Unsupervised 3D Feature Extraction and Edge Detection Algorithm

Charles Davi

March 12, 2019

1 Introduction

In this note, I'll present an unsupervised algorithm that can extract three-dimensional features from an ordinary two-dimensional image, and detect edges within the image, thereby extracting two-dimensional shape information, in each case in polynomial time.[1]

2 Feature Extraction

Several months ago, I presented an algorithm that can quickly identify features in an image with no prior information, which I explained in some detail in a paper entitled, "A New Model of Artificial Intelligence". In short, the feature identification algorithm works by partitioning an image into maximally distinct regions by making use of methods rooted in information theory.

[1]The code necessary to run the algorithm is available on my researchgate homepage, under the project heading "Information Theory".

Figure 1: An image of a bird, which is shown as rendered in Figure 2 below.

By changing the equation that governs the stopping condition in the feature identification algorithm, we can produce a modified version of the algorithm that runs longer, and identifies fine-grain features in an image, which is in turn called by the 3D extraction and edge detection algorithm that is the subject of this note. The entire algorithm has a run time that is $O(m \log(m))$, where m is the number of pixels in the image.[2]

[2]This run time counts the number of built-in Matlab operations performed, and as a result, vectorized operations are treated as a single operation.

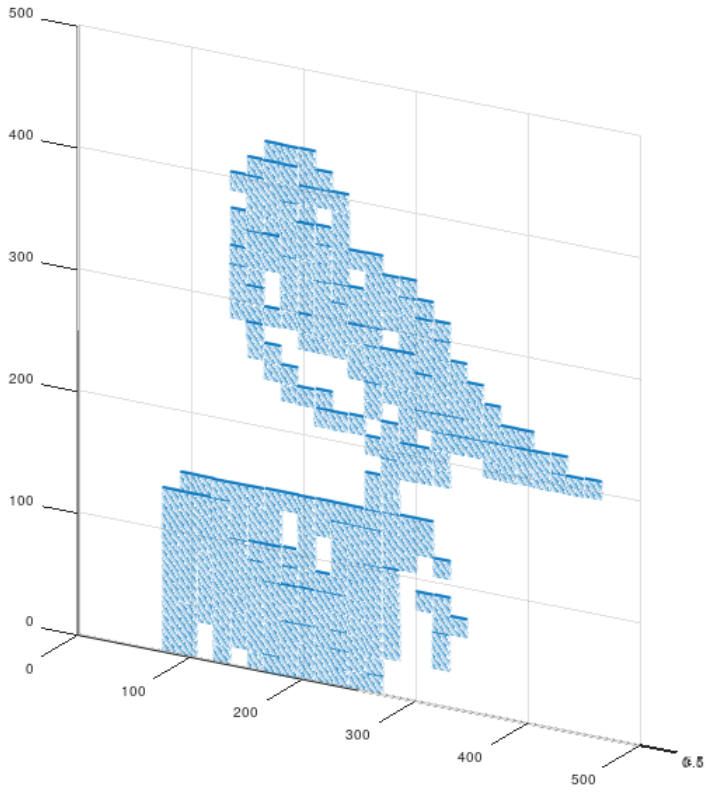

Figure 2: The image from Figure 1, as rendered by the algorithm.

The fine-grain algorithm is useful for making detailed distinctions between background and foreground. This is because the feature identification algorithm assigns a probability to each feature within an image that gives the likelihood that a given feature is part of the background of the image. It turns out that this probability is reliable, and so the background of an image generally has a probability of approximately 0, and the foreground generally has a probability of approximately 1.[3] By turning this probability into a depth parameter, we can map a two-dimensional image into a three-dimensional space, where a background feature has a depth of 1, and a foreground feature has a depth of 0 (i.e., one minus the foreground probability), in turn producing a three-dimensional rendering of the two-dimensional image, where the background features are recessed at a greater depth than the foreground features. If we use the fine-grain

[3]The reliability of this probability of course depends upon how pronounced the distinction between background and foreground is. If the image has an obvious background and foreground, then the probability is generally reliable. If, however, there isn't much of a distinction between background and foreground, then the probability is generally not reliable.

feature identification algorithm, the features will be smaller in scale, allowing for more detailed edge detection. Note that the algorithm removes any regions that have a depth of greater than .95, effectively removing the background of the image.[4]

Figure 3: Another image of a bird, which is shown as rendered in Figure 4 below.

The algorithm can also function as an edge detection algorithm, which is demonstrated in Figures 3 through 6. This is because the algorithm returns three separate vectors, X, Z, and Y, containing the width, height, and depth position of each foreground pixel, respectively. As a result, we can simply disregard the depth parameters contained in Y, focusing only on the two-dimensional shape information contained in the vectors X and Z.

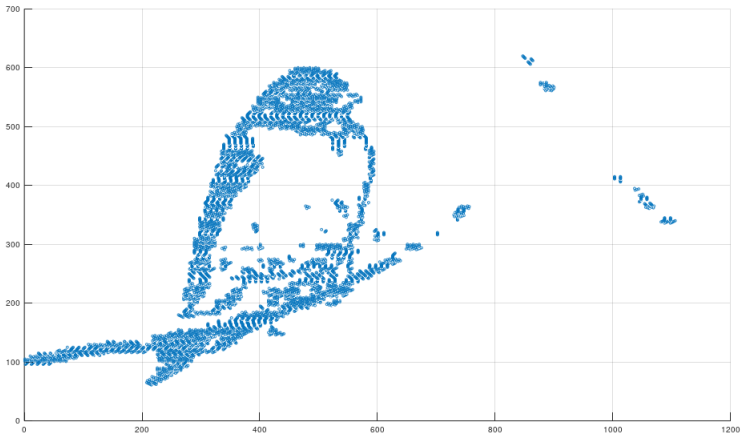

Figure 4: The image from Figure 4, as rendered by the algorithm.

[4]This is of course not the algorithm of choice if you're looking for a mathematically exact 3D rendering. But, it could be useful whenever a fast approximation of depth is required, and the only tool available is information generated by an ordinary two-dimensional camera.

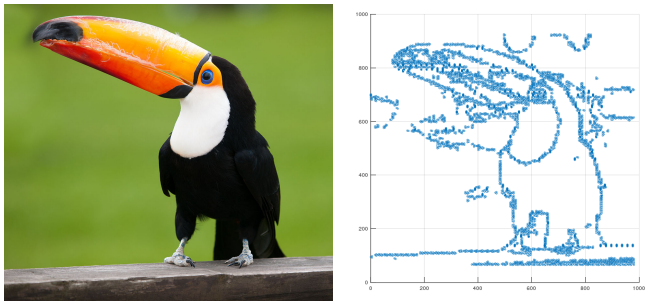

Figure 5: Another image of a bird (left), together with a rendering (right).

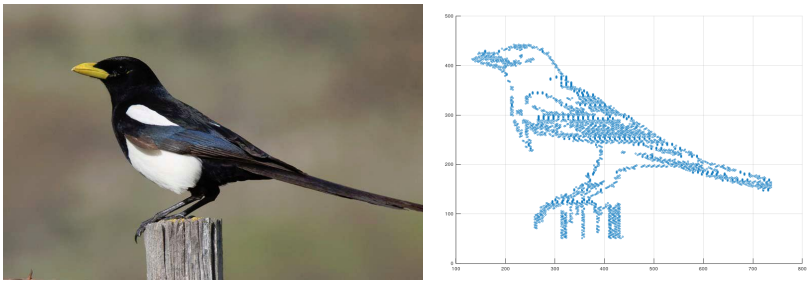

Figure 6: Another image of a bird (left), together with a rendering (right).