Garbage Sorting Manipulator Based on ResNet50

Shen Qirong (12010311), Zhou Mingyu (12012725), Wang Zimeng (12011711)
Wen Junzhe*(11910703), Liu Qian*(11912312)

Abstract—This course project aims to develop a new type of garbage sorting robot that can recognize the position of garbage through a camera and make corresponding posture changes so that it can reach the posture of picking up garbage. In this project, we train a machine learning model to recognize different types of garbage and sort them to the correct location based on ResNet50. We also use OpenCV to applied a machine vision work, and we finish a robotic simulation task based on Pybullet software.
[1]

I. Introduction

At present, garbage classification has become a hot topic in the global environmental protection cause. With the continuous acceleration of global urbanization, the amount of garbage is also increasing, and garbage classification is one of the important means to effectively solve the problem of garbage management. In this context, machine learning technology is widely used in the field of waste sorting to improve the efficiency and accuracy of waste sorting. Machine learning is an artificial intelligence technique that trains algorithms to recognize patterns and patterns and make predictions or decisions. In garbage sorting, machine learning can automatically identify the type of trash and sort the garbage into the correct bin by analyzing garbage images, sounds, or other sensor data. This technology can not only improve the accuracy and speed of garbage sorting, but also reduce the error rate and cost of manual operations. In recent years, many researchers and companies have begun to experiment with applying machine learning to the field of waste sorting. Google, for example, has developed a machine learning framework called TensorFlow[1] to train garbage sorting algorithms. At the same time, some researchers are also exploring how to apply deep learning technology to waste sorting to improve sorting accuracy. However, there are still some challenges with machine learning in the field of waste sorting. On the one hand, due to the wide variety of garbage, machine learning algorithms need to process a large amount of data and labels to get a good sorting effect. On the other hand, waste sorting involves complex factors such as environmental factors and human behavior, which can affect the accuracy and stability of machine learning algorithms. In summary, machine learning technology has broad application prospects in the field of waste classification, but more research and innovation are still needed to solve the corresponding challenges. As technology continues to advance, it is believed that machine learning will bring more efficient and accurate solutions for waste sorting.

In the project, we will first collect a large amount of junk image data, label and classify it. We will then use deep learning algorithms to train and optimize this data to achieve accurate sorting of garbage. Next, we will use computer vision technology to identify and sort waste in real time for subsequent automated processing. In terms of garbage sorting and treatment, we will use robotic arm simulation technology to design and implement an automated garbage sorting and treatment system. The realization of this project will help to improve the efficiency and accuracy of waste sorting, reduce the workload of manual waste disposal, and also help protect the environment and sustainable use of resources. This project will mainly include the following parts:

A. Machine learning based on ResNet: ResNet (Residual Network) is a deep convolutional neural network, proposed by Kaiming He of Microsoft Research Asia and others in 2015. The main feature of ResNet is the introduction of the concept of residual learning, which solves the problem of gradient vanishing and gradient explosion in deep neural networks through shortcut connection, so as to achieve deeper network structures. The advantage of ResNet (residual network) over other deep neural networks is that it can train very deep networks without the problem of gradient vanishing or gradient explosion. This is because ResNet uses residual blocks, which allow information to jump through the network, avoiding information loss and degradation as it passes through the network layer by layer. When dealing with large-scale data sets, the advantages of ResNet are even more pronounced. For example, on the ImageNet dataset, ResNet-152 has better performance than other deep neural networks, such as VGG-19 and Inception-v4. This is because ResNet can better capture details and features in images, which improves classification accuracy.

B. Computer vision based on OpenCV: OpenCV is an open-source computer vision library developed by Intel. It provides rich image processing and computer vision algorithms, including image filtering, feature extraction, object detection, face recognition and other aspects. OpenCV supports a variety of programming languages and platforms, including C++, Python, Java, etc., which can facilitate image processing and computer vision application development. In our project, by the help of the OpenCV, we first perform Gaussian filtering on the image,

[1]Our senior teammate Wen Junzhe and Liu Qian did not have much contributions to the project

perform binarization and grayscale processing, and then extract the outline of the items in the image after the opening operation, and finally calculate its center point.

