For the project we have done the requirements analysis.

we have many models to convert 2D to 3D human modeling. As we said in the proposal of the project we have decided to use Mediapipe library.

The media pipe library will lets model 2d human modeling to the 2-D key points and we planing to Tensorflow JS New technology to model in 3 D. we are looking at various options

and we will decide the best fit model for the project.

Literature Survey:

GHUM & GHUML: Generative 3D Human Shape and Articulated Pose Models

Authors: " Hongyi Xu Eduard Gabriel Bazavan Andrei Zanfir

William T. Freeman Rahul Sukthankar Cristian Sminchisescu"

link:
https://openaccess.thecvf.com/content_CVPR_2020/papers/Xu_GHUM__GHUML_Generative_3D_Human_Shape_and_Articulated_Pose_CVPR_2020_paper.pdf

summary:

The authors use two different models for 3D shape and articulated pose estimation: GHUM and GHUML. GHUM represents the 3D shape of a human body using a GPLVM-based model. GHUML is an extension of GHUM that models the articulated pose of a human body in addition to its shape.

The authors evaluated the proposed models on two publicly available datasets: the HumanEva-I dataset and the MPII Human Pose dataset. They compared their results with state-of-the-art methods and found that their approach outperforms the existing methods in terms of accuracy and efficiency.

GHUM is a generative model that estimates 3D human body shape from a set of 3D body scans. The authors used the SMPL body model to represent human body shape in a lower-dimensional latent space. They then trained the GPLVM to learn the mapping between the latent space and the 3D body

scans. The resulting model can be used to generate realistic 3D body shapes that are consistent with the input data.

GHUML is an extension of GHUM that incorporates articulated pose estimation. In addition to 3D body scans, GHUML also takes as input 2D joint locations, which are used to estimate the pose of the body. The authors used a modified version of the SMPL body model that can represent articulated poses. They then trained the GPLVM to learn the mapping between the latent space and the 3D body scans and the 2D joint locations. The resulting model can be used to generate realistic 3D body shapes and poses that are consistent with the input data.

The authors evaluated the proposed models on two publicly available datasets: the HumanEva-I dataset and the MPII Human Pose dataset. They compared their results with state-of-the-art methods and found that their approach outperforms the existing methods in terms of accuracy and efficiency. They also conducted experiments to show that the proposed models can generalize well to new individuals and can be used for tasks such as motion capture and virtual try-on.

In conclusion, the paper presents a promising approach for 3D human shape and articulated pose estimation using generative models. The proposed models are accurate and efficient, and they can be used for a variety of applications in computer vision, robotics, and virtual reality.

paper 2:

Total Capture: A 3D Deformation Model for Tracking Faces, Hands, and Bodies

Authors: " Hanbyul Joo, Tomas Simon, Yaser Sheikh"

Summary:

Total Capture is a research paper that presents a generative 3D deformation model for tracking and reconstructing human faces, hands, and bodies in motion from monocular RGB videos. The proposed model aims to capture not only the articulated pose of the subject but also its shape and appearance. The authors argue that a 3D model is superior to 2D models in terms of accuracy and robustness to occlusions and variations in lighting and background.

The model consists of three components: a shape prior, a motion prior, and an observation likelihood. The shape prior captures the subject's shape variations across different poses, while the motion prior

models the temporal consistency of the subject's motion. The observation likelihood estimates the likelihood of the observed image given the subject's pose, shape, and appearance.

The authors trained the model on a large dataset of RGB videos captured from multiple viewpoints, with ground-truth 3D pose, shape, and appearance annotations. They evaluated the model on various benchmarks, including the HumanEva and MPII datasets, and showed that it outperformed state-of-the-art methods in terms of accuracy and robustness.

Total Capture has many potential applications, such as virtual and augmented reality, gaming, animation, and biomechanics. The proposed model can also be extended to other domains, such as animal tracking and industrial automation. However, the model has some limitations, such as the need for accurate 3D annotations during training and the high computational cost of inference. The authors suggest several directions for future research, such as incorporating audio and tactile feedback and exploring more efficient inference algorithms.

Limitations:

While Total Capture presents a highly accurate and robust 3D deformation model for tracking and reconstructing human faces, hands, and bodies in motion, it also has some limitations that should be considered. Some of these limitations are:

The need for accurate 3D annotations during training: The model requires accurate 3D annotations of pose, shape, and appearance during training, which can be time-consuming and expensive to obtain. The annotations also need to be consistent across different subjects and viewpoints, which can be challenging in practice.

The high computational cost of inference: The model involves a large number of parameters, and the inference process can be computationally expensive, limiting its real-time applications.

Limited generalization to novel scenarios: While the model is highly accurate in the domains it was trained on, it may not generalize well to novel scenarios or domains, where the data distributions and variations may differ significantly from the training set.

The limited expressiveness of the model: The model is based on a predefined set of shape and motion priors, which may not capture all the variations and complexities of human motion and appearance.

The limited ability to handle occlusions: The model may struggle to handle occlusions, where parts of the subject's body or face are not visible, as it relies on the observation likelihood to estimate the subject's appearance.

Paper 3: Constructing 3D human model from front and side images

Constructing a 3D human model from front and side images is a challenging computer vision problem that involves recovering the 3D structure of a person's body from 2D images captured from different viewpoints. To address this problem, several approaches have been proposed in recent years that leverage deep learning and geometric modeling techniques.

One common approach is to use multi-view geometry to estimate the camera parameters and triangulate the 3D points corresponding to the person's body parts from the two images. This can be done using techniques such as stereo reconstruction, structure from motion, or bundle adjustment. However, this approach can be computationally expensive and may not work well in cases where the two images have significant differences in viewpoint or lighting.

