Advanced Robotic AI

Grasping and Motion Generation

Name: Seonghun Cha, Seonho Kim, Daewon Choi
Department: Department of Electronic Engineering, Department of Artificial Intelligence

1. Introduction

[Motivation]

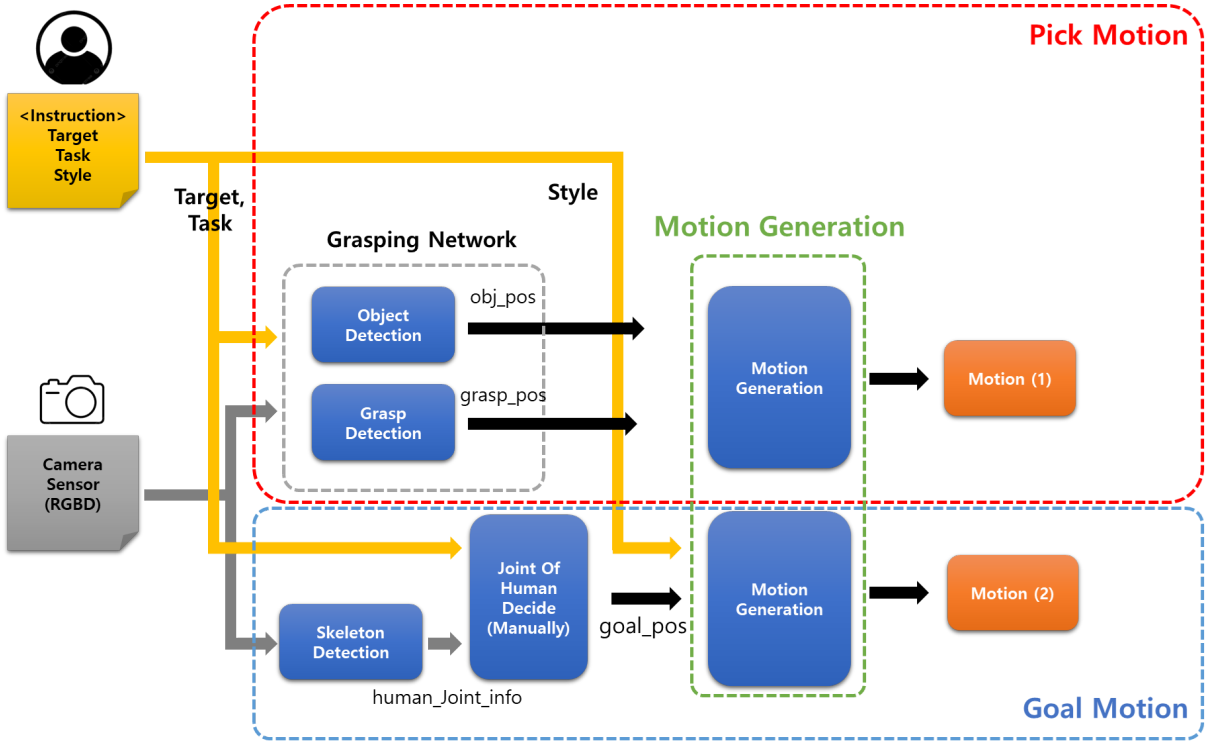
Many countries face the dual challenge of an aging population due to low birth rates and increased life expectancy, leading to escalating demands for elderly care and disability services. The shortage of caregivers, coupled with inadequate support for individuals with disabilities or the elderly, has prompted a rising trend: the use of assistive robots to meet these pressing needs.

The current works of assistive robots, however, are limited to performing specific tasks and often struggle to adapt to different objects and handle diverse shapes of the human body effectively. To address this limitation, we are implementing a skill-based approach that can be reused in learning novel tasks and can adapt to diverse environments.

[Goal of the project]

The primary objective of the project is feeding. In this task, we aim to accomplish three main goals.
- Firstly, we seek to detect human skeletons to provide a more personalized assistive service.
- Secondly, we aim to enable robots to effectively assist the elderly with natural movements and motion generation techniques.
- Lastly, we strive to enhance robustness by enabling detection of various objects.

