ChildSynth: Leveraging Synthetic Imagery for Automated Height Measurement in Malnutrition Prediction

David Berthiaume

May 2, 2024

1 Introduction

Predicting childhood malnutrition is a critical task in the field of public health. Malnutrition among children is still a significant public health problem in many developing countries. It is a primary cause of morbidity and mortality in children. Malnutrition is a condition that results from eating a diet in which one or more nutrients are insufficient, such that the diet causes health problems.

This paper proposes a novel approach to predict malnutrition in children using synthetic imagery. We leverage the power of synthetic data to help evaluate and train several deep-learning models to predict heights in children to support the prediction of malnutrition.

Obtaining data for training deep learning models is a challenging task. The data must be labeled and annotated, which can be time-consuming and expensive. Obtaining images of children to train a model to predict malnutrition is even more challenging. These images and height measurements must be taken in a controlled environment with trained personnel to ensure high accuracy. Furthermore, this data is highly sensitive since it involves images of children.

To combat these challenges, a synthetic data generator, ChildSynth, was developed to create images of children with pixel-perfect height measurements. This synthetic data generator can create an unlimited number of images of children with varying characteristics and pixel-perfect heights. The generated synthetic data can be used to evaluate and pre-train deep learning models to predict children's height.

2 ChildSynth

ChildSynth is a command-line program that uses procedural modeling and ray tracing to generate color images, depth maps, segmentation maps, keypoints, precise height measurements, and auxiliary information for synthetic children lying on a mat as viewed from different camera angles. The following example

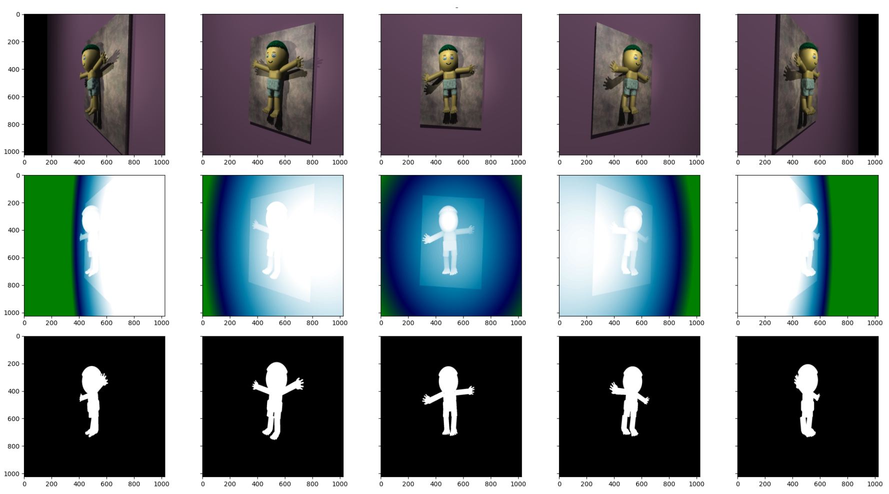

Figure 1: Example of a synthetic child generated by ChildSynth. The upper row consists of color images, the middle row consists of depth maps, and the bottom row consists of binary segmentation maps indicating whether each pixel is part of a child or part of the background.

command generates RGB images, depth maps, segmentation maps, and auxiliary text files with height measurements and other characteristics for 5 different camera angles for a total of 15 images.

```
python render_children.py --resy 512 --resx 512 --num_children 1
    --output_dir ./output
```

Figure 1 shows an example of a synthetic child generated by the above command.

The goal of ChildSynth is not to generate photorealistic images but to parametrically model children with infinite variations. Every element, the length, style, density, and color of hair, the facial characteristics, the skin and mat textures, even the size of each toe, is modeled parametrically and can be individually specified or sampled from various random distributions. Elements of the scene, including camera position, lighting, and image quality, are also parametrically modeled. One can specify over 60 different parameters to control how the children and scenes are rendered.

3 Approach

3.1 Rendering

ChildSynth renders Each parametrically modeled scene using ray tracing. The geometries in the scene are composed of spheres, cylinders, cubes, rounded cubes, planes, and blobs. Blobs themselves are composed of cubes, cylinders, and spheres and allow one to parametrically create smooth shapes, such as those found in a child's body. A smooth field is created from the multiple component objects within each bloc. Each component of a blob exerts a field in the space around it, and the sum of these fields determines the shape of the blob. When the combined field strength at any point in space exceeds a given threshold value, the surface at that point becomes part of the blob. These composite objects are beneficial for creating smooth shapes parametrically with fine-grained control over the individual features of the shape. Much of the complexity of the child's body is modeled using blobs, especially for the feet and hands (see figure 2).

