

IIIT HYDERABAD

COMPUTER VISION

Team Name - Deepview

---

# Instance Level Salient Object Segmentation

---

ANIKET JOSHI - 20161166  
PRAKYATH MADADI - 20161236  
VASHIST MADIRAJU - 20161222



INTERNATIONAL INSTITUTE OF  
INFORMATION TECHNOLOGY

---

HYDERABAD

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Problem Statement . . . . .	2
1.2	Motivation . . . . .	2
<b>2</b>	<b>Datasets</b>	<b>2</b>
<b>3</b>	<b>Training</b>	<b>3</b>
<b>4</b>	<b>Proposed Approach</b>	<b>3</b>
<b>5</b>	<b>Method</b>	<b>4</b>
5.1	MSRNet Architecture . . . . .	4
5.1.1	Bottom-Up Network . . . . .	4
5.1.2	Top-Down Network . . . . .	5
5.1.3	Implementation . . . . .	5
5.1.4	MSRNet-Results . . . . .	7
5.2	MCG . . . . .	7
5.2.1	MCG Results . . . . .	8
5.3	MAP Based Optimisation . . . . .	8
5.3.1	MAP Based Optimisation Results . . . . .	8
5.4	Conditional Random Fields (CRF) . . . . .	9
5.4.1	Unary Potential . . . . .	9
5.4.2	Pairwise Potential . . . . .	10
5.4.3	CRF Results . . . . .	11
<b>6</b>	<b>Results</b>	<b>11</b>

**Instance-Level Salient Object Segmentation[1]**  
**Github Link:** [Instance Level Salient Object Segmentation](#)

## Introduction

Image saliency detection has recently witnessed rapid progress due to deep convolutional neural networks. However, none of the existing methods is able to identify object instances in the detected salient regions. In this paper, we present a salient instance segmentation method that produces a saliency mask with distinct object instance labels for an input image.

## Problem Statement

The aim is to find a salient instance segmentation method that produces a saliency mask with distinct object instance labels for an input image. Most of the current segmentation methods are only designed to detect pixels that belong to any salient object, i.e. a dense saliency map, but are unaware of individual instances of salient objects.

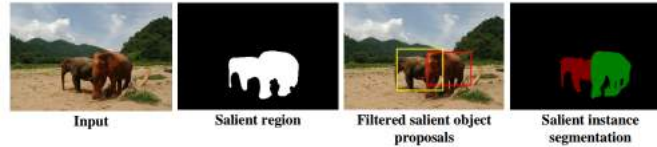


Figure 1: Instance Level Segmentation

## Motivation

Salient object detection attempts to locate the most noticeable and eye-attracting object regions in images. It is a fundamental problem in computer vision and has served as a pre-processing step to facilitate a wide range of vision applications. The proposed method extracts state of the art Saliency and Contour maps, as well as being able to identify individual object instances.

## Datasets

As salient instance segmentation is a completely new problem, no suitable dataset exists. Thus we have used a mixture of the standard datasets. They are:

- CSSD
- ECSSD
- DUT-OMRON

We have compiled a dataset consisting of around 6000 images by combining the above datasets.

## Training

During the training phase, we divided the dataset of 6000 images into 3 sets comprising of 3000 images for train data, 1000 images for validation and other 2000 images for testing. We used Adam’s optimizer instead of the classical stochastic gradient descent procedure to update network weights iteratively based on training data. The learning rate was kept to 0.00001 and binary cross entropy loss was used. The steps per epoch were kept 1000 and the batch size was kept at 4 as with higher batch size, we were getting memory error as the model is very big. The best weights based on validation loss was stored in every 5 epochs. Total number of epochs was kept to 2000.

## Proposed Approach

We can decompose the salient instance segmentation task into the following three sub-tasks.

1. Estimating binary saliency map.
2. Detecting salient object contours.
3. Identifying salient object instances (Salient instance generation and salient instance refinement).

We build a deep multi-scale refinement network and use it for salient region detection and salient object contour detection. Then, MCG (Multiscale combinatorial grouping) algorithm is used to find object proposals from the salient object contours. We generate a fixed number of salient object proposals on the basis of the results of salient object contour detection and further apply a subset optimization method to reduce the number of object proposals. Finally, the results from the previous three steps are integrated in a CRF model to generate the final salient instance segmentation.

