Autobots VIP Spring 2023
Final Report

Anthony Miyaguchi
acmiyaguchi@gatech.edu

2023-05-03

Contents

Summary

As part of the *Manipulation and Understanding* sub-team, I worked toward building a pick and place simulation using Handy as the base manipulator. I updated and refactored Handy ROS packages to work with ROS Noetic and Gazebo and created a Docker environment to facilitate usage. I reproduced benchmarks from the Grasp Keypoint Network (GKNet) paper using the source code and created ROS modules for inference. I also build the scaffolding for performing pick-and-place tasks, including calibration, workplace setup, perception, and manipulation. Relevant code is available on GitHub in standalone repositories or contributed back into various IVALab repositories.

I had the most collaboration with the following students:

- Nicholas Gilpin - Scene Understanding ROS modules
- Ashylnn Zheng - Validating Handy/D435/GKNet behavior in the lab
- Calvin Truong - Validating Handy/D435/GKNet behavior in the lab
- Tawhid Ahmad - Context around Factory Automation efforts

I also had help from graduate students in the lab:

- Runian Xu - Collaboration on simData, Handy, and GKNet pull requests
- Yiye Chen - Camera extrinsic code

Code Contributions

ivapylibs/camera

Repository: ivapylibs/camera

- [ivapylibs/camera] Refactor directory structure of the camera package #4
  - We move the scripts directory out into the root of the package. We also renamed some D435 modules and used `mayavi` instead of `pptk`.
- [ivapylibs/camera] Use find_packages and remove old requirements.txt #5
  - We use `find_package` and update the `setup.py` to use references to packages via `git+https://`. We review all `ivapylibs` packages and update the `setup.py` files to match.
- [ivapylibs/camera] Update CtoW_Calibrator_aruco with optional aruco_dict parameter #6
  - This allows estimating the camera extrinsic using user-defined values for the ArUco dictionary. The `aruco_dict` parameter is needed because the one used in simulation or physical experiments may not match the hardcoded value in the library.

ivaROS/ivaHandy

Repository: ivaROS/ivaHandy

- [ivaROS/ivaDynamixel] Run 2to3 and add docker for testing #3

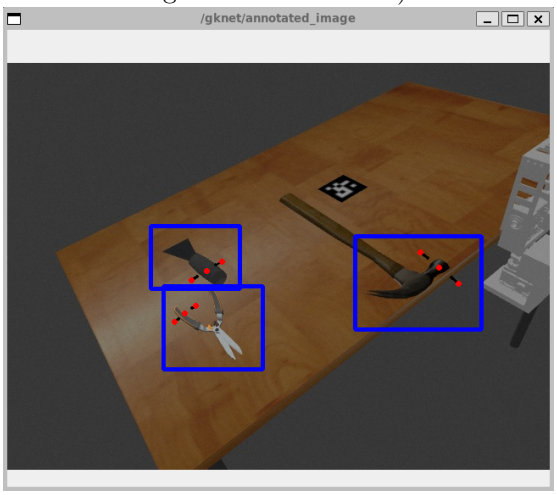
  - This PR enables Handy to run on ROS Noetic and Python 3 by fixing broken serial communication. We rewrote the main routine for writing to and from the serial device for string behavior changes in Python 2 vs. 3. We resolve this by avoiding using strings and passing a valid byte array/memory view to `pyserial`. We use the struct module to decode messages from the serial device.
- [ivaROS/ivaHandy] Add docker-compose configuration for running Handy #3
  - This adds documentation and docker configuration for running Handy. The containers bind against the host machine as if running on the host itself. See the documentation for more details.
- [ivaROS/ivaHandy] Add noetic base image #5
  - We add a new default docker image that uses ROS Noetic as the base. This image validates the changes made in the ivaDynamixel package in a reproducible environment.
- [ivaROS/ivaHandy] Update control, description, and gazebo packages for RViz/MoveIt control #6
  - We fixed mocked Gazebo controllers to work with MoveIt. The controllers now use the same namespaces as the physical robot and behave correctly with MoveIt. The controller scripts and launch files are also updated to work agnostic to simulation or physical robots.

