Abstract:

Contents

1 Introduction

2 Process Pipeline

The whole process of fault estimation can be seen as a pipeline. We start at the most basic starting block, the camera streams, and end at the most complex block, the fault estimation. This pipeline can be divided into 7 steps or phases.

In this section we give a basic overview over the whole process. We go into more detail in the following sections.

2.1 Stream Pre-Processing Phase

Firstly, we preprocess the seperate streams of the various depth sensors. This is done by aligning the point clouds and the RGB streams. There are multiple ways to align the point clouds, but we found that, due to the high noise level of the depth data, the best way to align the point clouds is to do it manually. Since we scale the point cloud into meters according to the meter-per-unit ratio provided by the depth camera, manual alignment is a matter of measuring the real-world distances along the axis and then translating the point cloud accordingly. Rotation on the other hand is a lot harder to do manually, so we use the NDT algorithm to align the point clouds, with the measured distances as an initial guess.

The NDT algorithm, Normal Distribution Transform algorithm, is a non-linear optimization algorithm, which finds the best rotation and translation to align the point clouds[1]. The more data we can provide to the NDT algorithm, the better the alignment will be. Therefore, if we provide the algorithm with the measured distances along the axis, it will be able to find the best rotation without changing the translation.

The fitness score is provided after the alignment and is a measure of how well the point clouds are aligned. The fitness score is calculated by comparing the aligned point clouds to the original point clouds. The fitness score represents the mean squared distance from each point in the aligned point cloud to its closest point in the static point cloud. Hence, the lower the fitness score, the better the alignment. Depending on the overlap of the point clouds the fitness score can vary a lot. If the point clouds are not overlapping significantly, the fitness score will be higher since the closest point is still far away and will be a bad indicator of a good alignment. Therefore, to validate the alignment of the point clouds we also use the overlap score. The overlap score is calculated by comparing the aligned point clouds to the original point clouds. The overlap score represents the percentage of points in the aligned point cloud that are within a certain distance of a point in the static point cloud. With both metrics, we can validate the alignment of the point clouds.

The rotation and translation of the point clouds are stored in the file that contains all meta-data about the recording so that we can use them to align the point clouds in the future.

2.2 Data Acquisition Phase

The second phase is the Data Acquisition Phase, where we process the streams of multiple cameras and store the camera intrinsics, the RGB stream, and the Depth stream. Using the intrinsics we can recreate the point cloud based on the depth data at any point in the future. It is possible to set the

2.3 Data Population Phase

Thirdly, to achieve the highest framerate, we calculate the skeleton based on the recorded data and add it to the dataset in a separate step. The RGB stream is used to create the dataset for the skeleton detection and the depth stream is used to create the point clouds for the calculation of global skeleton points. We store both the local 2D coordinates in accordance with the image used for the skeleton detection, as well as the global 3D coordinates based on the aligned point clouds. Additionally, OpenPose provides us with a confidence score for each joint.

2.4 Data Evaluation Phase

Next, we evaluate the recorded data and especially the detected skeleton. We focus specifically on the joints with a low confidence value. Since we have the skeleton data from two perspectives at the same time, we can compare the 3D coordinates of both streams. If we notice a missing joint, we can estimate its position using the same joint of the other stream, provided it has a relatively high confidence value. In other cases, we can improve the calculated skeleton, by forming a weighted average of the 3D coordinates based on the confidence of each Joint. Another step might be to update the 2D Coordinates of the skeleton but we might just leave that for future work.

2.5 Data Augmentation Phase

Once the data is cleaned and we filter recordings with too many missing joints, we can augment the data to simulate a larger amount of data with the ability to create faulty scenarios controlled. One major fault is the seemingly random detachment of the joint to a side. This especially affects the legs and arms, therefore we will have a bias toward limbs with this augmentation. Furthermore, we randomly move the joints with low confidence. Another fault is the disappearance of joints. We use the same bias as with the random detachment, i.e. we take the limb bias, as well as the confidence into consideration.

This phase allows us to create a large amount of data with a controlled amount of faults. This is important since we want to be able to train a model that can detect faults in the data. The augmented data is stored in a separate file so that we can use it to train the model and compare it to the original manually checked ground truth.

2.6 Model Training Phase

Using this enlarged dataset, we can train a Neural Network to recognise faults in the data. We use the depth data as input, the skeleton data as input, and a combination of both as input. We also experiment with different network layouts, such as a fully connected network, a convolutional network, and a combination of both. We use the augmented data to train the model and the manually checked ground truth to validate the model. We use the validation data to determine the best model and the best network layout. We use the best model to predict the faults in the data.

2.7 Model Evaluation Phase

Finally, we evaluate our model by calculating different error metrics such as the mean absolute error, the mean squared error, and the root mean squared error. We also calculate the accuracy of the model, which is the percentage of correctly predicted faults. We also calculate the precision and recall of the model. The precision is the percentage of correctly predicted faults out of all predicted faults. The recall is the percentage of correctly predicted faults out of all faults in the data. We also calculate the F1 score, which is the harmonic mean of the precision and recall. The F1 score is a good indicator of the overall performance of the model.

3 Stream pre-processing

4 Data acquisition

5 Data population

6 Data Evaluation

7 Data Augmentation

This phase allows us to create a large amount of data with a controlled amount of faults. This is important since we want to be able to train a model that can detect faults in the data. The augmented data is stored in a separate file so that we can use it to train the model and compare it to the original manually checked ground truth.

8 Model training

9 Model evaluation

10 Results

Here we present the results of our experiments. We first present the results of the experiments with the different network layouts. We then present the results of the experiments with the different input data. We also present the results of the experiments with the different error metrics. Finally, we present the results of the experiments with the different data augmentation techniques.

11 Conclusion

12 Future work

References

[1] Martin Magnusson. “The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection”. PhD thesis. Dec. 2009.