SmallDepthMask: Large DataSet Generation for Monocular Depth Estimation and Foreground Segmentation from few Internet Images

Anonymous CVPR 2021 submission

Paper ID ****

Abstract

Segmentation of the desired object along with depth estimation is useful in various applications like robotics and autonomous navigation. Any deep learning workflow to segment the desired foreground object in a scene require significant training data. The data generation process usually involve expensive hardware like RGB-D sensors, Laser Scanners or significant manual involvement. This paper presents a novel way to utilize only a small number of readily available png images with transparency for the foreground object, and representative background images from the internet and combine them to generate a huge dataset for deep learning utilizing current state of the art monocular depth estimation and segmentation techniques. Few example applications show the efficacy of the training data on detecting cattle on road for autonomous driving application, etc. The baseline models exhibit strong generalization to real scenarios.

1. Introduction

Expand the abstract with appropriate references. We need references for: 1. Monocular Depth 2. Image segmentation 3. Deep learning requiring large datasets 4. Generating data sets is cumbersome 5. Ours is novel work that uses few images to generate huge dataset that generalizes well Depth estimation and semantic segmentation are often used together in many vision tasks [10] like autonomous navitation of agents, augmented reality, self driving cars and other robotics applications. In all these application, identification of desired objects precisely in the scene and its depth estimate from the camera are crucial for safe and effective navigation. Modern RGB-D sensors like OAK-D are capable of simultaneously running advanced neural networks while providing depth from two stereo cameras and color information from a single 4K camera in the center. Deep learning based techniues using convolution neural nets have effective solution in both depth estimation and semantic segmentation. In general for high accuracy outcomes, a deep learning network is dependent on large training dataset availability. To gather such data itself incur high cost and time. For specific applications requiring several forground objects against variety of backgrounds become even more challenging in terms of simulating those scenarioes. Synthetic datasets using Virtual Reality have been proposed to that end.

Recent research indicates effective use of readily available images on the internet to curate training data. This paper introduces SmallDepthMask, a way to curate custom dataset containing millions of images by multiplexing desired foreground objects over representative background scenes, while also generating corresponding depth and foreground mask images. This significantly reduces the cost and time overheads. The authors also experiment by creating baseline models for several application contexts and show that the generated data successfully generalizes to detect relevant objcts in real scenes. Multiplexing, combined with random cropping, scaling and translation, make the data-generation fast and effective. With only 100 pair of background and foregroung images, the authors generate 4 million image triplets and effectively leverage existing SOTA models for depth estimation and semantic segmentation.

The main contributions of this paper are the following:

1. A novel effort to mix and match foreground and background images reducing the need for complex scene generation for data curation.

2. Curate large dataset to effectively train models for custom applications of detecting depth and mask for specific foreground objects over any target background, from a limited input of ready available internet images.

3. Combine image, depthmap and forground mask in a single dataset using current SOTA models for depth estimation and semantic segmentation.

4. To release curated dataset and the trained models making them publicly available. Researchers can use this

single dataset to do segmentation, train models to predict depth, or to predict both depth and mask.

Figure 1 represents a few representative examples of a dataset to help detect cattle on road, which is very common on Indian roads, leading to several accidents involving loss of life and property. The generated datasets and the trained models are publicly available.

2. Related Work

A depth image is an image channel in which each pixel relates to a distance between the image plane and the corresponding object in the RGB image. Monocular depth gives information about depth and distance and the Monocular Depth Estimation is the task of estimating scene depth using a single image[1]. Image Segmentation is the process of partitioning an image into multiple segements and it can be used for locating objects, boundaries [3]. RGBD image is a combination of a RGB image and its corresponding depth image[19].

Depth information is integral to many problems in robotics including mapping, localization and obstacle avoidance for terrestrial and aerial vehicles, autonomous navigation, and in computer vision, including augmented and virtual reality[11]. RGBD datasets usually collected using depth sensors, monocular cameras and LiDAR scanners are expensive and data collection is a time consuming job. The wellknown datasets for monocular 3D object detection are Context-Aware MixEd ReAlity (CAMERA), Objectron, Kitty3D, Cityscape3D, Synthia, etc. [] and these datasets have limitations like indoor only images, small number of training examples and sparse sampling.

To address the issues usage of expensive devices and small number of training examples, this paper proposes a technique to come up with a custom dataset by using existing accurate depth predictor models, like High Quality Monocular Depth Estimation via Transfer Learning(nyu.h5) [2].