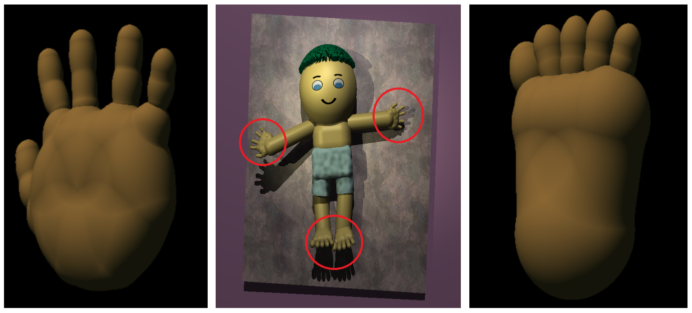

Figure 2: Hands and feet are rendered using blobs that consist of a complex arrangement of parametric spheres and cylinders. The strength and shape of each sphere and cylinder can be individually controlled to create a wide variety of shapes from the resulting field.

Each child is rendered from multiple view angles to provide a variety of perspectives. The camera is placed at different positions above the child to capture the child's body from multiple positions. By default, 5 different camera angles are used to render each child, from 30 to 150 degrees, in increments of 30 degrees, with 90 degrees being directly above the child, and 30 and 150 degrees being to the left and right of the child respectively. These multiple views allow one to predict the child's height with computer vision.

ChildSynth renders a depth image corresponding to each color image by storing the distance from the camera to the scene. The depth image and the color imagery are precisely coregistered.

Finally, ChildSynth renders a segmentation map for each child. The segmentation map is a binary image where each pixel is equal to 0 if it part of the background and 1 if it is part of the child. Along with the color and depth images, the segmentation map is helpful for training deep-learning models to predict the child's height.

3.2 Modeling

3.2.1 Parameters

ChildSynth can render millions of synthetic children, with no two being identical. It accomplishes this by sampling over 60 parameters from various probability distributions. For example, ChildSynth may place an optional spotlight around the child as a light source. The position of this spotlight is sampled from several normal distributions, with the light being most likely placed near the child. The color of this light source is sampled from three uniform distributions, one for each of the red, green, and blue intensities.

Several distributions are available in ChildSynth; all parameters can be optionally specified in the command line and support every probability distribution. Every parameter has a default distribution that is used when no distribution is explicitly provided.

3.2.2 Probability Distributions

This section provides a summary of the available probability distributions and their arguments. The following distributions are available in ChildSynth:

- **Constant**: The simplest distribution defines a single constant value that is selected with probability 1.

  Example usage:

  ```
          python render_children.py --smile_factor constant:0.7
  ```

- **Bernoulli**: The Bernoulli distribution defines a probability p that a binary random variable is equal to 1. This can be used to turn on or off features such as an individual light source, or the rendering of hari. Example usage: **bern:0.5**

  Example usage:

  ```
          python render_children.py --spotlight_1 bern:0.3
  ```

- **Uniform**: The uniform distribution is defined by two parameters, a and b, which are the minimum and maximum values of the distribution.

  Example usage:

  ```
          python render_children.py --hair_color_red unif
      :0.1,0.6
  ```

- **Normal**: The normal distribution is defined by two parameters, μ and σ, which are the mean and standard deviation of the distribution.

  Example usage:

  ```
          python render_children.py --head_size normal:0.8,0.2
  ```

- **Binomial**: The binomial distribution is used to sample an integer obtained from n independent Bernoulli samples with probability p. This returns the number of successes in these independent trials. To use this distribution, one specifies the number of trials n, and then the probability of success for each independent trial p.

  Example usage:

  ```
          python rener_children.py  --clothes_wrinkles binom
      :100,0.5
  ```

3.3 Colors

ChildSynth supports the specification of colors using the RGB color space where each of the red, green, and blue values lie in the range $[0, 1]$. Each channel is specified separately to allow maximum flexibility at the expense of more verbose command-line inputs. The following example specifies the color of hair with a red intensity of 1, a blue intensity of 0.25, and a green intensity sampled from a normal distribution with a mean of 0.2 and a standard deviation of 0.1.

```
python render_children.py --hair_color_red 1 --hair_color_blue 0.25
     --hair_color_green normal:0.5,0.1
```

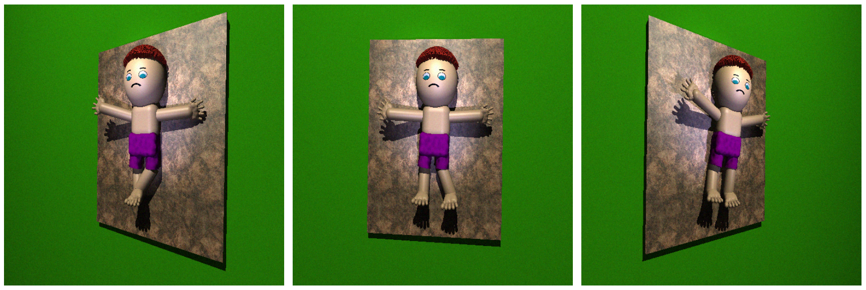

Figure 3: Hair color is specified with red, green, and blue intensities. The red and blue intensities are set to 1 and 0.25, respectively, while the green intensity is sampled from a normal distribution with a mean of 0.2 and a standard deviation of 0.1.

References