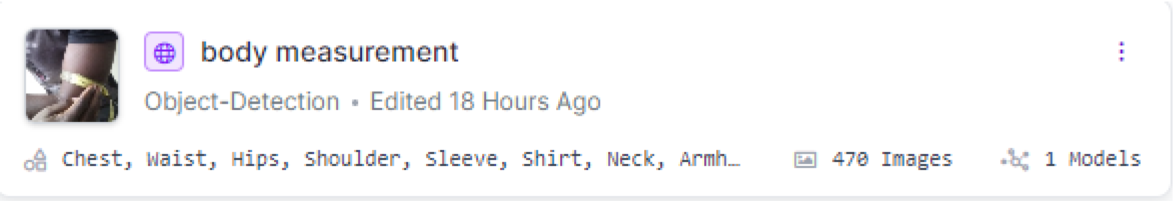
Image-Based Body Measurement Detection Challenge

Overview

This documentation provides a step-by-step guide to measure body circumferences, including chest, waist, hip, neck, wrist, sleeve, and shoulder circumferences, length, and width, from bounding box annotated TensorFlow Object Detection datasets in a Google Colab notebook and visual studio code. The approach involves using a pre-trained object detection model to locate a person’s body measurement in images, extracting bounding box coordinates, and making approximate measurements based on these bounding boxes.

Datasets collection and Annotation

- ✓ Image dataset collected randomly from the internet (human posing for body measurement) and managed to collect 85 images, which are annotated using a bounding box in Roboflow.
- ✓ Later performed Preprocessing and Augmentation in the annotated images, since we've got a small amount of dataset. So, we enhance and augment the dataset for further implementation.
- ✓ After applying several layers of augmentations, doing few preprocessing, our dataset was modified for further training and analysis.
- ✓ Took the raw URL of the pre-trained model implementations and also downloaded the TensorFlow object detection in CSV format for further prediction and to get circumferences, width, and length of the mentioned body part.
- ✓ We can also use key point annotation or 3D posing for body measurement detection, but the main challenge was finding the proper dataset and annotating using this dataset. If we can manage to use the above annotation method, we can certainly find a better performed and precise model for our Image-Based Body Measurement Detection. Which I’m highly interested working on later.