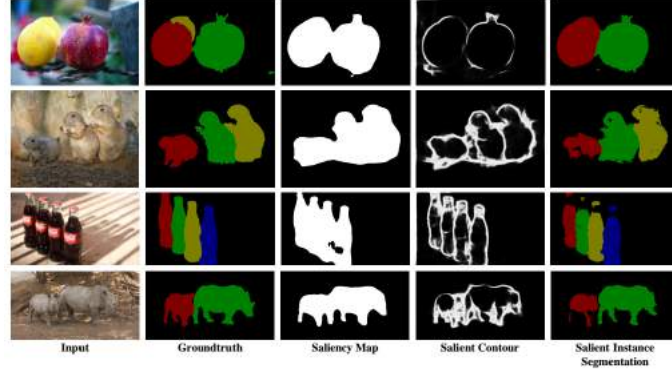


Figure 2: Results obtained by the Author

## Method

We formulate both salient region detection and salient object contour detection as a binary pixel labeling problem. Since salient objects could have different scales, we propose a multiscale refinement network (MSRNet). MSRNet is composed of three refined VGG network streams with shared parameters and a learned attentional model for fusing results at different scales.

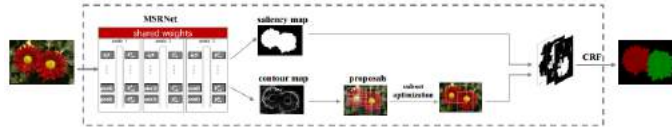


Figure 3: Pipeline

## MSRNet Architecture

As we can conclude from the properties of the above networks, a network should consider both bottom-up and top-down information propagation and output a label map. We want the saliency map and contour size to have the same resolution as the input image. Thus the output should be a label map of same resolution as the input image.

### Bottom-Up Network

- We use a modified VGG network for this with additional Convolutional layers for this.

- Takes low level features as input, such as colours and texture, and propagates it up through the layers.
- Information from an input image needs to be passed from the bottom layers up in a deep network before being transformed into high-level semantic information.

## Top-Down Network

- We use a top down model to use and incorporate high level semantic information.
- It is propagated from the top layers further down, and is integrated with low-level information obtained from the intermediate stages of the Bottom-up network.
- This integration of high level information with low level features, results in high precision contour detection results.

## Implementation

To implement the MSRNet we have used Keras-Tensorflow libraries. Results of the implementation and some particular layers are given below.

1. The output resolution of the transformed VGG network is 1/8 of the original input resolution i.e **output at fc7 should be of 1/8 resolution as input image.**

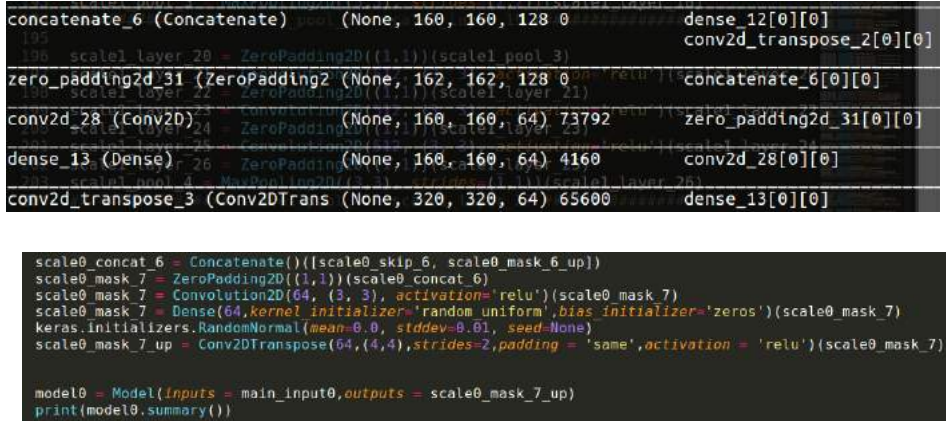
```
max_pooling2d_5 (MaxPooling2D) (None, 40, 40, 512) 0
zero_padding2d_19 (ZeroPadding2D) (None, 64, 64, 512) 0
conv2d_14 (Conv2D) (None, 40, 40, 1024) 4719616
dropout_1 (Dropout) (None, 40, 40, 1024) 0
conv2d_15 (Conv2D) (None, 40, 40, 1024) 1649600
dropout_2 (Dropout) (None, 40, 40, 1024) 0

scale0_pool_5 = MaxPooling2D((3,3), strides=(1,1))(scale0_layer_34)
#####scale0_pool_5 = scale0_layer_35#####

scale0_layer_36 = ZeroPadding2D((12,12))(scale0_pool_5)
scale0_layer_37 = Convolution2D(1024, (3, 3), activation='relu',dilation_rate=(12,12))(scale0_layer_36)
scale0_layer_38 = Dropout(rate = 0.5, noise_shape=None, seed=None)(scale0_layer_37)
scale0_layer_39 = Convolution2D(1024, (1, 1), activation='relu')(scale0_layer_38)
scale0_fc7 = Dropout(rate= 0.5, noise_shape=None, seed=None)(scale0_layer_39)
```