A variety of RGBD datasets in which images are paired with corresponding depth maps(D) have been proposed through the years. Some of the frequenlty used RGBD datasets are the Kitti dataset [7], the Synthia dataset [14], Make3D dataset [15], NYU dataset [16]

The dataset Kiiti [7] is the well known RGB-D dataset collected using a vehicle equipped with a sparse Velodyne VLP-64 LiDAR scanner and RGB cameras, and features street scenes in and around the German city of Karlsruhe. The Primary application of this dataset involves perception tasks in the context of self-driving. Synthia [14] is a street scene dataset with depth maps of synthetic data, requiring domain adaptiation to apply to real world settings. Cityscapes [5] provides a dataset of street scenes, albeit with more diversity than KITTI. Sintel [12] is another synthetic dataset which mainly comprises of outdoors sences.

Megadepth [9] is a large-scale dataset of outdoor images collected from internet, with depth maps reconstructed using structure-from-motion techniques, but this dataset lacks in ground truth depth and scale. The RedWeb [18] dataset provide depth maps generated from stereo images which are freely available in large-scale data platforms such as Flicker. The datasets MegaDepth and RedWeb can be easily computed with the existing MVS methods.

Make3D [15] provides RGB and depth information for outdoor scenes. The NYUv2 dataset [16] is widely used for monocular depth estimation in indoor environments. The data was collected with a Kinect RGBD camera, which provides sparse and noisy depth returns. These returns are generally in-painted and smoothed before they are used for monocular depth estimation tasks. As a result, while the dataset includes sufficient samples to train modern machine learning pipelines, the "ground-truth" depth does not necessarily correspond to true scene depth.

Most of the existing datasets consists of indoor images, or outdoor images of city streets. For every specific application, like detecting animals roaming on roads for self driving or assisted driving cars, or people inside a room for autonomus room cleaners etc. researchers need to curate specific dataset to train relevant deep learning models. The present work makes the task of curating dtaset extremely simple and cost effective.

3. Dataset Generation Method

The curated dataset must have following objectives:

1. dataset which dedicatedly includes foreground object.
2. dataset should drive deep learning models and generalize.
3. dataset should provide accurate dense depth maps.
4. dataset should provide foreground mask
5. dataset can be stored offline or generated online during training phase dynamically

4. Method

4.1. Data Acquisition

The first step to curate data is to determine a target application scenario and thus determine the foreground object(s) and the representative background context. At the same time the dataset must have sufficient variability to include majority of the types and views that the trained deep network may see when deployed.

We propose to download or take RGB image of n foreground object(s) and m background images (we used $n =$

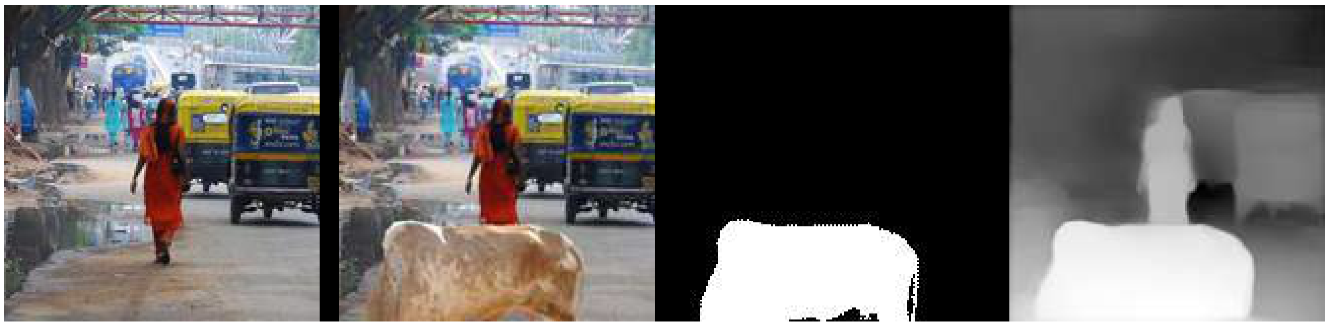

Figure 1. Sample Record which contains the background image, a cow overlayed on top of background, its mask and depth images

$m = 100$) balancing the types and views. For example, for cattle on roads dataset, we shose several cow, bull and calf types, individual or in group, sitting, standing or walking, and from various angles. Similarly for background, we chose backgrounds of streets, storefronts, main roads, highways, markets, railway tracks, landscapes, garbage piles etc. PNG images with transparency are readily available on the internet for almost any desired foreground object. Such images will easily allow to generate foreground mask from non-transparent pixels. If not, tools like GIMP [8], combined with deep learning forground extractors (Authors used remove.bg [?] that uses a combination of Image based techniques and DNN to separate foreground from background) can help generate the required PNG foreground images.

