \text{otherwise} \end{cases}$$



(a) 8^{th} image.



(b) 25^{th} image.



(c) Selected voxels over the 8^{th} image.

(d) Selected voxels over the 25^{th} image.

Figure 9: Projection of the selected voxel points on two sample images of the toy dinosaur at different angles.

Finally, a *occupancy thresholding* operation is applied and just the points having an *occupancy* of at least 32 out of 37, meaning that they appear inside at least 32 of the 37 masks, are selected as the points which are part of the 3D mesh of the toy dinosaur. These points represent the ones that have

not been carved out from the original voxel grid.

Figure 9 shows the projection of the selected voxel grid points after the *occupancy thresholding* operation on two selected images of the toy dinosaur at different angles.

5 Results

The systems proposed manages to build a proper 3D mesh of the toy dinosaur as seen from the projection of the selected voxel points in Figure 10.

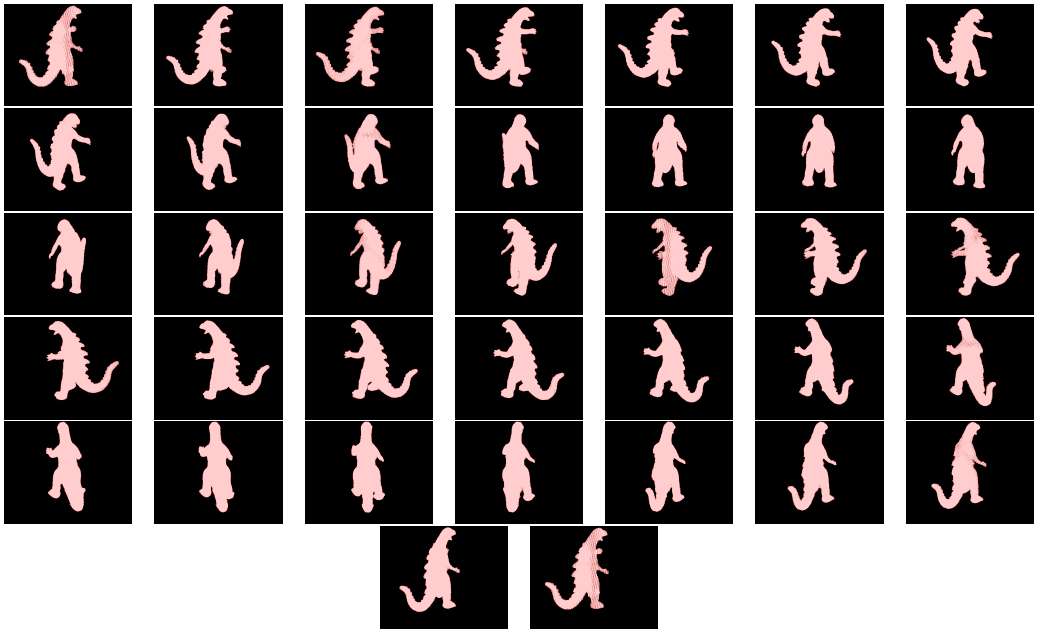
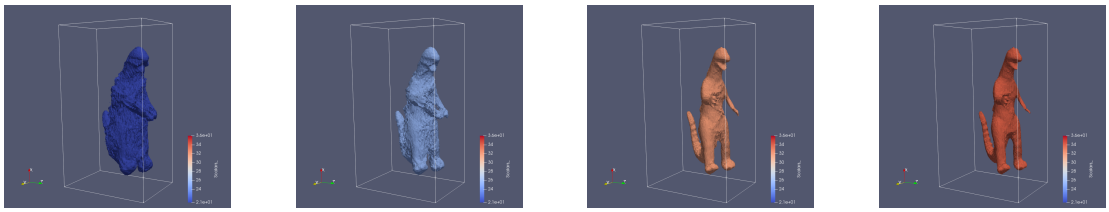


Figure 10: The selected voxel points projected over the 37 perspectives of the toy dinosaur.

In addition, the time employed to perform the voxel carving is not too high, as the operation takes less than 6 seconds.

Moreover, the occupancy threshold of 32 is considered the best among different possibilities, as lower values lead to the inclusion of background points in the 3D mesh of the toy dinosaur, while higher ones result in loss of details, such as in the fingers of the dinosaur.

Figure 11 highlights the differences in the 3D mesh of the toy dinosaur obtained while considering different occupancy thresholds.



(a) Threshold = 21. (b) Threshold = 25. (c) Threshold = 32 (selected). (d) Threshold = 34.

Figure 11: Resulting 3D mesh of the toy dinosaur while considering different occupancy threshold values.

Future work could consist in trying to derive the color of each voxel by estimating it from the images of the toy dinosaur, in order to obtain a colored 3D mesh.

References

[1] *Visual Geometry Group - University of Oxford*. URL: https://www.robots.ox.ac.uk/~vgg/data/mview/.

[2] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995. URL: https://www.python.org/.

[3] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000). URL: https://opencv.org/.

[4] James Ahrens, Berk Geveci, and Charles Law. "Visualization Handbook". In: ed. by Charles D. Hansen and Christopher R. Johnson. Burlington, MA, USA: Elsevier Inc., 2005. Chap. ParaView: An End-User Tool for Large Data Visualization, pp. 717–731. URL: https://www.sciencedirect.com/book/9780123875822/visualization-handbook.

[5] Riccardo Spolaor. *Voxel Carving for 3D Reconstruction from Images*. https://github.com/RiccardoSpolaor/Voxel-Carving-for-3D-Reconstruction-of-Images. 2023.

[6] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.