Figure 4: Output at fc7 is 1/8 resolution as input image (320x 320)

2. We use refinement modules to combine the top-down and bottom-up information.
  - We integrate a “refinement module”  $\mathbf{R}$  to invert the effect of each pooling layer and double the resolution of its input feature map if necessary.
  - Each refinement module  $\mathbf{R}_i$  takes as input the output feature map  $\mathbf{F}_{itd}$  of the refinement module in the top-down pass before.
  - The output feature map  $\mathbf{F}_{ibu}$  of the aforementioned extra convolutional layer attached to the corresponding pooling layer in the bottom-up pass.
  - The refinement module  $\mathbf{R}_i$  works by first concatenating  $\mathbf{F}_{itd}$  and  $\mathbf{F}_{ibu}$  and then feeding them to another  $3 \times 3$  convolutional layer with 64 channels.
  - The final output of the refinement stream is a probability map with the same resolution as the original input image.



```

concatenate_6 (Concatenate)      (None, 160, 160, 128) 0      dense_12[0][0]
conv2d_transpose_2[0][0]
zero_padding2d_31 (ZeroPadding2D) (None, 162, 162, 128) 0      concatenate_6[0][0]
conv2d_28 (Conv2D)                (None, 160, 160, 64) 73792    zero_padding2d_31[0][0]
dense_13 (Dense)                  (None, 160, 160, 64) 4160     conv2d_28[0][0]
conv2d_transpose_3 (Conv2DTranspose) (None, 320, 320, 64) 65600    dense_13[0][0]

scale0_concat_6 = Concatenate()([scale0_skip_6, scale0_mask_6_up])
scale0_mask_7 = ZeroPadding2D((1,1))(scale0_concat_6)
scale0_mask_7 = Convolution2D(64, (3, 3), activation='relu')(scale0_mask_7)
scale0_mask_7 = Dense(64, kernel_initializer='random_uniform', bias_initializer='zeros')(scale0_mask_7)
keras.initializers.RandomNormal(mean=0.0, stddev=0.01, seed=None)
scale0_mask_7_up = Conv2DTranspose(64, (4, 4), strides=2, padding='same', activation='relu')(scale0_mask_7)

model0 = Model(inputs = main_input0, outputs = scale0_mask_7_up)
print(model0.summary())

```

Figure 5: Output dimension, same as the input Dimension(320 x 3)

## MSRNet-Results

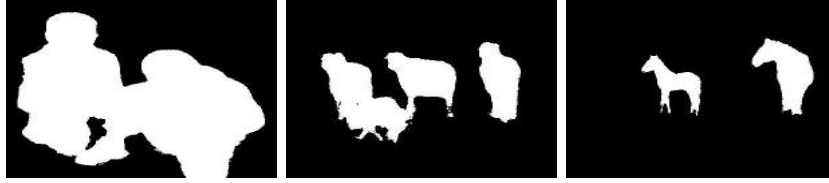


Figure 6: Saliency Maps

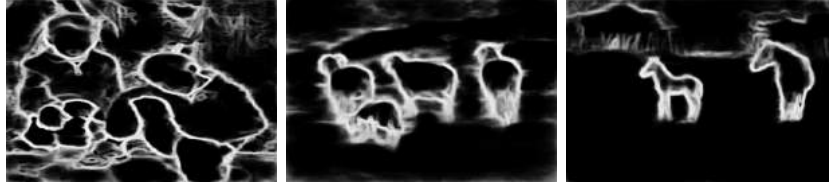


Figure 7: Contour Maps

## MCG

Given the salient object contours, generated from the *MSRNet*, we use *MCG*[3] to generate object proposals. *MCG* is a unified approach for bottom-up hierarchical image segmentation and object candidate generation. We replace the contour detector *gPb* in MCG with our MSRNet based salient object contour detection.

