Extract Human-Object Interaction to 3DMesh

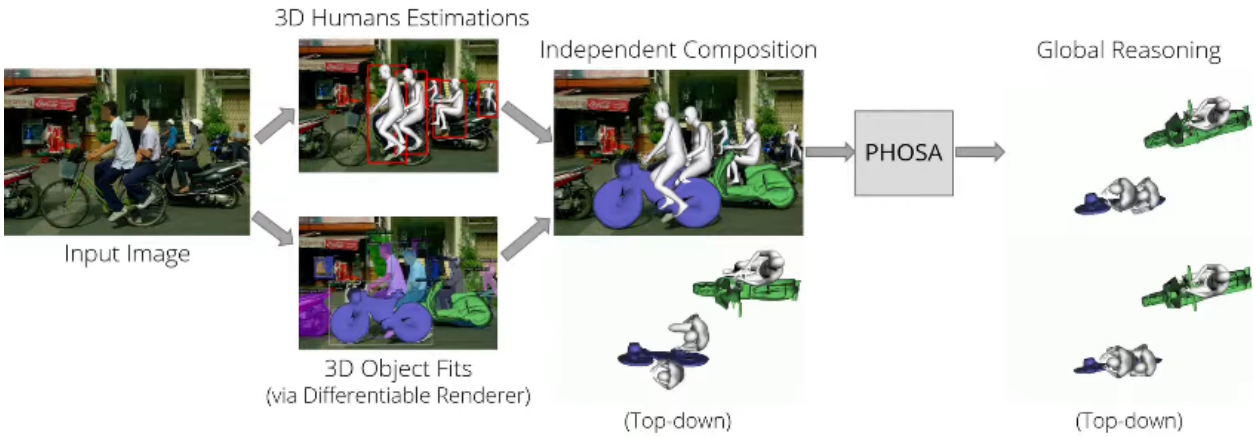
Abstract

We choose "**Perceiving 3D Human-Object Spatial Arrangements from a Single Image in the Wild**" paper from **Facebook AI Research,** which concentrates on extracting objects interactions without any scene- or object-level 3D supervision. We aim for providing a 2D image as input and outputting reasonable human-object interaction 3D scenes through a FLASK API.

Table of Contents

1. Introduction

We present PHOSA, Perceiving Human-Object Spatial Arrangements, an approach that recovers the spatial arrangements of humans and objects in 3D space from a single image by reasoning about their intrinsic scale, human-object interaction, and depth ordering. Given the input image (top left), we show two possible interpretations of the 3D scene that have similar 2D projections (bottom left). Using priors of humans, objects, and their interactions, our approach is able to recover the more reasonable interpretation (bottom row).



2. Overview of our method, PHOSA

Given an image, we first detect instances of humans and objects. We predict the 3D pose and shape of each person and optimize for the 3D pose of each object by fitting it to a segmentation mask. Then, we convert each 3D instance in its own local coordinate frame into world coordinates using an intrinsic scale. Using our Human-Object Spatial Arrangement optimization, we produce global consistent output, as shown here. Our framework produces plausible reconstructions that capture realistic human-object interaction, preserve depth ordering, and obey physical constraints.

building such tools by learning the natural size distributions of object categories without any supervision. Our underlying thesis, bourne out by experiment, is that contextual cues arising from holistic processing of human-object arrangements can still provide enough information to understand objects in 3D.