C. Robotic simulation based on Pybullet: Pybullet[5] is an open source simulation software based on a physics engine, developed by Robotics and AI Group at Toyota Technological Institute at Chicago. It can be used to simulate the motion and interaction of various physical systems such as robots, objects, and vehicles, and supports a variety of programming languages and platforms, including Python, C++, ROS, etc. The main features of Pybullet are that it is fast, flexible, and easy to use. It adopts an efficient physics engine and optimization algorithms to achieve high-speed, high-precision physics simulation. At the same time, Pybullet also provides a wealth of APIs and tools, which can easily perform model construction, control algorithm design, data visualization and other operations, so that users can quickly conduct simulation experiments and application development. Pybullet has a wide range of applications, including robot control, object grasping, motion planning, virtual reality and other fields. In our project, we built a simulation environment for the robotic arm to perform tasks in PyBullet, added a model of the robotic arm to the environment, and tried to make it automatically perform the tasks we set.

II. Method

A. Problem Formulation

The objective of this project is to develop a real system which could handle a garbage classification problem with robot arm. Before that, we are finishing this task in a simulated environment PyBullet. The garbage can normally be categorized into six classes: paper, cardboard, trash, glass, metal and plastic, but they are all different in material and shape leaving a sever problem in grasping. However,this is not the same way with our aim which is to facilitate a garbage classifier. To simplify this, we use mapped squares for easy grabbing.

And subsequently guide the robot arm to deposit the garbage into the appropriate tray or box. The focus of this problem formulation is specifically on the computer vision aspect, disregarding the grasp problem and assuming that the robot arm is capable of executing the desired actions. The problem can be decomposed into the following key components: 1.Garbage Classification: The system needs to accurately identify and classify different types of garbage objects based on real pictures attached to a cube. This involves developing a computer vision model that can effectively recognize various types of garbage such as plastic, paper, glass, and metal. The model should be able to handle variations in lighting conditions, object orientations, and possible occlusions. HOG[2] feature is also tried for square recognition, but it costs more time so it was not adopted.

2. Get Position and Shoot: In the environment we need to get the 2D coordination of the cube. Than this coordination will be used send to the controller to guide the arm to grasp. Than the camera in the end of arm will shoot when the arm is ready to catch the cube, this means the arm is just over the cube and have got close enough to it. The latest shot picture will be used to classify the garbage cube.

3. Tray and Box Assignment: Once the garbage object is identified and classified, the system should determine the appropriate tray or box to which the object should be deposited. Each tray or box represents a specific category of garbage, and the system needs to associate the recognized garbage object with the corresponding destination.

4. Robot Arm Guidance: Given the classification and tray/box assignment, the system should generate instructions for the Panda robot arm [6]to perform the grasping and depositing actions. Since the grasp problem is simplified in this project, the focus is solely on providing accurate coordinates or positions for the robot arm to move and deposit the garbage into the assigned tray or box.

Overall, the problem formulation involves developing an efficient and accurate computer vision system that can recognize and classify different types of garbage objects, associate them with the appropriate tray or box, and generate the necessary instructions for the Panda robot arm to deposit the garbage accordingly. The system should be robust to variations in object appearance and environmental conditions, while ensuring reliable and precise coordination between the computer vision system and the robot arm.

In modeling the garbage cube, we use Blender to do the modeling and mapping work. The software will give users an initial 1*1*1 cube. By adding an new material module, we can use a picture as the basic color of the new material. But until now the picture will automatically overspread the whole cube other than just show in one face. So we need to edit the material we have created that assign it to a upwards face.

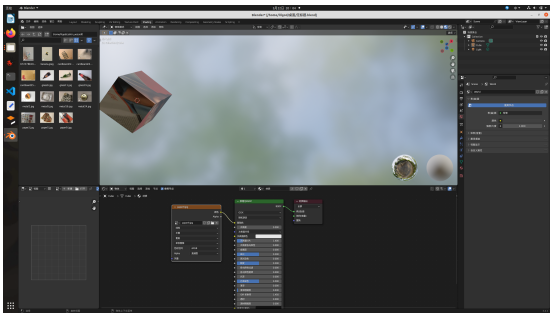

Fig. 1: The overspread cube

However, there still exist the problem that the picture can't show integrally in the face and to solve this problem

we need enter the uv editing function to edit the shown picture and make it perfectly covered the face.