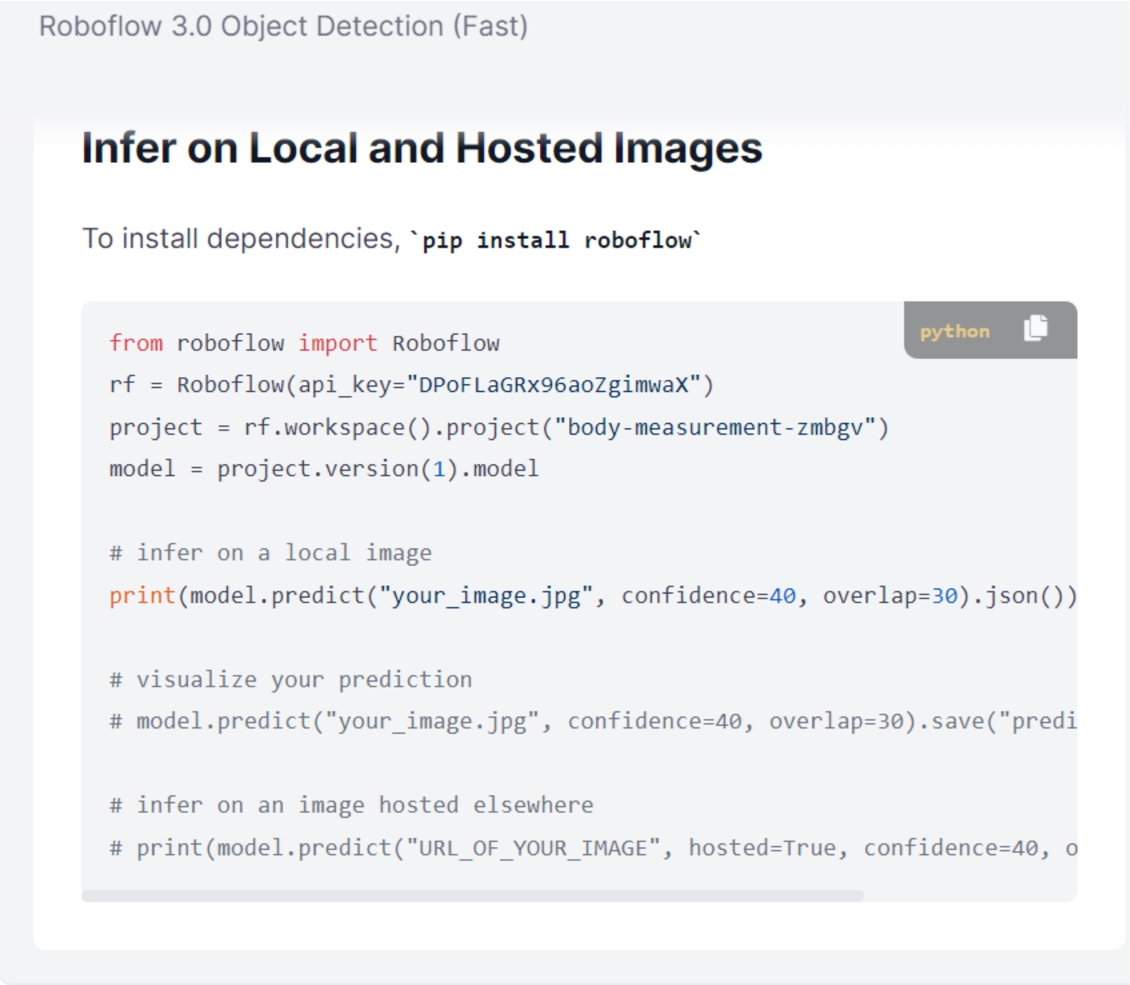
body measurement

Object-Detection • Edited 18 Hours Ago

Chest, Waist, Hips, Shoulder, Sleeve, Shirt, Neck, Armh…

470 Images

1 Models



Roboflow 3.0 Object Detection (Fast)

Infer on Local and Hosted Images

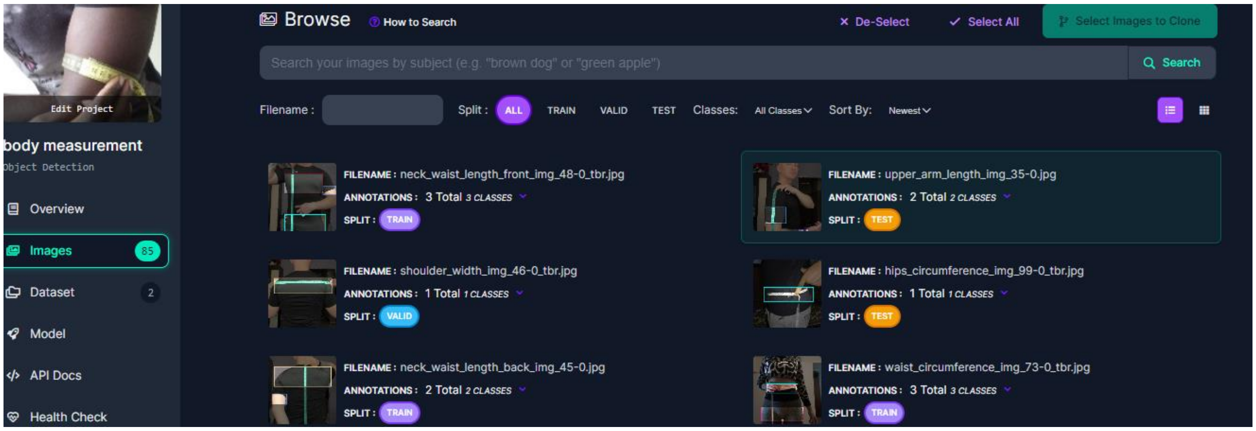
To install dependencies, `` `pip install roboflow` ``

```python
from roboflow import Roboflow
rf = Roboflow(api_key="DPoFLaGRx96aoZgimwaX")
project = rf.workspace().project("body-measurement-zmbgv")
model = project.version(1).model

# infer on a local image
print(model.predict("your_image.jpg", confidence=40, overlap=30).json())

# visualize your prediction
# model.predict("your_image.jpg", confidence=40, overlap=30).save("predi

# infer on an image hosted elsewhere
# print(model.predict("URL_OF_YOUR_IMAGE", hosted=True, confidence=40, o
```