The procedure is listed below:

- Given an input image, we first generate four salient object contour maps (three from scaled versions of the input and one from the fused map).
- Each of these four contour maps is used to generate a distinct hierarchical image segmentation represented as an ultrametric contour map (UCM).
- These four hierarchies are aligned and combined into a single hierarchical segmentation, and a ranked list of object proposals are obtained.



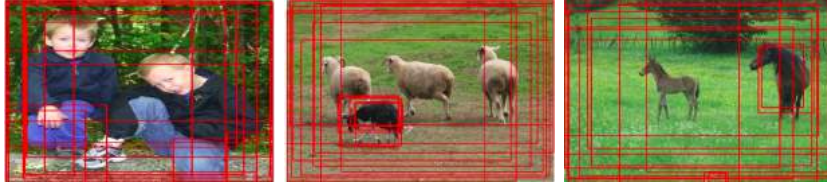


Figure 8: Proposals

## MCG Results

### MAP Based Optimisation

Given the set of initially screened salient object proposals, we further apply a **MAP**[2] based subset optimization method to produce a compact set of object proposals.

- These proposals tend to be highly overlapping and noisy.
- Based on the *Maximum a Posteriori* principle, a novel subset optimization framework is used to generate a compact set of detection windows out of noisy proposals.
- The number of remaining object proposals in the compact set forms the final number of predicted salient object instances in the image.
- Each remaining salient object proposal is a detected salient instance.

We can then use these proposals to generate the final instance segmentation using CRF based optimization.

### MAP Based Optimisation Results

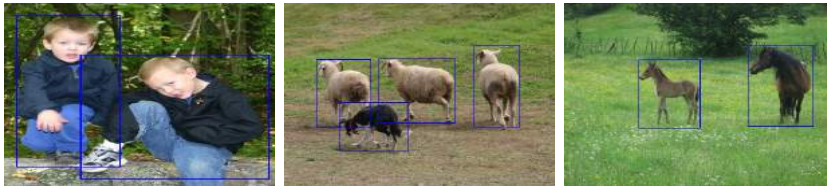


Figure 9: Best Proposals

## Conditional Random Fields (CRF)

We integrate the saliency map generated from the MSRNet and the object proposals generated from MCG, we use a fully connected Conditional Random Field (CRF) model, which generates the final instance segmentation result. It helps remove discrepancies between the union of all detected salient instances and the union of all detected salient regions.

The CRF model helps optimize the pixel labelling problem ensuring smoothness along the image. It optimizes an energy function which considers the *Unary* and *Pairwise* potentials of the pixels. The Unary potential of the pixel is determined by building a probability map based on the object proposals and the saliency map. The Pairwise potential is obtained based on the pixel neighbourhood.

### Unary Potential

The Unary potential is the negative logarithm of the pixel label probability map generated from the proposals and the saliency map.

$$U_i = -\log(P_i)$$

- Let the number of object proposals generated in the previous step is  $K$ .
- The background is considered as the  $K+1^{st}$  class, and the salient instance segmentation is treated as a multi-class labeling problem.
- At the end, every pixel is assigned with one of the  $K+1$  labels using a CRF model.

### Foreground pixel (determined from the saliency map)

- If a salient pixel is covered by a single detected salient instance, the probability of the pixel having the label associated with that salient instance is 1.
- If a salient pixel is not covered by any detected salient instance, the probability of the pixel having any label is  $1/K$ .
- If a salient pixel is covered by  $t$  overlapping salient instances, the probability of the pixel having a label associated with one of the  $t$  salient instances is  $1/t$ .

## Background pixel

- If a background pixel is not covered by any salient instances, then the probability of the pixel being a background pixel is 1.
- If a background pixel is covered by  $t$  overlapping salient instances, the probability of the pixel having a label associated with one of the  $t$  salient instances is  $1/(t+1)$  and the probability of the pixel having the background label is also  $1/(t+1)$ .

## Pairwise Potential

The pixel neighbourhood decides the pairwise potential of the pixel. The pairwise potential consists of two Gaussian kernels, which help enforce smoothness in the image.