4.2. Multiplexing and Depth Generation

This step is to place each foreground object on to several background images generating a fg-bg image and the corresponding mask corresponding to the foreground placement and scale. Depth is computed from the fg-bg image via the model proposed by Ibraheem Alhashim et al. in their paper titled ”High Quality Monocular Depth Estimation via Transfer Learning” [2][1]. This model takes 448×448 size images as input, hence we resize all backfround images to this size while maintaining their aspect ratio.

The data generation process is completely online and produces one batch of images for training a deep model. By repeating one forgound object k times for each background image and repeating another k times with horizontally flipped version of the same foreground, one can generate $2kmn$ fg-bg images. For $k = 20$ and $n = m = 100$ this becomes $400,000$ fg-bg images. Algorithm 1 descrbes the data generation process

This work generates transparent foreground images and masks using GIMP [8] software and depth maps by using The main source images for the dataset are 100 scene images, and 100 images of objects.

A maximal random crop of $448X448$ without affecting the image aspect is done on background images. The object picked for this custom dataset creation is stray animals (mostly cows/bull/calf etc.). We have taken care to include single animals, group of animals, front, back and side poses of animals, all age group animals, and many coloured cows. Fig. 2 shows few of the sample scene and foreground images used for the creation of this dataset. From these 100 selected foreground objects, we have created 200 transparent objects. Several tools like Photoshop's magic wand, lasso tool, online resources to remove backgrounds[2], were used to create transparent foreground images. These 100 images are flipped horizontally to get 200 foreground images.

4.3. Data Curation and Processing

To create the main dataset of 400K images, we used the selected 100 background images and the 200 foreground images. The entire procedure followed to create the dataset is represented in the form of algorithm.

Fig. 3 shows the three outcomes of the above algorithm for a set of images.

4.3.1 fg_bg images

To generate fg_bg images, the foreground image is overlaid on background images randomly for 20 times. To place a foreground image on the background, a center point(x, y) in the range of 0 to 447 is randomly picked, and a scale between 0.3 to 0.6, which identifies the area overlaped by foreground image on the background image is also randomly picked. Next the foreground image is scaled and placed on top of background image centered at (x, y). Save this overlaid image with 224×224 resolution. As the number of foreground images are 200, the total number of overlaid images per background becomes 4000. By repeating the same procedure for all the 100 background images, the total number of fg on bg images becomes 400000. A set of

[1]source code for depth estimation model: https://github.com/ialhashim/DenseDepth/blob/master/DenseDepth.ipynb

[2]https://www.remove.bg/

Algorithm 1: Generate Dataset(*[bgimages]*, *[fgimages]*, k, b)

```
input  : $m$ Background Image paths, $n$ Foreground Image paths, multiplexing factor $k$, batch size $b$ must be multiple of $k$
output: Yield $2kmn$ fg-bg, mask and depth images in batches of size $b$
for offline use it creates 3 folders with fg_bg, mask and depth each having $2kmn$ images;
for $bg \leftarrow 1$ to $m$ do
    for $fg \leftarrow 1$ to $n$ do
        for $i \leftarrow 1$ to $k$ do
            $croppedbg \leftarrow$ take maximal random crop of $448 \times 448$ from $bg$ without affecting the aspect ratio;
            randomly pick a center point $(x, y)$ in range $[0, 447]$;
            randomly pick a scale in range $[0.3, 0.6]$ (ratio of area $fg$ covers $bg$);
            create $fg - bg$ image by resizing the $fg$ to scale and place it on top of $croppedbg$ centered at $x, y$ calculated;
            calculate binary mask from current placement of $fg$ by thresholding transparency channel. save $fg - bg$ image and mask add $fg - bg$ image to a batch;
        if $b$ new fg-bg images generated then
            run depth model on batch and save corresponding depth images
```

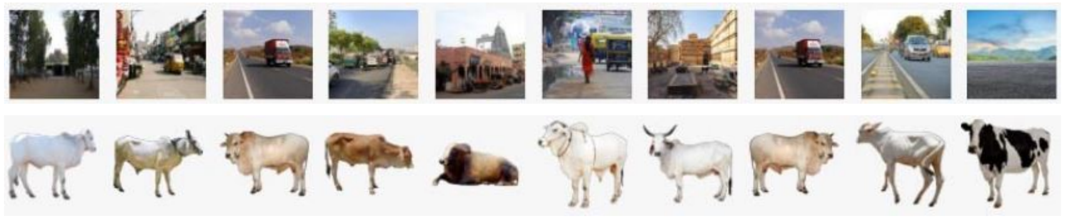

Figure 2. Scene and foreground object images

sample images after overlaying foreground on background are shown in Fig.