FILENAME : neck_waist_length_front_img_48-0_tbr.jpg
ANNOTATIONS : 3 Total 3 CLASSES
SPLIT : TRAIN

FILENAME : upper_arm_length_img_35-0.jpg
ANNOTATIONS : 2 Total 2 CLASSES
SPLIT : TEST

FILENAME : shoulder_width_img_46-0_tbr.jpg
ANNOTATIONS : 1 Total 1 CLASSES
SPLIT : VALID

FILENAME : hips_circumference_img_99-0_tbr.jpg
ANNOTATIONS : 1 Total 1 CLASSES
SPLIT : TEST

FILENAME : neck_waist_length_back_img_45-0.jpg
ANNOTATIONS : 2 Total 2 CLASSES
SPLIT : TRAIN

FILENAME : waist_circumference_img_73-0_tbr.jpg
ANNOTATIONS : 3 Total 3 CLASSES
SPLIT : TRAIN

A few snips of our pretrained dataset in Roboflow

Approach

The approach consists of the following steps:

Setup and Imports: Import necessary libraries and set up the Colab environment.

Loading the pretrained Model: Load a pre-trained TensorFlow Object Detection model to perform object detection.

Load the Dataset: Load the dataset containing annotated bounding boxes and images.

Defining a Function to Perform Inference: Create a function to perform inference on the dataset using the loaded model.

Performing Inference and Measure Circumferences: Loop through the dataset, perform inference to obtain bounding box predictions, and calculate circumferences based on the bounding boxes.

Usage

Follow these steps to use the provided code for body circumference measurement:

Step 1: *Set Up Colab Environment*

Before we start, we must ensure we have access to Google Colab with proper set up and installation libraries.

Step 2: *Import the Code*

Importing all the libraries and making sure we've all the libraries that we're going to need later for further implementation and analysis.

Step 3: *Load the Model*

Providing path to the pre-trained TensorFlow Object Detection model checkpoint. This model should be capable of detecting people in images. We've got a pretrained model already from Roboflow. Now we have accessed the model to get the measurements accurately.

Step 4: *Load the Dataset*

Now loading the annotated and downloaded TensorFlow Object Detection dataset. We must ensure that our dataset includes images and bounding box annotations for the relevant body parts.

Step 5: *Run the model with appropriate dataset and model*

The code will load the dataset, perform inference on the images, calculate circumferences based on bounding boxes, and print the measurements for the circumferences, width, and length based on the annotated dataset.

Step 6: *Interpret the Measurements*

The code will print or store the calculated circumferences for each image. Now we can interpret the measurements based on our requirements. It is precise that the measurements are approximate and based on the bounding box dimensions.

Dependencies

The overall pretraining dataset and implementation relies on the following dependencies:

Roboflow: We collected the image from internet and annotated based on image nature and given category using bounding box. After then, took the pretrained model and annotated dataset in tensorflow object detection in csv format for further interpretation and prediction.

TensorFlow: Used for loading the pre-trained object detection model and performing inference.

NumPy: Used for numerical calculations.

OpenCV: Used for image handling and preprocessing.

TensorFlow Object Detection API: Provides utilities for object detection, including model loading and inference.

Important Considerations

Accuracy: The approach assumes approximate measurements based on bounding boxes. For more accurate measurements, we can consider using more advanced techniques such as keypoint detection or pose estimation using appropriate amount of data.

Dataset: Ensure that the dataset includes accurate bounding box annotations for the body parts you want to measure.

Conversion Factor: The code uses an approximate conversion factor (e.g., pixels to centimeters) for measurements. Adjust this factor based on the dataset and image resolution.

Model Choice: Choose a pre-trained object detection model that can effectively detect people in the dataset that we find from our annotation in Roboflow.

Refinement: Depending on the specific requirements, we may need to refine the measurements or implement more advanced algorithms in future.