ivaROS/GraspKpNet

Repository: ivaROS/GraspKpNet

- [ivaROS/GraspKpNet] Reproduce training and benchmark results via Docker #3
  - This PR prepares the repository for ROS integration. I rerun the benchmarks of the GKNet paper on available datasets and pre-trained models. I rehost datasets on a public Backblaze bucket (see docs/INSTALL.md). I also refactor the repository structure into a proper Python package that is easier to navigate and use externally. I created a docker image with an updated PyTorch version and forked DCNv2 code to build without a GPU. I also create a docker-compose configuration for running the benchmarks.
- [ivaROS/GraspKpNet] Add ROS module for GKNet perception and inference #4
  - I added a ROS module to process images into their grasp key points. It will dump out a series of key points and an annotated image. The pick-and-place code transforms the grasps into the world coordinate frame.
- [ivaROS/GraspKpNet] Add functionality to rank grasping poses on a per object basis #5
  - This refactors the code to separate annotation into a node and adds filter-ranked poses per object. I've added a small

OpenCV-based utility to stream image topics (which works better than image view via docker) and manually draw rectangles.



ivalab/simData

Repository: ivalab/simData

- [ivalab/simData] Generate preprocessed model files for imgSaver via catkin project #1
  - I refactor a Matlab script that generates model SDF files into the Catkin build process via CMake. The new build script exports the generated SDF files. We can reference the SDF files from a launch file using the `find_package` macro. We use this package extensively from the pick-and-place repository.
- [ivalab/simData] Add collision/physics to rendered models #2
  - With @ruinianxu's help in enabling physics, I also add collision to the model SDF using model meshes. The patch allows better simulation of items and the manipulator in Gazebo.

Autobots-Visman/segmentation

Repository: Autobots-Visman/segmentation

- [Autobots-Visman/segmentation] Add skeleton of packages with initial unit tests and docker configuration #1
  - I helped add boilerplate code to the segmentation/perception code for the sub-team. The PR uses idiomatic ROS module patterns, including basic testing mechanisms.

Autobots-Visman/pick-and-place

Repository: Autobots-Visman/pick-and-place

The Autbots-Visman/pick-and-place repository is the primary deliverable for the Spring 2023 semester. It contains a (mostly) functional demo that integrates GKNet and Handy. It falls short of the pick-and-place goal but demonstrates smaller system components.

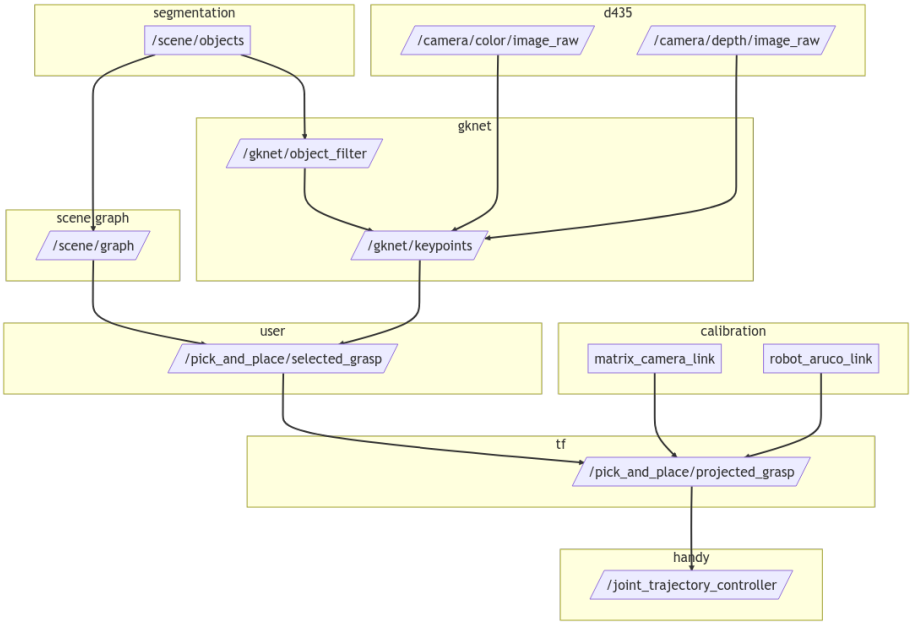


Figure 1: General architecture of modules/nodes involved in the pick-and-place simulation.

- ros/autobots_realsense2_description@02e12b2146
  - This package contains a modified version of urdf files for the realsense2 camera. The main difference in this model is that we can turn off gravity for the camera link. The modified urdf is useful for simulating the camera in a fixed position. The included launch file will configure the topics and align the depth image to the color image.
- ros/autobots_calibration@02e12b2146
  - This package implements a calibration node that uses `ivapylibs/camera` to compute camera extrinsic with a single ArUco on a workspace. It publishes information to tf2 and a latched calibration topic. The package is agnostic to running in simulation or on a physical robot.
- ros/autobots_handy_simulation@02e12b2146
  - This package contains the Gazebo simulation for the Handy manipulator. It includes a launch file that will spawn the robot, tabletop, and camera in a fixed position. It also includes the control scripts for moving the manipulator based on GKNet inference.

