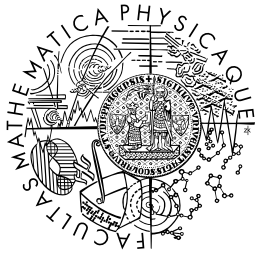

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Lada Kudláčková

Manipulating Objects through Deictic Gesture Recognition

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: RNDr. David Obdržálek, Ph.D.

Study programme: Computer Science

Prague 2024

I declare that I carried out this master thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Manipulating Objects through Deictic Gesture Recognition

Author: Lada Kudláčková

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. David Obdržálek, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Use the most precise, shortest sentences that state what problem the thesis addresses, how it is approached, pinpoint the exact result achieved, and describe the applications and significance of the results. Highlight anything novel that was discovered or improved by the thesis. Maximum length is 200 words, but try to fit into 120. Abstracts are often used for deciding if a reviewer will be suitable for the thesis; a well-written abstract thus increases the probability of getting a reviewer who will like the thesis.

Keywords: gesture recognition, object manipulation, autonomous control

Název práce: Manipulace s objekty pomocí rozpoznávání ukazovacích gest

Autor: Lada Kudláčková

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: RNDr. David Obdržálek, Ph.D., Katedra teoretické informatiky a matematické logiky

Abstrakt: Abstrakt práce přeložte také do češtiny.

Klíčová slova: rozpoznání gest, manipulace s objekty, autonomní řízení

Contents

Introduction

Opening statement: introducing the research field, stating the problem.
My motivation, goals and research limitation.
Overview of the thesis structure.

1 Task Analysis

1.1 Theoretical Background

1.1.1 Basic Definitions

Definition of key words: gesture recognition, object manipulation, autonomous control

Deictic Gesture

A deictic gesture is a gesture that indicates direction or location from the perspective of the person performing the gesture. It refers to a real or virtual environment and its meaning depends on the context. It can be used to specify direction and location or to identify a person or object from the environment. It could often be expressed by adverbs such as "here" and "there" or by demonstrative pronouns such as "this" and "that".

The pointing gesture is the most common deictic gesture. Other examples are gestures based on head movements or eye gaze.

Pointing Gesture

A pointing gesture is performed by extending the arm in the appropriate direction, usually using the index finger or hand to indicate the direction.

Pointing with the index finger is a cross-cultural behavior that can be explained by human development. Infants most commonly use their index fingers for tactile exploration of their environment and they often use the gesture of the extended index finger for a variety of purposes before they acquire its social meaning.

This gesture may represent the pointing of a ray, which is given by, for example, the eyes (as the origin) and the index finger, or it may have a more symbolic meaning, such as when a person points outside their field of vision.

1.1.2 Human-Robot Interaction (HRI)

brief description of HRI;

remote vs. proximate interactions;

roles of humans and robots in interaction: Supervisor, Operator, Mechanic, Peer, Bystander, Information consumer, Mentor (taxonomy from paper: M. A. X. Goodrich and A. C. Schultz, “Human-Robot Interaction: A Survey,” Foundations and Trends R© in Human- Computer Interaction, vol. 1, no. 3, pp. 203–275, 2007.);

areas of application: industrial, search and rescue, medical, social, ...

1.2 Task description

1.2.1 Pick and Place Task

Performing 'Pick and Place' using pointing gestures:

Pick:

- Determine a object that was selected with a pointing gesture
- Navigate close to the object
- Identify the object and compute its exact coordinates
- Pick the object

Place:

- Determine a target location from a pointing gesture
- Navigate close to the location
- Place the object to the location

1.2.2 Task Specification

Requirements and restrictions:
gesture recognition based on image processing (available sensors - depth cameras), proximate robot control by single user, no interaction with other robots or humans, static indoor environment (robotic lab), safe manipulation with objects, safe navigation (obstacle avoidance without unnecessary emergency braking), ...

1.3 Goals

1.3.1 Implementation of Mobile Manipulator

Design and implement a mobile manipulator that performs 'Pick and Place' tasks according to the given requirements;

1.3.2 Comparison of deictic gestures types

Metric: the distance between the correct coordinates (of the selected object or location) and the intersection of the pointing ray and the floor.

Experiment with different ways of using deictic gestures:
- a pointing ray calculated from a pair of skeleton coordinates (head - hand, elbow - wrist, shoulder - wrist)
- pointing with or without visual feedback (pointed ray shown in rViz)

2 The state of the art

2.1 History of Gesture Recognition

Summary of gesture recognition techniques; historical development of sensors; ...