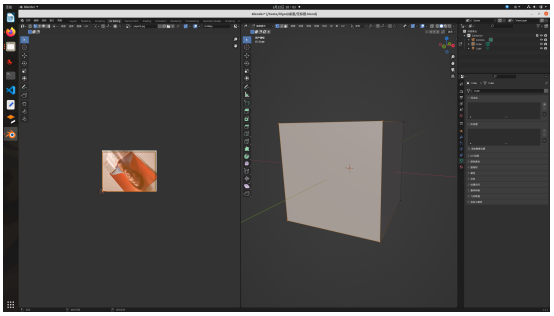

Fig. 2: Edit the shown picture

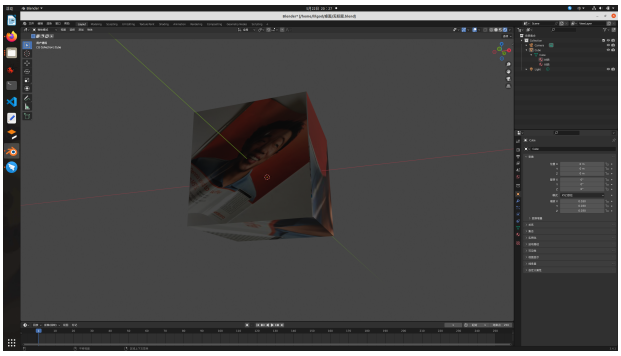

Fig. 3: Edit the shown picture

Finally, we can make the picture showed in the upwards face integrally and than export in obj file.

B. ResNet50

ResNet50[3], short for Residual Network with 50 layers, is a widely used convolutional neural network architecture. It was proposed by Kaiming He et al. in their seminal paper "Deep Residual Learning for Image Recognition" in 2015.ResNet50 is an extension of the original ResNet architecture, which introduced the concept of residual learning to overcome the challenge of training very deep neural networks-degradation

In our project, to analyze visual imagery, we apply ResNet50 to classify garbage with different materials. This neural network can classify 6 kinds of trash by its materials.(paper,cardboard,trash,glass,metal,plastic). The ResNet50 has 5 layers each with convolution layers and batch normalization layers and a fully connected layer as a decesions. For Convolutional Neural Network,

$$z^{[n]} = W^{[n]}x + b^{[n]}$$

$$a^{[n]} = \sigma(z^{[n]})$$

Where W is the weight. b is the bias and σ is the activation function(ReLU). Then we can have ,

$$a^{[n+2]} = \sigma(W^{[n+1]}a^{[n+1]} + b^{[n+1]})$$

However for ResNet50 ,

$$a^{[n+2]} = \sigma(W^{[n+1]}a^{[n+1]} + b^{[n+1]} + a^{[n]})$$

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 4: ResNet50

At the core of CNNs[4] are convolutional layers, which apply a set of learnable kernels to the input image. These filters convolve across the image, performing local operations such as edge detection or feature extraction. By stacking multiple convolutional layers, CNNs are capable of capturing increasingly complex and abstract features.

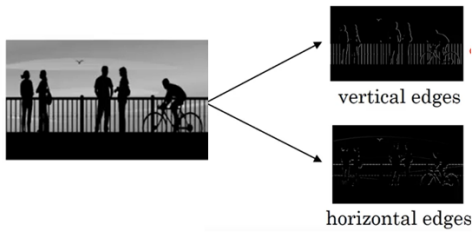

Fig. 5: Apply vertical and horizontal kernals

Pooling layers, typically applied after convolutional layers, reduce the spatial dimensionality of the features based on the idea that the pixels in neighbor represent the same thing. Max pooling, for example, selects the maximum value within a region, effectively downsampling the feature maps. This helps in reducing computational complexity and achieving translation invariance.

Fig. 6: a) Original b)Max pooling with 3*3 c) Max pooling with 5*5

Fully connected layers are often added at the end of CNNs to perform classification tasks. They connect all the learned features from the previous layers to a final output layer, which produces predictions and probabilities for different classes.