3. Related Work

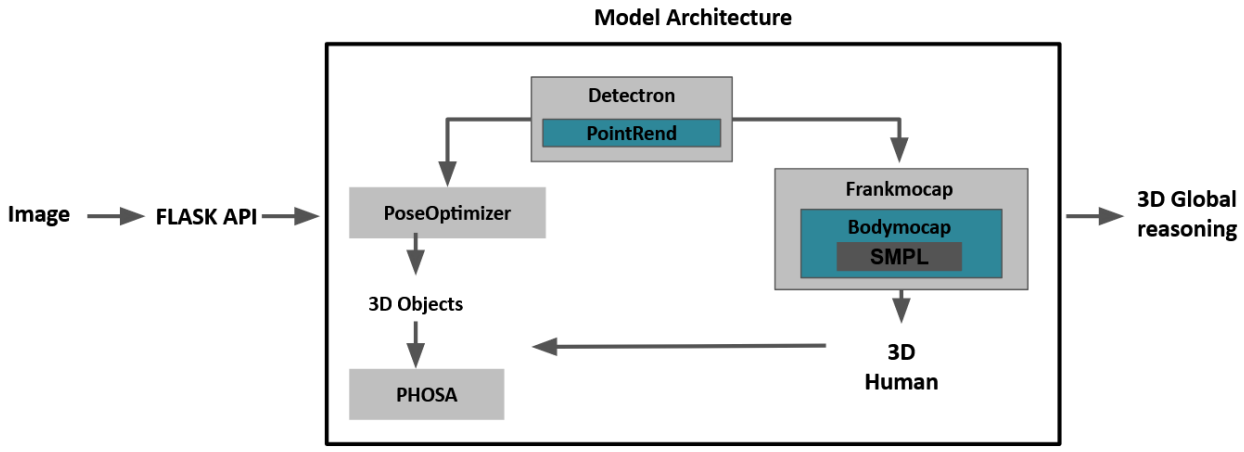
3D human pose and shape from a single image. Recovering the 3D pose and shape of a person from a single image is a fundamentally ambiguous task. As such, most methods employ statistical 3D body models with strong priors on shape learned from large-scale 3D scans and with the known kinematic structure to model the articulation

4. Method

Our method takes a single RGB image as input and outputs humans and various categories of objects in a common 3D coordinate system. We begin by separately estimating 3D humans and 3D objects in each predicted bounding box provided by an object detector. We use a state-of-the-art 3D human pose estimator to obtain 3D humans in the form of a parametric 3D human model (SMPL), and use a differentiable renderer to obtain 3D object pose (6-DoF translation and orientation) by fitting 3D mesh object models to predicted 2D segmentation masks. The core idea of our method is to exploit the interaction between humans and objects to spatially arrange them in a common 3D coordinate system by optimizing for the per-instance intrinsic scale, which specifies their metric size. In particular, our method can also improve the performance of 3D pose estimation for objects by exploiting cues from the estimated 3D human pose.

6. System Architecture

Flask API receives an image captured from the wild then the process of the model starts working starting from detectron to predict human instance segmentation masks and object detection which contains pointrend.

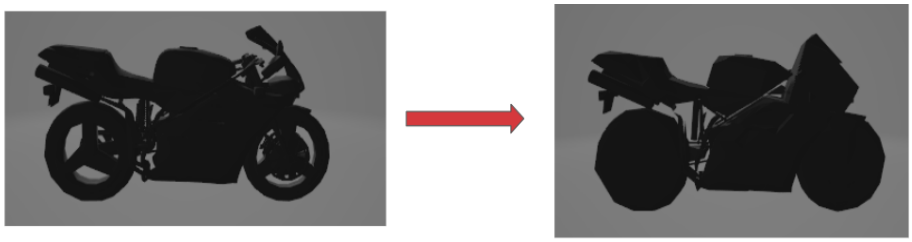


7. Methodology

6.1 Data acquisition & Preparation:

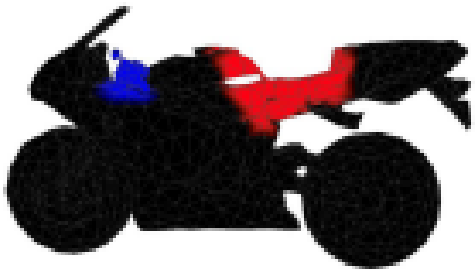
- COCO is large-scale object detection, segmentation, and captioning dataset with more than 331k images.
- Contains challenging images of humans interacting with everyday objects obtained in uncontrolled settings.