2.2 Localization and navigation with deictic gestures

These are some (not all) examples of what I want to mention here:

Deictic gestures for multi-robot systems

Paper:
B. Gromov, L. M. Gambardella and G. A. Di Caro, "Wearable multi-modal interface for human multi-robot interaction," 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Lausanne, Switzerland, 2016, pp. 240-245, doi: 10.1109/SSRR.2016.7784305.

Use of the pointing gesture for localization

Paper:
B. Gromov, L. Gambardella, and A. Giusti. Robot Identification and Localization with Pointing Gestures. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 3921–3928 https://people.idsia.ch/ gromov/repository/gromov2018robot.pdf

3D Motion planning with pointing gestures

Paper:
B. Gromov, J. Guzzi, L. Gambardella, and A. Giusti. Intuitive 3D Control of a Quadrotor in User Proximity with Pointing Gestures. IEEE International Conference on Robotics and Automation (ICRA), 2020 https://people.idsia.ch/ gromov/repository/gromov2020intuitive.pdf

2.3 Interpretation of gestures

Papers:

Chaudhary, A (2018). Robust Hand Gesture Recognition for Robotic Hand Control. Springer. ISBN 978-981-10-4798-5 https://doi.org/10.1007/978-981-10-4798-5

Alikhani, M., Khalid, B., Shome, R., Mitash, C., Bekris, K.E., Stone, M. (2020). That and There: Judging the Intent of Pointing Actions with Robotic

Arms. AAAI.https://ojs.aaai.org//index.php/AAAI/article/view/6601

2.4 Object detection with pointing gestures and speech recognition

Li-Heng Lin, Yuchen Cui, Yilun Hao, Fei Xia, Dorsa Sadigh (2023). Gesture-Informed Robot Assistance via Foundation Models. https://arxiv.org/abs/2309.02721

A. Ekrekli, A. Angleraud, G. Sharma, R. Pieters (2023). Co-speech gestures for human-robot collaboration. https://arxiv.org/abs/2311.18285

3 Gesture Based Robot Control

3.1 Gesture Based "Pick And Place"

Two gestures are needed to execute the "Pick And Place" task successfully: a pointing gesture to indicate the position and another gesture to confirm it.

During the selection of the object and the target location, the user should face the camera, with all objects lying on the floor between him and the camera. The scene is displayed on the rViz screen. Once the objects are detected by the vision system, their images are marked in blue.

The user can then initiate the selection of an object by pointing at it with his right hand. To confirm the gesture, the user raises his left hand while still pointing at the object.

The object closest to the intersection of the pointing ray and the floor is selected. Its image is marked in red. The target location of the selected object can then be determined. The user determines the location in the same way as before: by pointing to the location and raising his hand.

The target location is on the floor and has to be selected inside the safety frame that is shown in the rViz. The frame represents a space that is safe for the robot to move around, there are no obstacles except for the detected objects.

Once the target location is selected, it is marked in red in rViz, gesture detection is completed and the resulting data is sent to the robot.

3.2 Gestures

3.2.1 Pointing Gesture

The user can choose from three types of pointing gestures. Each type is represented by a pair of joints that determine the pointing ray. The first joint in the pair is the origin and the second determines the direction of the ray:

- Shoulder, wrist (default option).
- Elbow, wrist.
- Head, hand.

The pointing gesture indicates the point where the pointing ray intersects the floor and allows selection of the object or target location.

There is an option to show or hide the corresponding pointing ray. If the ray is visible, it is displayed from its origin to the intersection with the floor.

The pointing gesture has to be performed with the right hand and the first joint has to be positioned higher than the second. These constraints help to reduce the number of falsely detected gestures.

3.2.2 Raising Hand Gesture

The hand gesture consists of lifting the hand. It has to be performed with the left arm and the hand has to be raised above the head.

When pointing with the right arm, the user confirms the pointing gesture by raising the left hand. If no pointing gesture is performed in the moment, the raising hand gesture is ignored.

3.3 Gesture Recognition with ORBBEC Astra SDK

3.3.1 Limitations

ORBBEC Astra SDK provides tools for skeleton recognition and person tracking. The maximum distance for skeleton recognition is 4 meters. Multiple persons can be tracked at the same time.

The skeleton is represented by a set of joints and their positions. The head corresponds to single joint, eye positions and other details are unavailable.

Three joints are given for each arm: shoulder, elbow and hand. The fingers are not recognizable.

