CS 5335: Robotic Science and Systems (Spring 2023)
HW 6 : Grasping

Report by : Francis Jacob Kalliath

Q1)

a)The benefits are:

Speed:

Simulators can replicate interactions between objects far more quickly than real-world tests, which enables researchers to produce massive datasets of grasp attempts in a relatively short period of time.Researchers can readily change the experiment's parameters, such as the object's size, shape, and substance, using a simulator to produce a wide variety of gripping scenarios for developing and evaluating grasping algorithms. The Simulator provides real-time visualization of the grasp attempts and gives a better understanding of the performance and behavior of the trial.

Safety:

Grasping real-world objects can be risky, particularly if they are delicate, pointy, or heavy.Researchers can limit the possibility of equipment damage by employing a simulator to completely eliminate the risk of injury to human operators.
Additionally, this enables researchers to test grasping algorithms on risky or challenging-to-handle objects that would be impossible to use in actual studies.

Control:

Researchers can precisely manipulate the experiment's conditions via a simulator, including the lighting, camera angle, and object placement.
In order to design and assess grip algorithms, they are able to conduct consistent and reproducible trials as a result of this.

Replicability:

The datasets produced by a simulator are highly reproducible, allowing other researchers to compare their own algorithms and assess their performance using the same dataset.

b)

The unrealistic physics is because of inaccuracies that are caused because of inaccurate object model, inadequate collision detection by the model. Hence, the object is going through the base plane or pushed through the plane. For instance, the gripper pick some objects in an angled position which is not perpendicular which leads to uncertainty of the object resulting in the object falling off. But here it's considered correct.

Another issue that can arise in grasping simulators is unrealistic object slippage or sliding when grasped. This can be particularly noticeable when viewing the simulation from a parallel view with the ground plane, as it can be easier to see when the object is not staying in place or moving in unnatural ways. For example the banana is only being held at one end and also the car is being held by the roof instead of the center of the car. In some cases, the gripper may not have enough force to hold the object securely, or the simulator may not accurately model the friction between the gripper and the object.

Also, inaccurate collision detection, unrealistic object deformations, and difficulty in controlling the gripper to achieve a desired grasp are additional issues for inaccurate prediction. These issues can affect the accuracy and usefulness of the grasping simulator for training and testing grasping algorithms, so it is important to carefully evaluate and tune the simulator to ensure that it accurately models the physics of the real-world objects and grippers.

Q2)
a)
Code in Trainer.py

```
##########################
self.main_path = nn.Sequential(
    nn.Conv2d(in_channels=40,out_channels=128, kernel_size=1, stride=1, padding=0),
    nn.ReLU(True),
    nn.Upsample(scale_factor=8, mode='bilinear', align_corners=True),
    nn.Conv2d(in_channels=128,out_channels= 4, kernel_size=1, stride=1, padding=0),
)

self.residual_path = nn.Sequential(
    nn.Upsample(scale_factor=4, mode='bilinear', align_corners=True),
    nn.Conv2d(in_channels=24, out_channels=4, kernel_size=1, stride=1, padding=0),
)

self.final_blocks = nn.Sequential(
    nn.Conv2d(in_channels=4, out_channels=2, kernel_size=1, stride=1, padding=0),
    nn.ReLU(True),
    nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True),
)
##########################
#raise NotImplementedError
```

```
###############################################
x_wide = self.backbone1(img)
x_narrow = self.backbone2(x_wide)

x_main = self.main_path(x_narrow)
x_residual = self.residual_path(x_wide)

z = x_main + x_residual
output = self.final_blocks(z)
return output
###############################################
raise NotImplementedError
```

b)

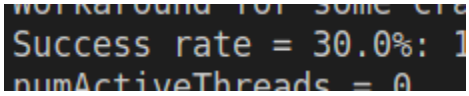
Both local and global information is essential for successful grasping. Local information refers to object characteristics such as shape, size, and texture, which play a critical role in determining the grasp's success for data from the input. Identifying potential obstacles that can interfere with the grasp is also facilitated by local information.

On the other hand, global information provides a broader perspective on the scene's overall layout and the relative positioning of objects and environments. This type of information helps determine which object to grasp first, especially when multiple graspable objects are present in the scene. Hence, the global information is not always needed but if a combination of both local and global information is given then an effective grasping operation can be accomplished.

But in the case of cluttered scenes the global information can affect the robots ability to reach and grasp the object successfully.The orientation and configuration plays an important role in determining whether the grasp can be successfully completed.