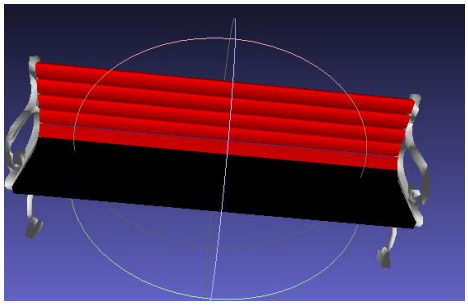
- So now we need to obtain and prepare these meshes for each category and prepare them using meshlab.
- All the meshes are pre-processed to be watertight and are simplified to make the optimization more efficient.
    - For the pre-processing, we first fill in the holes of the raw mesh models (e.g. the holes in the wheels or tennis racket) to make the projection of the 3D models consistent with the silhouettes obtained by the instance segmentation algorithm.



    - Finally, we reduce the number of mesh vertices using MeshLab.

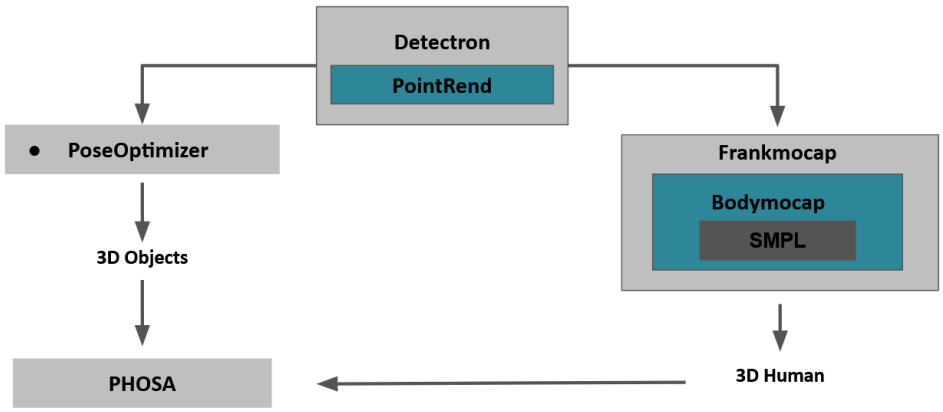
- The basic idea behind the model performance is that the model exploits the interaction areas between humans and each object of the previously listed. So we had to label the interaction areas manually on the meshes for each new class using Meshlab.





6.2 Model Architecture:

a. **Detectron:** predict human instance segmentation masks and object detection.
b. **Frankmocap** : for Human Pose Estimator to predict 3D human meshes.
c. **PoseOptimizer**: Find the optimal poses of an object based on object instances segmented from Detectron.
d. **PHOSA:** Optimizes the human-object interaction based on some losses
e. **Neural_rendrer**: For 3D meshes representation.



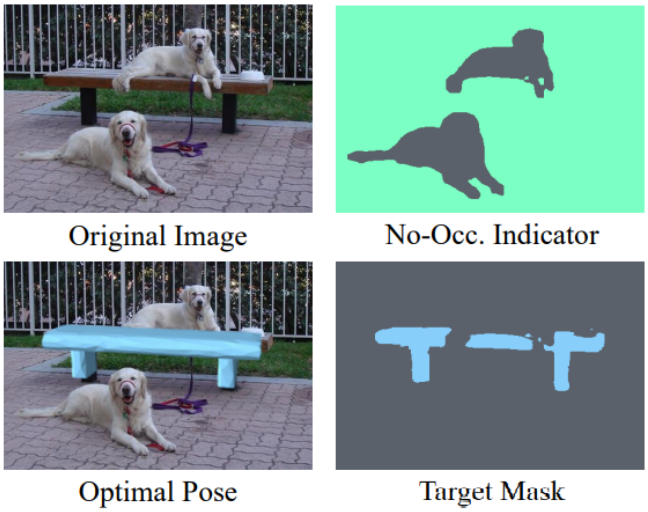
6.3 Evaluation

In this section, we provide quantitative and qualitative analysis on the performance of our method on the COCO-2017 dataset. We focus our evaluation on 8 categories: baseball bats, benches, bicycles, laptops, motorcycles, skateboards, surfboards, and tennis rackets. These categories cover a significant variety in size, shape, and types of interaction with humans.