4.3.2 masks of fg_bg images

The mask is calculated for every fg_bg image by setting a binary image to transparency channel of foreground image. These 400000 images are also stored in the particular folder of the dataset and a set of sample masks of fg on bg images are shown in Fig.

4.3.3 depth maps of fg_bg images

We used nyu.h5 model for depth calcualtion from Depth estimation proposed by [2]. This model requires input images to be of 448x448 resolution and produces 224x224 size depth image. A set of sample depth images are shown in fig. 3.

4.3.4 Directory Structure

5. Example Applications

5.1. Detecting cattle on roads

Describe importance and if possible related work and how deep learning is not yet applied. Give example images

We can include adaptive placement of foreground to bemore realistic and also occlusion handling. However norte that this is only preliminary The segmentation I used gave us road, skiy, tree etc. If I remember correctly, I used the bottommost row in image that have a threshold number of sky pixels. And based on the max depth and span of the non sky part determined a formula to scale. The cattle was placed on ground area only Finally, any objects that come on top of cow region in segmentation are put on top in the generated image.

I can formally write the algo tomorrow.

5.1.1 Baseline Model

Discuss the model with loss function used and results along with generalized results

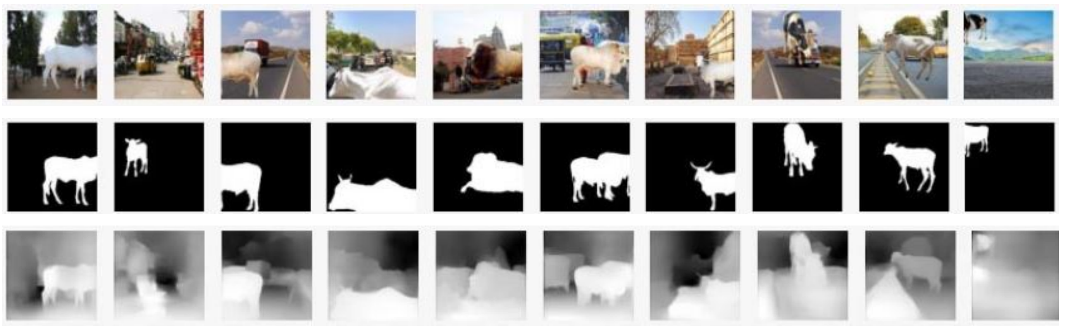

Figure 3. Three images resulted from the algorithm used. (top) A scene image on which a foreground object is positioned at random location with random scale, (middle) respective mask for the scene image, and (bottom) calculated depth by using a model.

5.2. Example 2

5.3. Example 3

5.4. Data Statistics

Every image (fgbg, mask, depth) in the dataset are of size 224 X 224. The distribution of fgbg, mask and depth values for XYZ dataset are shown in fig. The dataset has 400000 records. A train-test-split of (70-30) gives a training set size of 280000 and 120000. A sample record in the dataset contains paths to all the images as shown below.

('./data/bgimages/bgimg099.jpg', './data/out2/images/fgbg392483.jpg', './data/out2/masks/mask392483.jpg', './data/out2/depth/fgbg392483.jpg')

6. Experiments

In this section, we provide a baseline for monocular depth estimation and segmentation on the XYZ dataset. The state-of-the-art models for image segmentation are variants of U-Net and fully convolutional networks (FCN)[6]. long skip connections are used to skip features from the contracting path to the expanding path in order to recover spatial information lost during downsampling [20]. Short skip connections can be used to build deep FCNs. By using both long and short skip connections we proposed a model following U-Net architecture. This model has one encoder and two decoders, each meant for mask and depth prediction.

6.1. Model

We have designed a model with one encoder and two decoders with skip connections. The architecture is shown in Fig 4. The total no.of parameters of this model are 5,525,568. We have trained this model on the entire XYZ dataset from scratch. During training the network is trained with the batch size of 64 for 10 epochs using SGD optimizer [4]. We have used OneCycleLR scheduler [17] with a maximum Learning rate of 0.1. This made the initial learning rate as 0.0099. The Deep Convolutional Neural Networks encoder is fed with a image (224 X 224) and the first decoder outputs a mask image and and second decoder outputs a depth image. To reduce overfitting[13], this work employed Random Rotation, Random Grayscale, Color Jitter, random horizontal flips and random chaneel swaps for data augmentation.

The Loss is calculated with the help of L1 loss and Structural Similarity (SSIM) at both the decoders. We have also employed regularization for weight penality.