Q3)

a)



During model training, adjustments are made to the weights to minimize the loss function on the training data. However, this may not necessarily ensure good performance on the validation set. This is because the validation loss can fluctuate and be influenced by different factors such as data shuffling, outliers, and difficult examples. Whenever the model encounters a challenging example during a specific epoch, this can lead to a sudden increase in the validation loss.

Furthermore, the validation loss curve can become unstable if there is not enough validation data available. A lack of validation data can cause the validation loss curve to display unexpected jumps, making it challenging to accurately evaluate the model's performance after each epoch. Therefore, it is important to have sufficient validation data to maintain a stable validation loss curve, which can provide better insights into the model's performance during the training process.

b)
The validation loss is an imperfect metric to determine the predictive capability of the network for grasps. The reason being, the current loss function used, which is cross-entropy, imposes a significant penalty on the network for predicting an incorrect grasp location. Therefore, if the network predicts a valid grasp location, but it leads to a high loss term (such as a false negative), the validation loss will be high, indicating inadequate performance.

c)
Success Rate: 30%

The objects in the dataset are aligned with the axis, but the assessment simulator positions them with random rotations.

4
a)
Code in dataset.py

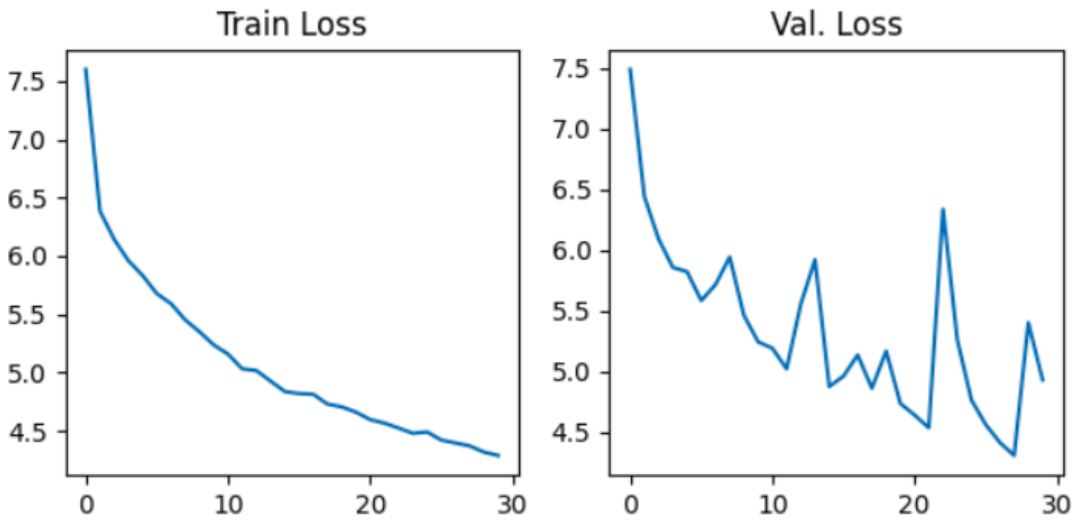
```
################################
# Randomly select a rotation angle
rotation_angle = np.random.choice([0, 90, 180, 270])
rotation_angle = int(rotation_angle)
# Rotate the image
img = TF.rotate(img, rotation_angle)
#old action
x, y,r = action
#image dimension
_,dim_x,dim_y = img.shape
#compute action based on the rotation angle
if rotation_angle == 0:
    newX, newY, newR = x, y, r
elif rotation_angle == 90:
    newX, newY, newR = dim_y - y - 1, x, 1 - r
elif rotation_angle == 180:
    newX, newY, newR = dim_x - x - 1, dim_y - y - 1, r
elif rotation_angle == 270:
    newX, newY, newR = y, dim_x - x - 1, 1 - r

action = np.array([newX, newY, newR])
################################
```

b)

Success Rate : 46%

```
Workaround for some crash in
Success rate = 46.0%: 100%|
numActiveThreads = 0
```



c)
Affine transformations enable us to randomly shift the image both horizontally and vertically, mimicking the object's presence in different locations within the scene.It is important to ensure that the object stays within the workspace and does not move outside it. To maintain the original image size, we can add padding to the image.

Update grasp point, add the translation values to the original coordinates.
new_x = original_x + dx (horizontal translation)
new_y = original_y + dy (vertical translation)
This will update the grasp point coordinates. If the grasp point goes beyond the image boundary, wrap it around the opposite side of the image.

Acknowledgement

Shankara
Arun
Anush
TEjaswini