- The first kernel depends on pixel positions (p) and pixel intensities (I), and encourages nearby pixels with similar colors to take similar salient instance labels.
- The second kernel only considers spatial proximity, the pixel positions (p).

$$\theta_{ij} = \mu(x_i, x_j) \left[ \omega_1 \exp \left( -\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2} \right) + \omega_2 \exp \left( -\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2} \right) \right]$$

$$\text{where } \mu(x_i, x_j) = 0 \text{ if } x_i = x_j \\ = 1 \text{ otherwise}$$

- The hyperparameters control the scale of the kernels. Higher the value of the parameters, greater the effect of the kernel on the energy of the function.

The energy function (used by CRF), which needs to be optimized is:

$$E(x) = - \sum_i \log P(x_i) + \sum_{i,j} \theta_{ij}(x_i, x_j)$$

## CRF Results

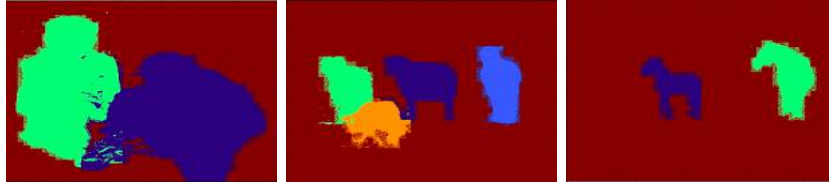
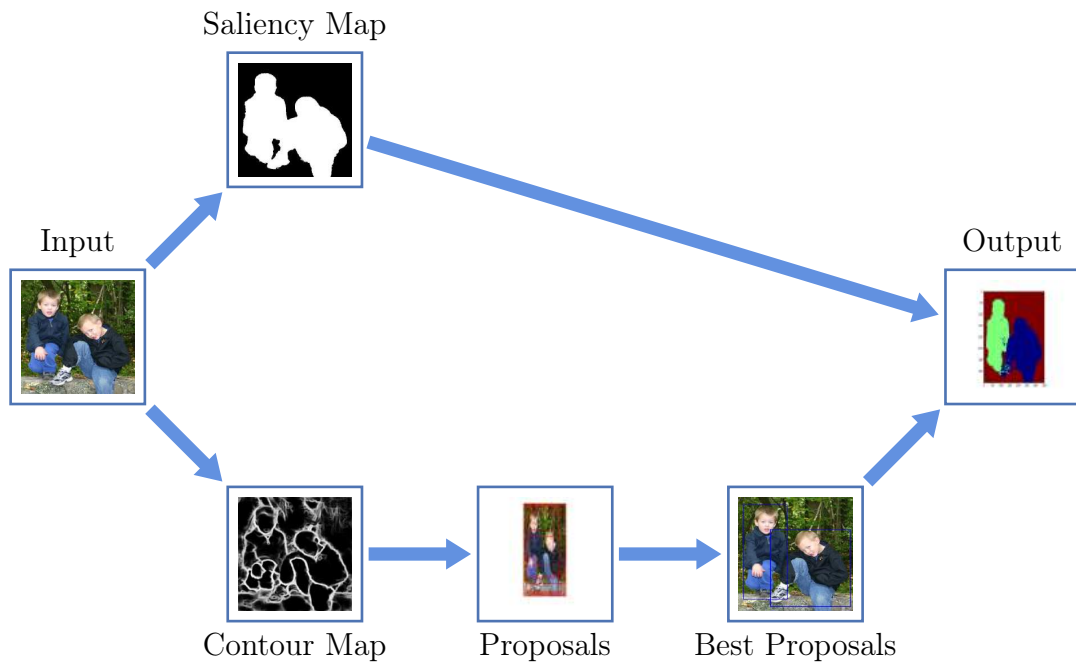


Figure 10: Instance Segmentation

## Results

So in conclusion we follow the following steps:

- Use MSRNet to generate the Saliency Maps and Contour Maps.
- Use MSRNet in MCG to generate object proposals
- Do MAP based optimization to get best proposals
- Generate Instance Proposals using CRF Model