Objective function to optimize 3D spatial arrangements. Our objective includes multiple terms to provide constraints for interacting humans and objects:

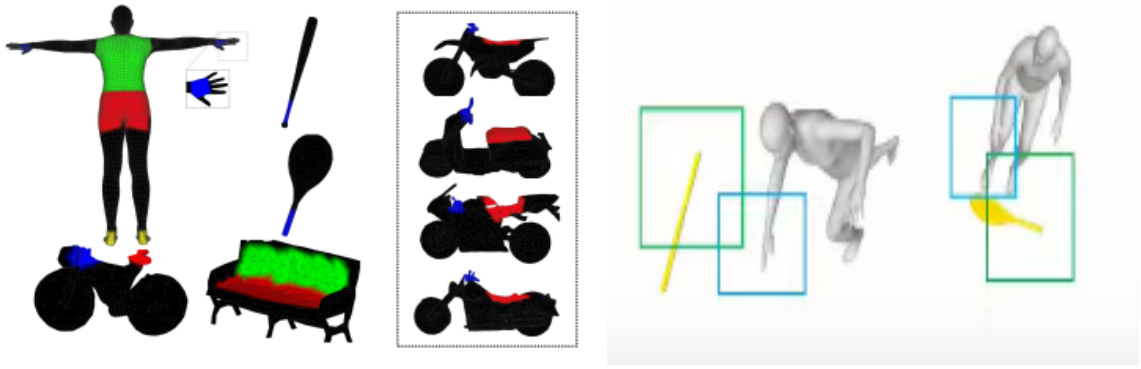
$$L = \lambda_1 L_{\text{occ-sil}} + \lambda_2 L_{\text{interaction}} + \lambda_3 L_{\text{scale}} + \lambda_4 L_{\text{depth}} + \lambda_5 L_{\text{collision}}. \quad (5)$$

- **Occlusion-aware silhouette loss:**
  For optimizing object pose. Given an image, a 3D mesh model, and instance masks, our occlusion-aware silhouette loss finds the 6-DoF pose that most closely matches the target mask (bottom right).



- **Human-object interaction loss:**
  - A part labeling approach an inspiration taken from PROX
  - Taking the contact regions on the human body and on each object mesh to encode interaction parts.

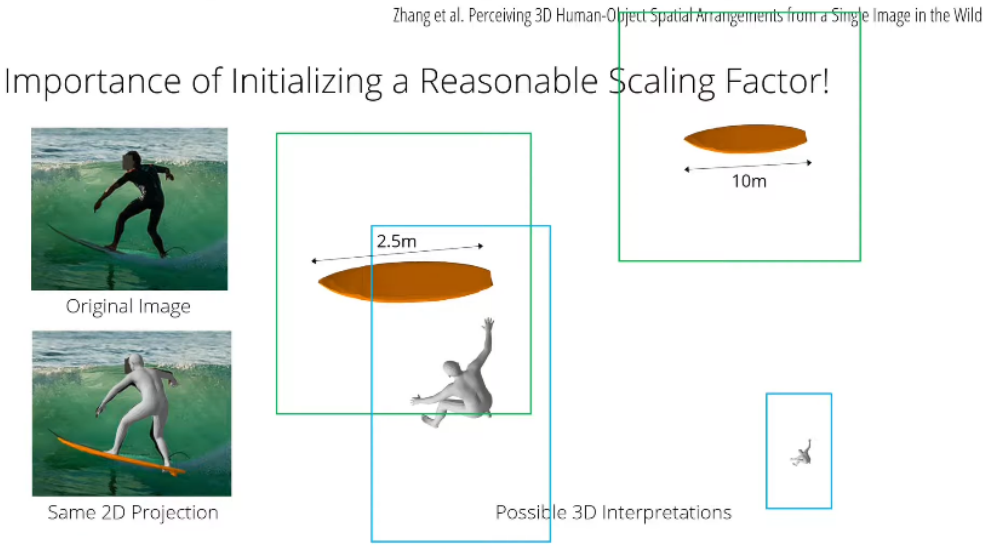
In order to identify human-object interaction, we first determine if two parts interact by using 3D bounding boxes. For example, people usually grab tennis rackets by the handle using their hands As shown here, the handle of the tennis racket and the hand of the person is not in contact, but their 3D bounding boxes overlap.
We impose our loss which pulls the interacting parts closer together. Here, we visualize the improved arrangement. To identify human-object interaction, the initial size of objects is very important.