For training our network with two decoders, we defined the same loss function L between y and $\hat{y}$ as the weighted sum of two loss function values.

$$L(y,\hat{y}) = \lambda L_{term1}(y,\hat{y}) + (1-\lambda)L_{term2}(y,\hat{y}) \qquad (1)$$

The first loss term $L_{term1}(y,\hat{y})$ is the point-wise L1 loss defined on mask values at the first decoder and on depth values at the second decoder.

$$L_{term1}(y,\hat{y}) = \frac{1}{n}\sum_{x=1}^{n} \mid y_i - \hat{y}_i \mid \qquad (2)$$

we have also used weight decay... do we need to add it in the form of euation???????????

The second loss term $L_{term2}(y,\hat{y})$ uses a commonly used metric for image reconstruction task i.e., SSIM. Many recent tood depth prediction CNNs employed this metric. The loss term is redefined as shown in equation as SSIM has an upper bound of one.

$$L_{term1}(y,\hat{y}) = \frac{1 - SSIM(y,\hat{y})}{2} \qquad (3)$$

Different weight parameters λ were tried and we have ended with a value $\lambda = 0.84$. The final loss function is as follows.

$$L(y,\hat{y}) = 0.84 * L_{term1}(y,\hat{y}) + 0.16 * L_{term2}(y,\hat{y}) \qquad (4)$$

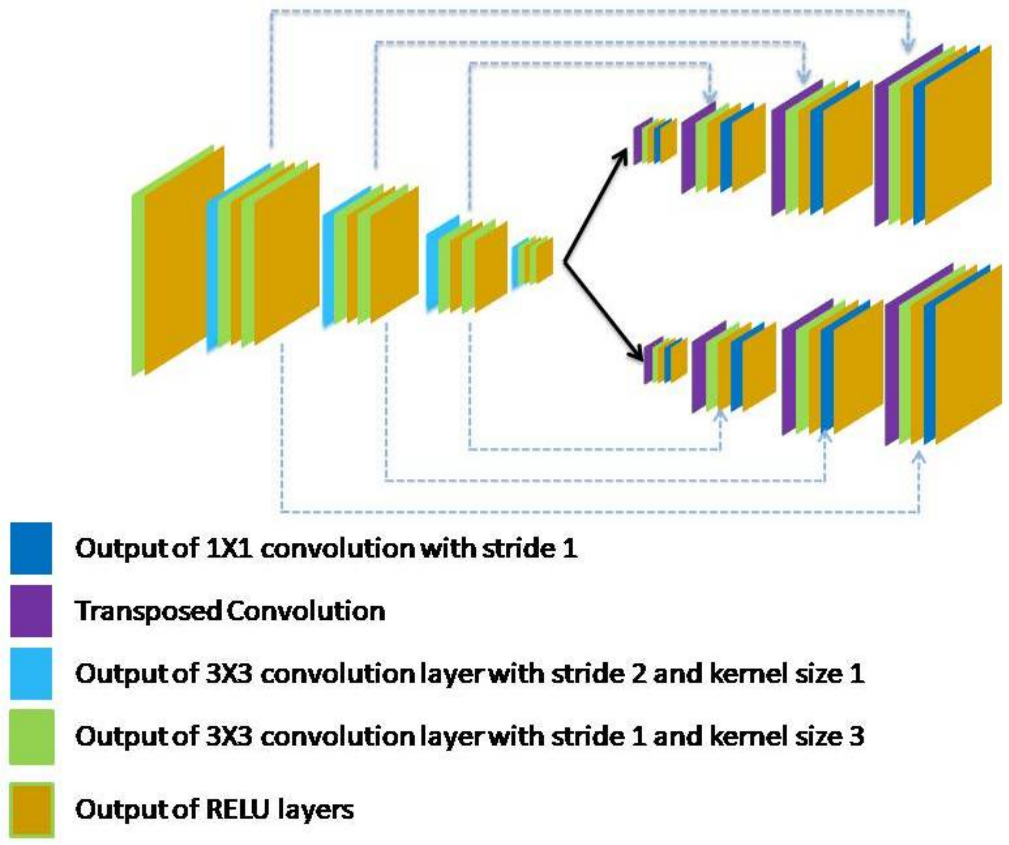
Output of 1X1 convolution with stride 1
Transposed Convolution
Output of 3X3 convolution layer with stride 2 and kernel size 1
Output of 3X3 convolution layer with stride 1 and kernel size 3
Output of RELU layers

Figure 4. Network Architecture

6.2. Evaluation

6.3. Analysis

7. Conclusion

8. Acknowledgement

This paper and the research behind it would not have been possible without the exceptional support and computing facilities of my Institution, Vishnu Institute of Technology. Mention TSAI