[Method Overview]



The primary methodology consists of three sub-methods: "Pick Motion," "Goal Motion," and "Motion Generation." The following paragraphs provide an explanation of the functioning of each method. Firstly, the system initiates by obtaining input from the initial user, which includes information regarding the command. This command comprises details about the "Target" object, the "Task" to be executed, and the desired "Style" of motion. In the "Pick Motion" method, the user's command is paired with data from a camera sensor (RGBD) and provided to the Grasping Network. The Grasping Network analyzes the input data to determine the location of the Target object and provides instructions on how to grasp it. Moving on to the "Goal Motion" method, it operates in two stages. Initially, Skeleton Detection is used to detect the positions of a person's joints, providing fundamental information required to execute the user's motion command. Subsequently, a manual analysis of the user's command is performed to establish the final goal, which is the Joint position. Lastly, the "Motion Generation" method is responsible for creating the robot's movements towards the target location. This stage involves the actual generation of movements, where the design and implementation of the robot's motion to reach the target location take place. Consequently, this enables the execution of actions based on the user's command.

2. Install instruction

[Ubuntu]

1. Clone the repository
   1.1. (in your workspace)

```
$ git clone https://github.com/choibigo/robot_ai.git
```

2. Docker execution
   2.1. Docker file build

```
$ cd docker_folder
$ docker build --tag robot_ai_project:1.0 .
```

   2.2. Docker image execution

```
$ docker run -it -v {clone_path}:/workspace --gpus all --env
DISPLAY=unix$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix --name
robot_ai_project_container robot_ai_project:1.0 /bin/bash
```

   2.3. if container status is Exited(0)

```
$ docker start robot_ai_project_container


$ docker exec -it robot_ai_project_container /bin/bash
```

※If you encounter a GPU error during doing docker run, you can refer it to

- **(docker: Error response from daemon: could not select device driver "" with capabilities: [[gpu]].)**

```
$ distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
   && curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo
apt-key add - \
   && curl -s -L
https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.li
st | sudo tee /etc/apt/sources.list.d/nvidia-docker.list
$ sudo apt-get update && sudo apt-get install -y
nvidia-container-toolkit

$ sudo systemctl restart docker
```

2.4. (In local) X host execution

```
$ xhost +local:docker
```

3. Code usage instruction

[Training motion primitive]

```

```

[Demo execution]

```
$ cd workspace/

$ python src/main.py --task feed --object banana --style airplane
```

4. Demo visualization

[Skeleton module]