SDK also supplies the detection of the grip gesture. I considered using it as a confirmation gesture but preferred the hand raising gesture because the grip was often not detected.

3.3.2 Occlusion

The most common cause of gesture recognition errors is occlusion. Only relatively small objects (no taller than 20 cm) were used for the experiments. If the objects were larger, many false detections occurred because the objects often obscured parts of the person's body and the joints of the corresponding skeleton were not correctly identified.

Occlusion also often occurs when more than one person is in front of the camera. Therefore, it is better to perform the Pick And Place task when only one person is in the scene because then gesture recognition is most robust.

However, if more than one person is present, each person can perform a pointing gesture to select an object or target location.

3.3.3 Recommendations

When I tested skeleton recognition with a single person and small objects, most of the errors were caused by the person's posture.

For best results, the person should be facing the camera, not turning the body or crossing his limbs, as this can lead to errors such as interrupted tracking of the person, misidentification of joints, and false recognition of gestures.

In case of difficulties with skeleton recognition, it may help if the person moves closer to the camera or extends his arms out to the sides.

4 Design of Robotic System

4.1 Basic Structure

The proposed robotic system contains two main components: a vision system and a mobile robotic manipulator. The vision system is static, camera remains at the designated location during the task.

The robot starts at the initial position where it waits for messages from the vision system. Once the object and target position are selected and a result message is received, the robot navigates to the object, moves it to the target position and returns to the initial position.

4.1.1 Vision System

The main purpose of the vision system is to interpret the environment: to detect objects and the skeleton of a person with its gestures. The position of the objects and the tracked person is limited - outside a given frame, the detection is unreliable.

For the ORBBEC Astra, the distance must be less than 4 meters. Therefore, I decided to use a vision system separate from the mobile robotic manipulator. Otherwise, with a camera attached to the robot, only objects very close to it could be detected.

The depth camera is connected to a laptop where the input data from the camera is processed. The resulting message is sent to the robot's desktop computer.

4.1.2 Mobile Robot Manipulator

The robot consists of a mobile vehicle with a robotic arm and a gripper.

The vehicle is equipped with a laser scanner that enables localization and safe autonomous navigation in the environment. The on-board computer serves as the robot's ROS master and is connected to a desktop computer.

The arm with gripper is attached to the vehicle. It needs to be set up so that there are no collisions with the robot or the floor during manipulation. The reach of the arm should also be limited so that it does not move within the field of view of the scanner, as this would trigger an emergency stop.

The arm computer controls both the arm and the gripper and is available via ROS and Dashboard Server.

4.2 Hardware

4.2.1 Neobotix MP-500

The MP-500 mobile robot is a differential-wheeled robot with two large drive wheels and one small one at the rear. It is one of the most robust Neobotix mobile robots with a weight of 70 kg.

Its main components are a mobile platform, laser scanner, on-board computer, battery pack, manual charger and wireless joystick. Additional components can be attached to the mobile platform.

The robot can be used for material transport, with a load capacity of 80 kg. It is designed for indoor operation and has a speed of up to 1.5 m/s.

A Sick S300 safety laser scanner with a maximum range of 30 meters is mounted in the front of the mobile platform. The scanner provides data that is used for localization, navigation and collision avoidance.

Detection of a person or obstacle in the safety field immediately triggers an emergency stop.

4.2.2 Robotic Arm UR5

The Universal Robots UR5 manipulator consists of a robotic arm, a control box with a teaching pendant and a battery.

The six-axis arm is composed of extruded aluminum tubes and rotational joints (Base, Arm, Elbow, Wrist 1, Wrist 2, Wrist 3). The Base is the first joint of the kinematic chain in which the arm is mounted to a fixed surface or a mobile platform. The last joint to which the tool is attached is Wrist 3.

All joints have a motion range of 360 degrees. The reach of the arm is 0,85 m from the center of the base, the area directly above and below the base is out of reach. The weight is 18.4 kg and the maximum payload is 5 kg.

The teaching pendant provides a GUI for control of the arm, commands can also be sent remotely using dedicated ports.

4.2.3 Weiss Robotics GRIPKIT

A two-finger gripper is connected to the UR5 arm using the Weiss Robotics GRIPKIT module. Its maximal opening stroke is ? TODO

4.2.4 ORBBEC Astra camera

TODO

4.2.5 Computers and network

For the vision system, an Acer TravelMate P214 notebook is used. The ORBBEC Astra camera is connected via USB.