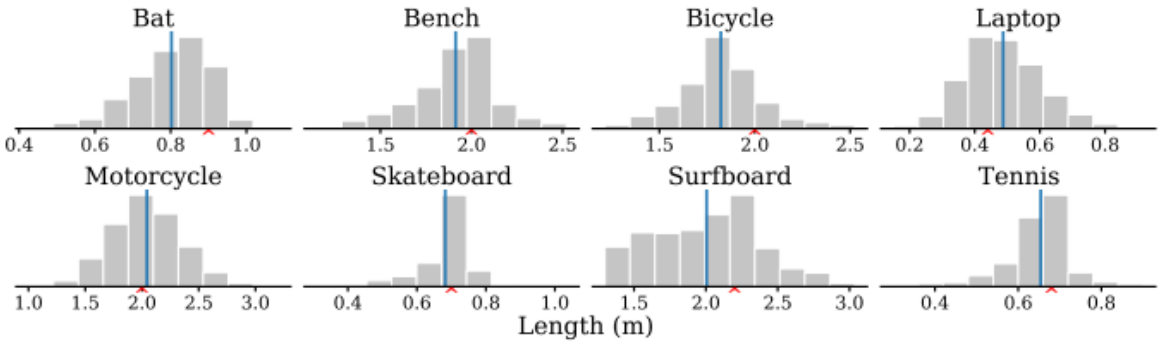
- **Scale loss:**

Taking a prior on human and object intrinsic scale:

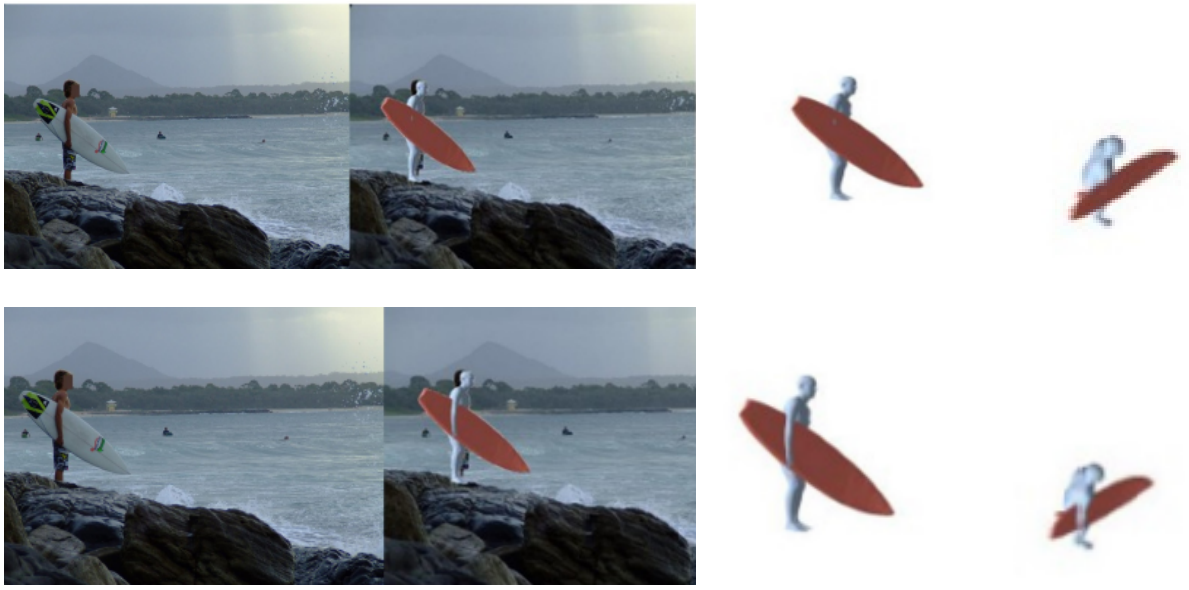
- If the surfboard is initially a reasonable size, say two and a half meters long, then the surfboard is close enough to the person to correctly detect the interaction.