Another approach is to use deep learning models such as convolutional neural networks (CNNs) or generative adversarial networks (GANs) to directly regress the 3D shape and pose of the person's body from the 2D images. This can be done using methods such as 3D morphable models or mesh-based representations. These approaches have shown promising results in reconstructing realistic and detailed 3D human models, but they also require large amounts of training data and can be computationally intensive.

Overall, constructing a 3D human model from front and side images is a challenging and active research area, with many different approaches being developed and refined. While significant progress has been made in recent years, there is still much work to be done to make these techniques more accurate, efficient, and applicable in real-world scenarios.

Limitaions:

One of the main limitations of constructing 3D human models from front and side images is that the process is sensitive to the quality and consistency of the input images. Even minor variations in lighting,

background, and pose can significantly impact the accuracy of the resulting 3D model. This can be challenging to achieve in real-world scenarios, where lighting conditions and backgrounds can vary widely.

Paper 4: Applying 3D Human Model in a Posture Recognition System

The paper "Applying 3D Human Model in a Posture Recognition System" presents a posture recognition system that utilizes a 3D human model to recognize human postures from a single depth image. The proposed system includes three main steps: body part detection, 3D pose estimation, and posture recognition.

In the first step, body parts (head, neck, shoulders, elbows, wrists, hips, knees, and ankles) are detected using a combination of depth thresholding and a skin color model. In the second step, a 3D human model is fit to the detected body parts using an iterative closest point (ICP) algorithm. Finally, in the posture recognition step, a support vector machine (SVM) classifier is used to classify the posture based on the 3D joint positions of the fitted human model.

The proposed system was evaluated on a dataset of 10 different postures and achieved an average recognition rate of 95.2%. The experimental results demonstrate the effectiveness of the proposed system in recognizing human postures from a single depth image.

Overall, the paper presents a novel approach to posture recognition that utilizes a 3D human model and achieves high recognition rates. The proposed system has potential applications in fields such as human-computer interaction, gaming, and healthcare. However, the system is limited to recognizing postures in a controlled environment and may not perform well in real-world scenarios with complex backgrounds and lighting conditions.

Limitations:

There are several limitations of applying a 3D human model in a posture recognition system, including:

Complexity: Implementing a 3D human model in a posture recognition system requires sophisticated algorithms and techniques, making it a complex and computationally intensive task.

Hardware requirements: The use of 3D human models in posture recognition systems often requires specialized hardware, such as depth cameras or 3D scanners, which may not be readily available or affordable.

Calibration and alignment issues: Accurate recognition of human postures requires precise calibration and alignment of the 3D human model with the real-world environment. This can be challenging and time-consuming, especially in dynamic environments where the model may need to be recalibrated frequently.

Limited training data: Developing accurate posture recognition models requires large amounts of training data, which may be limited or difficult to obtain for certain postures or movements.

Human variations: The effectiveness of a posture recognition system based on a 3D human model may be affected by variations in human anatomy and movement patterns. This can make it challenging to develop a model that works well for all individuals, particularly those with unique physical characteristics or movement styles.

Requirements Analysis:

Building a 3D human modeling project in Python involves several steps:

Acquiring 3D human body data: This can be done by using open-source datasets or by creating custom datasets using sensors such as depth cameras or motion capture systems.

Preprocessing the data: The acquired data needs to be preprocessed before using it to build a 3D human model. This includes tasks such as cleaning the data, removing noise, and aligning the different 3D scans.

Creating a 3D mesh: The preprocessed data can be used to create a 3D mesh representation of the human body using libraries such as PyMesh or Blender.

Rigging the 3D model: The 3D mesh needs to be rigged to a skeleton or a set of joints, enabling it to be posed or animated. This can be done manually or by using libraries such as OpenPose or TensorFlow.

Texture mapping: Texture mapping involves applying an image to the 3D mesh to give it color or texture. This can be done using libraries such as PyOpenGL or Pygame.

Rendering the 3D model: The final step is to render the 3D model, allowing it to be viewed from different angles or lighting conditions. This can be done using libraries such as PyOpenGL or Pygame.

Python provides a range of libraries and tools that can be used to build a 3D human modeling project. Some of the commonly used libraries include PyMesh, Blender, OpenPose, TensorFlow, PyOpenGL, and Pygame. However, building a 3D human modeling project can be a complex task that requires a strong understanding of 3D modeling, programming, and mathematics. Therefore, it is recommended to have prior experience in these areas before attempting to build a 3D human modeling project in Python.

Media pipe:

MediaPipe Pose Detection is a computer vision library developed by Google that enables real-time human pose estimation using machine learning. The library uses a machine learning algorithm to detect the pose of a person in an image or video and provides 33 landmarks for different body parts such as the nose, shoulders, elbows, wrists, hips, knees, and ankles.

Here is an example of how to use MediaPipe Pose Detection in Python:

```
import cv2
import mediapipe as mp

mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose
```

```
cap = cv2.VideoCapture(0)
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      continue

    image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
    image.flags.writeable = False
    results = pose.process(image)
    image.flags.writeable = True
    image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

    if results.pose_landmarks:
      mp_drawing.draw_landmarks(
        image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS)

    cv2.imshow("MediaPipe Pose", image)
    if cv2.waitKey(5) & 0xFF == 27:
      break

cap.release()
cv2.destroyAllWindows()
```

In this code snippet, we are capturing the video feed from the camera and passing it through the MediaPipe Pose Detection algorithm to detect the pose of the person in the video. The detected

landmarks are then drawn on the video feed using the mp_drawing.draw_landmarks() function. The min_detection_confidence and min_tracking_confidence parameters specify the minimum confidence values required for the algorithm to detect and track the pose landmarks.

Overall, MediaPipe Pose Detection is a powerful tool for real-time human pose estimation that can be easily integrated into Python projects using the mediapipe library.