A Lenovo ThinkStation P330 desktop computer controls the mobile manipulator. It is connected to the Neobotix MP-500 mobile robot via an Ethernet cable.

The connection between the computers is established via WiFi, messages are sent using SSH.

4.3 System Software

TODO
Ubuntu 20.04.
Robot Operating System, Noetic.

5 Implementation

5.1 Vision system

5.1.1 Image Processing with ORBBEC Astra Camera

The image data is sent from the camera to the connected laptop for processing.

There is the ROS Master and several other individual ROS nodes running on the notebook. Some ROS nodes are involved in image processing, while others provide tools such as geometric calculations or displaying detected objects and skeletons on the rViz screen.

ROS nodes communicate with each other using ROS messages and share information about the progress of their subtasks.

I used two main tools to process the camera data: the ROS Astra camera driver "ros_astra_camera" for object detection and the "ORBBEC Astra SDK" for skeleton detection.

Both use OpenNI as an intermediate layer to access the camera data. I couldn't run them at the same time because it led to runtime errors, so I decided to split the process into two separate phases. First, the "ros_astra_camera" driver is started. Once all objects are detected, the driver stops.

In the second phase, data is exposed using the "ORBBEC Astra SDK" until both pointing gestures are confirmed and the result is sent to the robot. Since gesture recognition is performed within the ROS system, additional ROS packages were needed to publish the body tracking data provided by the SDK as ROS messages.

5.1.2 Installation

ROS Noetic

The notebook with Ubuntu 20.04 was used, for which the recommended version of the ROS distribution is ROS Noetic. I followed the instructions from http://wiki.ros.org/noetic/Installation/Ubuntu to download and install the ROS Noetic package.

Astra and OpenNI SDKs

For the ORBBEC Astra camera, I needed to install the Astra SDK and the OpenNI SDK for Linux.

Both SDKs are available at https://www.orbbec.com/developers.

ROS Driver for Astra camera

I downloaded the ROS driver package
from https://github.com/orbbec/ros_astra_camera and installed the dependencies according to the instructions on http://wiki.ros.org/astra_camera.

The "ros_astra_camera" package supports the ROS distributions Kinetic and Melodic. I needed to find and test multiple versions of the "ros-*-libuvc-*" libraries, as they were not released specifically for ROS Noetic.

This problem was already solved on the ORBBEC GitHub page, so I followed the advice and installed the missing dependencies using:

```
$ apt install ros-noetic-rgbd-launch libuvc-dev
```

I built the package with "catkin_make" command and was able to run a few code samples that show the camera data on the screen.

ROS Packages for Image Processing

I downloaded three ROS packages from the Shinsel Robots repository on https://github.com/shinselrobots.

The "pcl_object_detection" package allows object detection in the camera data provided by the ROS Astra driver using the Point Cloud Library.

The "astra_body_tracker" and "body_tracker_msgs" packages publish body tracking data from the Astra SDK as ROS messages.

Several environment variables have to be set to indicate the paths to the AstraSDK and OpenNI subfolders.

For example, if "/home/user/AstraSDK" is the folder containing the Astra SDK and "/home/user/OpenNI-Linux-x64-2.3.0.66" is the folder containing the OpenNI SDK, the settings can be made by running these commands in the terminal:

```
$ export ASTRA_SDK=/home/user/AstraSDK
$ export ASTRA_ROOT=/home/user/AstraSDK
$ export ASTRA_SDK_INCLUDE=/home/user/AstraSDK/include
$ export ASTRA_SDK_LIB=/home/user/AstraSDK/lib
$ export OPENNI2_INCLUDE=/home/user/OpenNI-Linux-x64-2.3.0.66
$ export OPENNI2_REDIST=/home/user/OpenNI-Linux-x64-2.3.0.66/Redist
```

5.1.3 The Vision System Code

Catkin Workspace

Catkin is the official build system for ROS. Project packages that are placed in the same catkin workspace can be built all at once.

My catkin workspace folder contains following ROS packages:

- ros_astra_camera
- task_execution
- rviz_screen
- pcl_object_detection
- pointing_gesture

Program Overview

The main launch file is task_execution.launch. It starts the ROS Astra driver, rViz and other ROS nodes involved in the task: task_execution_node, pcl_object_detection_node and pointing_gesture_node.

The task_execution_node subscribes to ROS messages "pcl_object_detection/detected_objects" and "body_tracker/intersection".

The "pcl_object_detection/detected_objects" message contains an array of coordinates of the detected objects, the "body_tracker/intersection" message contains the coordinates of the intersection of the pointing ray and the floor.