- If we were to initialize the surfboard to be unrealistically large, the 3D bounding boxes would no longer overlap.
- To start our optimization process, we use an internet search to find the usual size of objects
- The red caret denotes the size resulting from the hand-picked scale used for initialization. The blue line denotes the size produced by the empirical mean scale of all category instances at the end of optimization.



- **Ordinal depth loss(Correct depth ordering):**
  The depth ordering inferred from the 3D placement should match that of the image. While the correct depth ordering of people and objects would also minimize the occlusion-aware silhouette loss.



- **Collision loss:**

  It is used to avoid interpenetration of humans and objects that occupying the same 3D space. Promoting proximity between people and objects can exacerbate the problem of instances occupying the same 3D space.

6.4. Deployment:

- After testing the model on COLAB notebook, we switched to the next step(AWS Cloud), We worked on EC2-AWS Linux-based instance which is a standard computing unit on amazon web service

- We uploaded our project files to AWS cloud as it will facilitate Our work along with the productivity speed, not to mention GPU resources. After some network issues, we managed to use FLASK Successfully to deploy our model not only on a local server But also publicly for anyone to use at 35.86.170.176:5000

- **Now our model is ready to deploy! We chose to display our output divided into four images:**
  1. Instance segmentation for the human and objects in the input picture
  2. human 3d mesh
  3. Interaction between human and object(front view)
  4. Interaction between human and object(top view)
- **Some of the drawbacks of the App is Time complexity: We are actually running four models altogether to obtain this result:**

  1- detecrton segmenter to obtain the first picture

  2- frankmocap to generate human mesh

  3- neural-renderer to create 3d object mask

  4- And last but not least our PHOSA model to optimize interaction

Between 3d human mesh and 3d object mesh.

- **Some workaround solutions:**

  1- instead of initializing our model randomly (far away from the best weights), during training time we consistently used the optimized weights (output) as an input for the next iterations.

  2- we managed to reduce the number of iterations to only 25 instead of 100 keeping most of the great quality of the output. (Accepted tradeoff)

  3- we can also reduce time complexity by switching to a higher GPU Like RTX or workstations GPU.

7. Challenges

7.1 Model Drawbacks:

- **Human pose failure:** Our human pose estimator sometimes incorrectly estimates the pose of the person.
- **Object pose failure:** The predicted masks are sometimes unreliable for estimating the pose of the object.
- Incorrect reasoning about interaction due to scale.

7.2 Implementation Challenges:

- Randomization
- Inference time
- Detectron different versions
- Torch version conflicts
- Prepare objects & JSON files
- Data annotation
- Sequential processes of the project

8. Future Work:

- Add more classes
- Enhance complex images
- Make human objects more reliable
- Change body motion capture with Whole-body Motion Capture (body + hands)
- Real-time human-object interaction
- Define mesh for specific human features