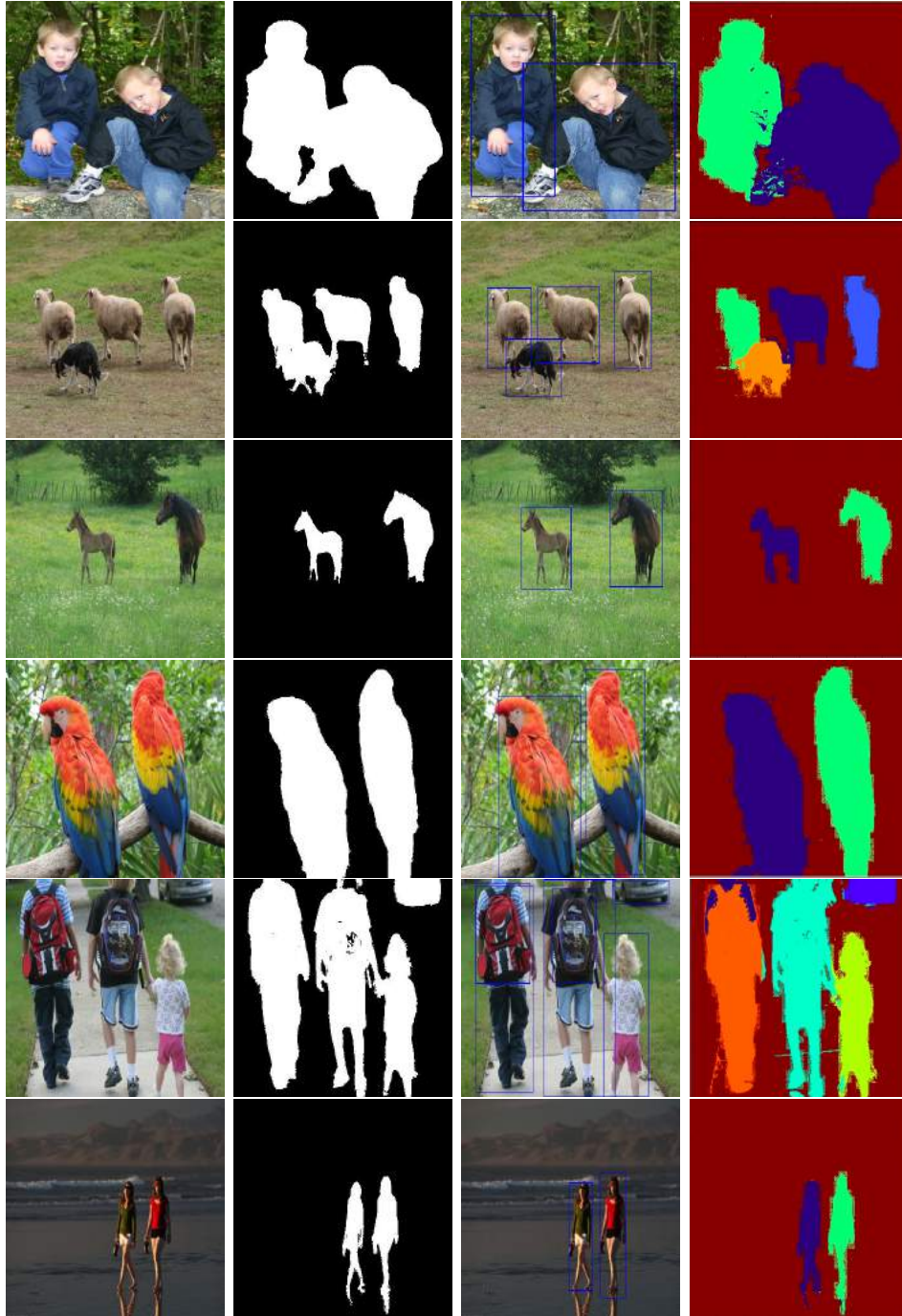


Figure 11: Our Results

## References

- [1] Liang Lin Guanbin Li Yuan Xie and Yizhou Yu. “Instance-Level Salient Object Segmentation”. In: *CVPR*. 2017.

- [2] Z. Lin X. Shen B. Price J. Zhang S. Sclaroff and R. Mech. “Unconstrained salient object detection via proposal subset optimization”. In: *CVPR*. 2016.
- [3] Jonathan T. Barron Jordi Pont-Tuset Pablo Arbeláez and Ferran Marquesu. “Multiscale Combinatorial Grouping for Image Segmentation and Object Proposal Generation”. In: *CVPR*. 2014.