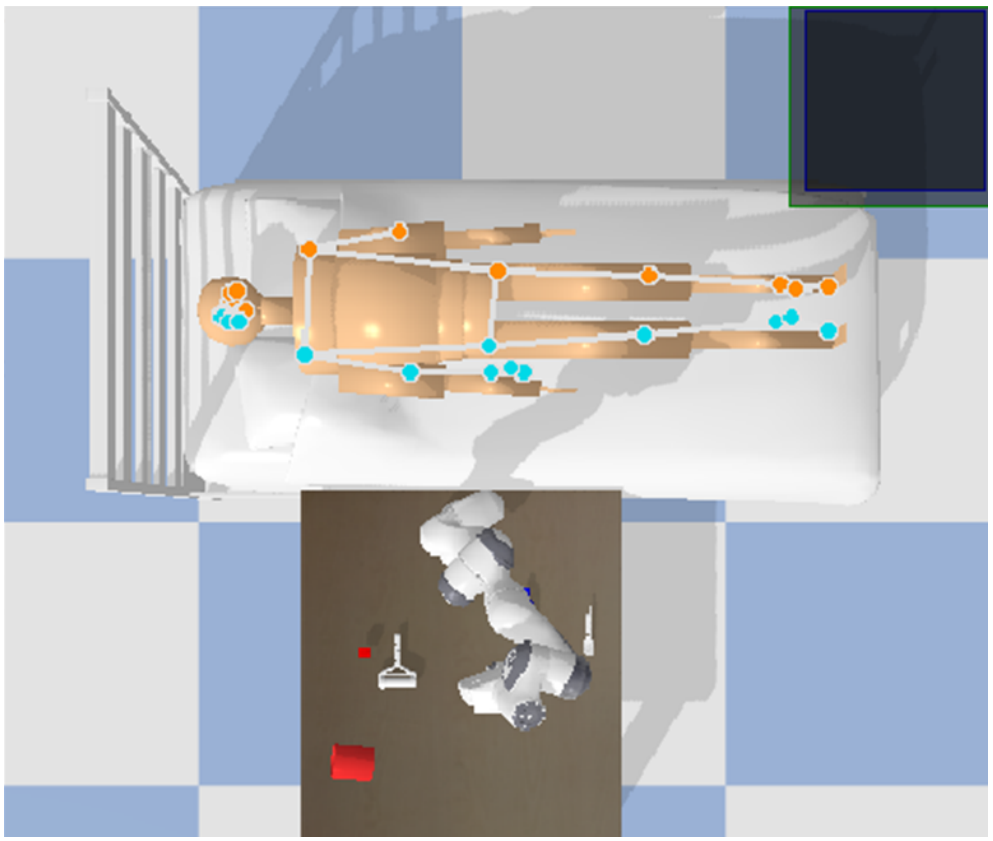


Fig. 1.mediapipe in simulation

This figure illustrates the process centered around a person lying down, as captured by a stationary camera. It begins by showing the steps for using the Skeleton Detection algorithm:
Camera Capture: A fixed camera records the individual lying down.
Skeleton Detection: The Skeleton Detection algorithm precisely detects the person's joints from the recorded imagery, This allows us to pinpoint the location of body parts and joints
Following this, the process proceeds to:
Command Analysis: Subsequently, a manual analysis of the user's command is conducted. The command contains information regarding the robot's actions and is used to determine the target location.
Robot's Target Location Determination: Based on the analyzed command, the robot's ultimate target location is determined. This target location plays a pivotal role in defining the actions that the robot will execute on the person lying down.

[Grasping module]

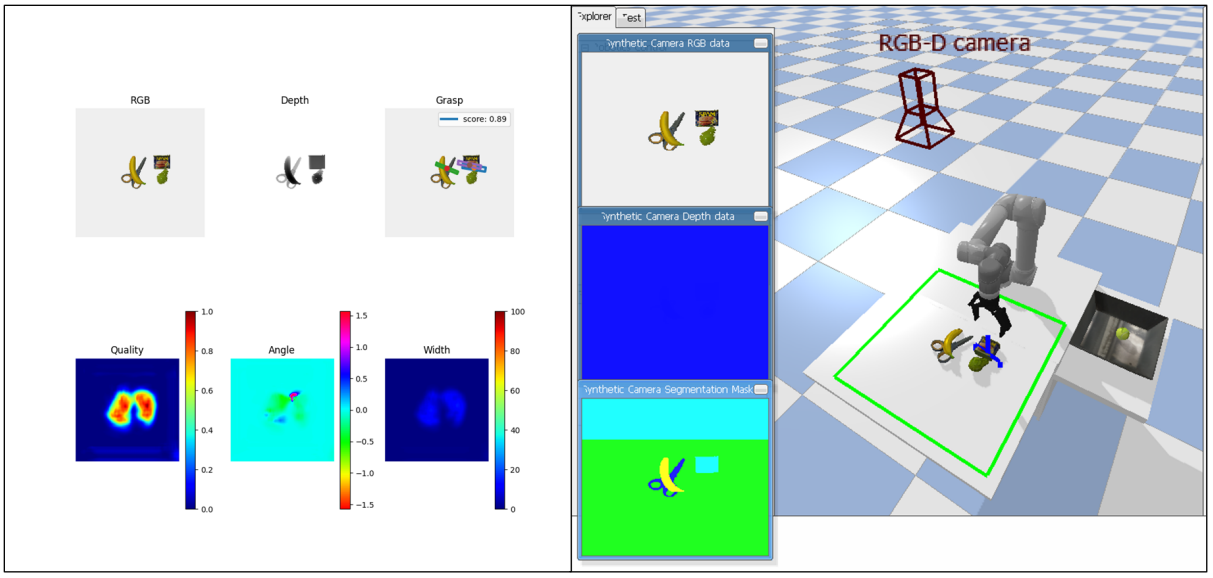


Fig. 2. GR-ConvNet network in simulation

Fig. 2. illustrates the results of detecting grasping poses using the GR-ConvNet network for images in the simulator. The overall flow involves taking RGB and depth images as input, followed by preprocessing. These processed images then pass through the network to extract quality maps, angle maps, and width maps as output. Subsequently, leveraging information from these three maps, the system identifies appropriate grasp poses and attempts grasping.

In the current simulation demo, grasping was performed on multiple objects piled up, prioritizing those with higher scores. In future iterations, the system will be extended to detect and grasp objects corresponding to language commands.

5. Real-world implementation

(video)