The training set is from kaggle, with 6 class each with around one thousand pictures(trash,plastic,paper,cardboard,metal,glass) (https://www.kaggle.com/datasets/keras/resnet50). Restricted by hardware tools, we simplify this to around 400 pictures each. The training set up is done with 7 epoches and a batch size of 32.After training with 7 epoch, the accuracy is up to 94%.

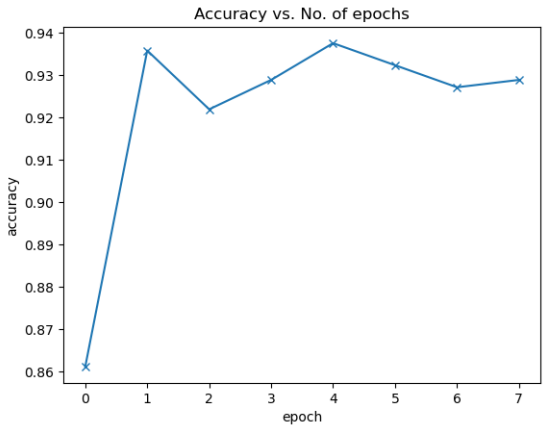

Fig. 7: Training accuracy

III. Experiment

Our research topic is based on the CNN convolutional neural network algorithm garbage sorting manipulator. We designed an experiment to validate the feasibility of the system we designed and built. The specific procedure is illustrated in the diagram below. We focused on the success rate of garbage classification recognition and the success rate of the grasping system in our system's simulated environment. We used these two key factors to verify the feasibility and reliability of our garbage sorting manipulator based on the neural network classifier.

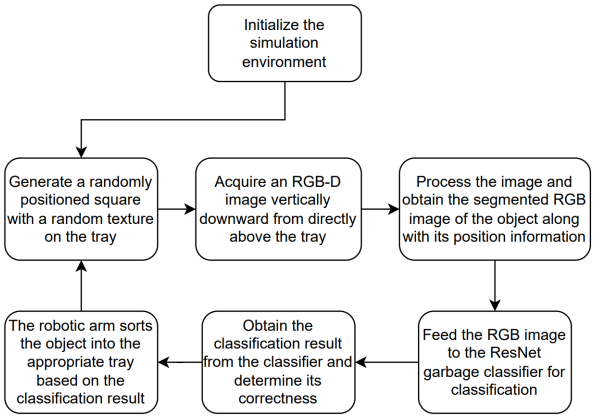

Fig. 8: Experimental Flowchar

A. Simulation Environment

In this experiment, we built a physical simulation environment using pybullet. For the construction of the simulation environment, the ground model used the "plane.urdf" provided by pybullet. The gravity acceleration was set to -9.8. We used the Franka-Panda robotic arm simulation model with a base coordinate of (0, -0.5). Around the robotic arm, we placed four trays with dimensions of 0.6*0.6 meters as containers. The center coordinates of these trays were (0, 0), (0.8, -0.4), (-0.8, -0.4), and (0, -1). The tray in front of the robotic arm served as the space for generating the blocks to be recognized and grasped, while the other three trays served as collection containers for different types of blocks.

For the configuration of the blocks, we used small square blocks with dimensions of 0.04*0.04*0.04 meters. These blocks will be attached with photos from the ResNet official training set, official testing set, and self-shot images. The photo size will be adjusted to be consistent with the surface area of the square block. These photos will include four categories labeled as "paper," "cardboard," "glass," and "metal" respectively, with equal proportions of 1:1:1:1. During the simulation experiment, each block will be randomly assigned one photo, and they will be generated on the tray at a fixed pose with random coordinates.

Fig. 9: blocks with picture in simulation

B. Image Processing

During the experiment, we first randomly selected one block out of the 15 blocks with attached photos and placed it at a random coordinate inside the tray in front of the robotic arm. We used the pybullet.getCameraImage() function to obtain an RGB-D image from a vertical top-down view above this tray and processed the image using OpenCV.

In this part, we already got the image from the camera in the simulated world. The only thing we need to do is image segmentation. The process is :

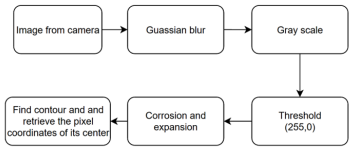

Fig. 10: Image Process

First, we use Gaussian Blur to avoid misidentification of boundaries.Gaussain Blur uses the kernal which combine the nearby pixels together with the weight of the Gaussian distance. Then we trun the image into gray scale since we only want to get the position of the block before binarize

image to intensity of 0 or 255. After that, we could have a white target block and a black background. Applying corrosion and expansion to the image to make the identification more robust. At last, findcontours function in cv2 is called to find the outline of the block. Once we have the outline, we could transform the pixel location to the coordinate simulated world and operate the robot arm to pick up object.