When the intersection message is received for the first time, the nearest object is calculated. The object is represented by its index in the detected object array, which is then published in the "task_execution/pointed_object_index" message.

The second received intersection message indicates the target location. Once received, the node creates a result file and writes the coordinates of all detected objects, the coordinates of the target location and the index of the selected object.

Then the node connects to the robot's computer using SSH, transfers the file there and remotely starts the robot's main program.

Object Detection

Package pcl_object_detection:
https://github.com/shinselrobots/pcl_object_detection
ROS node for detecting objects on a flat surface, using Point Cloud Library (with the ros_astra_camera package).

Modification of this package:
added limitation of size and position of objects (detection frame) to avoid false detection;
added custom messages, rViz markers, ...;

Subscriber of ros_astra_camera topic (point clouds) and selected object topic. Publisher of detected objects topics (coordinates, ...) and point cloud topics (clusters, planes, ...) for rViz.

5.1.4 Gesture detection

Packages: astra_body_tracker:
https://github.com/shinselrobots/astra_body_tracker
Publisher of ROS topic for body tracking information (from the ORBBEC SDK).

pointing_gesture:
modified astra_body_tracker package to get skeleton data;
added code to detect gestures, rViz markers, ...

Publisher of pointing_gesture topic (as geometry_msgs).

5.1.5 Stream switching

Problem: skeleton data was not provided in the ORBBEC SDK (without license), I needed to switch between data streams (using custom ROS messages): launch ros_astra_camera driver and pcl_object_detection node, when the object detection is complete, stop the stream and run the ORBBEC SDK with pointing_gesture package to get body tracking.

5.1.6 Object selection and target location

task_control_node (will be renamed):
Subscriber to object detection and pointing gesture topics;
provides calculations of pointing ray intersection and selection of object.
Sends data to mobile manipulator PC over SSH (coordinates of objects and target location, info about selected object).

TODO - some notes

Why I choose ORBBEC Astra camera over Kinect ONE (v2):
difficult installation of tools and libraries for a ROS Interface to the Kinect One (dependencies on ROS Hydro/Indigo distribution, no available packages for ROS Noetic).
new wrapper!!!!!!!!! Installation of ORBBEC SDK for Linux and dependencies (OpenNI2, libsfml-dev, ...).

Package ros_astra_camera:
https://github.com/orbbec/ros_astra_camera
OpenNI2 ROS wrapper for Orbbec 3D cameras.

5.2 Navigation of Autonomous Vehicle

5.2.1 Installation

Neobotix:
Packages: https://github.com/neobotix/
ros-noetic-amcl, ros-noetic-map-server, ros-noetic-move-base, ...

5.2.2 Map of Environment

Mapping procedure, selecting the map for navigation, visualization with RViz...

5.2.3 Navigation to Goal

Goal definition, movement (path, obstacle avoidance, ...).

5.3 Object Manipulation

5.3.1 Installation

Universal Robots:

Packages:
Universal_Robots_ROS_Driver https://github.com/UniversalRobots/Universal_Robots_ROS_

Universal_Robots_Client_Library
https://github.com/UniversalRobots/Universal_Robots_Client_Library

ur5_moveit_config
https://github.com/ros-industrial/universal_robot/tree/noetic-devel/ur5_moveit_config

5.3.2 Mobile Manipulator URDF

URDF for Neobotix, UR5 and gripper.

5.3.3 MoveIt Setup Assistant

How to create config and set up arm positions.
How to set up arm limits.
Simulation in rViz.

5.3.4 Code

ur_robot_driver;
ROS.urp;
move_it_planning;
trajectory commands;

5.3.5 Objects coordinates

approximate coordinates of objects obtained from the vision system;
robot navigates to objects;
exact objects coordinates from LIDAR (lidar_scan topic subscriber).

6 Experiments

Experiments descriptions:

Experiments with different ways of using deictic gestures:

- a pointing ray calculated from a pair of skeleton coordinates (head - hand, elbow - wrist, shoulder - wrist)
- pointing with or without visual feedback (pointed ray shown in rViz)

Experiments measurements:

...

result evaluation; what went wrong; future work, possible improvements

Conclusion

Results of experiments - summary.
Which gestures are well recognised by Astra camera;
most accurate pointing gestures - compare results with related work.

Suggestions for improvement.

7 Appendix

List of Figures

List of Tables

List of Abbreviations

A Attachments

A.1 First Attachment