C. Classfier

The RGB image was inputted into the neural network classifier to obtain object classification attribute information. Based on the object's coordinates and classification information, the robotic arm would grasp the object and place it into the corresponding category's three boxes. Paper and cardboard were placed in the tray on the right side of the robotic arm, glass was placed in the tray behind the robotic arm, and metal was placed in the tray on the left side of the robotic arm. At the same time, we obtained the true classification label of the block and the label outputted by the classifier. We calculated the accuracy of the classifier in the simulated environment based on these two pieces of data and displayed it in real-time. The following steps will provide a detailed explanation of each step.

We conducted a total of 200 experiment trials and collected data on the accuracy of the classifier's classification and the success rate of the robotic arm's grasping. The classifier's classification success rate was calculated by comparing the classifier's outputted classification label with the true label represented by the block. The success rate of the robotic arm's grasping indicates whether the robotic arm successfully grasped the block and placed it in the corresponding tray. The statistical results are shown in the two images below.

D. Result and Analysis

IV. Discussion

A. limitation

After finished our course project, We discussed in our group and we reflected all over the material we got in the project. We think that the project is not that perfect. In this part, we will discuss the limitation of our project and we will propose an expectation in the future work.

The limitation of our project is obvious. After the conclusion in our group, we think the whole project is still more like a simple demo of garbage sorting and robotic arm simulation. Our deep learning model can recognize certain types of daily garbage, but we think the accuracy and the efficiency of our model may become lower when it comes to a more complicated situation in which the information of the garbage increased and the task of garbage sorting requires more specific classification. Also, in this project, we finished the task in a simulation environment. It was difficult for us to apply our project to reality due to the lack of time and background knowledge.

B. expectation

In the future, our group want to develop further in this project. We plan to train a better deep learning model which can recognize more types of garbage and perform more accurate and efficient in a more realistic garbage sorting task. We are also looking forward to apply our project material to a concrete robotic arm to make our work more significant in solving the real problem we are facing nowadays. We may need to solve more complicated and difficult problems and learn more about the machine learning and robotics fields. But we believe the project we finished in this semester is a good beginning of our willing. We will keep digging deeper into it.

V. Conclusion

In conclusion, our project successfully applied a deep learning model convolutional neural network to classify garbage with different materials based on RestNet50. We also use OpenCV computer vision method to process the image of the objects and attribute the information of the image, and we utilized Pybullet to construct a physics simulation enviornment to apply a pick and place task on the robotics arm. We successfully achieve what we intended to finish in this project. When finishing our work in this project, we all acquired a deeper knowledge of machine learning and computer vision method, and the ability of robotics simulation is improved. Actually at the beginning, we wanted to do more work related to the garbage sorting task, like calculate out the position of a real garbage and apply a pick and place work efficiency. But when we tried to go further, we found the task was too complicated due to the lack of a real garbage model in the simulation software, and our ranging algorithm was not perfect enough. We found it unstable when we applied the ranging algorithm on the computer vision work parts. So in the future, We think that our work in this project can be improve. First, we can develop a more mature ranging algorithm and to make our robotics arm finish a more complicated grabbing task. Moreover, we want to go further in the simulation part. We can build a better simulation environment involving more realistic models of garbage instead of a block with the image on it. In this way we can truly dig out the possibility of applying our project into real daily garbage sorting tasks.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 886–893 vol. 1, 2005.
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015.
[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. Commun. ACM, 60(6):84–90, may 2017.
[5] Jacopo Panerati, Hehui Zheng, Siqi Zhou, James Xu, Amanda Prorok, and Angela P. Schoellig. Learning to fly - a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. CoRR, abs/2103.02142, 2021.
[6] Saif Sidhik. Franka ros interface: A ros/python api for controlling and managing the franka emika panda robot (real and simulated). Dec 2020.