Major Points Learned

Software engineering fluency is valuable when synthesizing a system

My time this semester in the VIP taught me the value of having professional software engineering experience under my belt. I have a significant amount of systems experience, be it Linux systems administration, build-system configuration, or distributed systems. It helped me identify various workflow issues in the lab that came between me and a physical pick-and-place simulation using Handy. I was comfortable diving deep to resolve issues with serial communication, software differences between major revisions of ROS, and existing robotics software found across various repositories.

It was nice to be recognized for my help when resolving issues, especially in other folks' workflows. For example, I helped a student in the lab work use a docker and a newer version of ROS to reduce overhead with the Mary surveillance robot. Many of my pull requests were less about functional change and more about reducing the cognitive burden for people interested in using the software in the future.

It was also relatively straightforward to build non-trivial modules in ROS because I've had exposure to large codebases and have seen many of the patterns used in the framework. I had fun while building many of these modules. Still, I recognize that it would have been a more frustrating experience without the fluency needed to assemble the various subsystems.

Remote collaboration without access to the hardware is challenging

Working on robotics without a robot is very hard. I work best when I have an accurate mental model of how all of the software I touch interacts. I only had this mental model once I saw Handy in person during my February Atlanta trip. Up to this point, it was difficult to understand how code needed to be structured to solve particular manipulator tasks. After seeing the robot in action and the software needed to run, it was much easier to see the path forward.

After leaving Atlanta, I found the small interactions with people in the lab made it easier to communicate with people online. I learned that pairing with people over a voice call was an efficient use of time and helped the other person get over what I consider a high barrier to entry. Meeting people made it easier to merge pull requests, e.g., simData and GraspKpNet PRs. Conversely, it was difficult when the code I wrote on my computer did not work on the lab computer for one reason or another. Various issues around Docker, CUDA driver versions, and `xauth` caused problems. It is challenging to figure out the best way to write code for someone else to run, without having access to the actual hardware.

It's been a valuable learning experience working both in-person and online, taking away unexpected perspectives.

Expectations

My experience from the Fall tempered my expectation for the VIP this semester. I was ready to be the only online graduate student in the course and to push along at my own pace. I had been planning to visit Atlanta since the end of the Fall semester, so I expected that I would have an opportunity to play around with some of the manipulators in the lab.

I set up three goals from the last semester:

- Finish the "VisMan Learning Adventures" tutorial series and build out skeleton code as an exercise for future students.
- Collaborate with the Multi-Robot Coordination team on the Factory Automation task, particularly around Gazebo simulations.
- To perform a literature review and pose and research problem.

These goals were a valid waypoint, but what happened in practice differed from the plan. I was actually one of four OMSCS students, and I had a surprising number of interactions with students online and in-person. I also dove deeper into the depths of software development and did not have much time to help out with Factory Automation or to complete the "VisMan Learning Adventures" as-is. Overall, my VIP experience this semester differed from my initial expectations, but I gained valuable skills and unexpected opportunities for collaboration.

What If: Additional Semester

I would focus on completing the pick-and-place demo with Handy and GKNet if I had another semester to expand or improve my contributions to the VIP. There are slight differences in scope depending on whether I have one or three credit units to spend.

I plan on taking another semester with one credit unit this Fall, which involves finishing integrating the simulation components. I had a fruitful experience collaborating with a few other students as they ran the simulation code in the lab and piece-meal tested various components. Concretely, here are a few of the tasks needed:

- Validating the calibration module against a physical RGB-D camera and workspace.
- Validating GKNet against a physical RGB-D camera with a variety of objects.
- Validating TF transforms of grasp key points against calibration extrinsic with manipulator trajectories.
- Implement more realistic weights and center of gravity on simulated tools.
- Implement unit tests for calibration and simulation modules.
- Implement simulated grasping with a manipulator.
- Implement target location to move manipulator.

The overall goal is to complete a physical pick-and-place experiment. Most

of the written code for running the physical experiments (launch files, scripts, documentation) would come from students with access to the lab. The secondary goal would be to integrate the manipulator simulation into the broader context of an assistive task, such as Factory Automation. However, this is contingent on having a suitable code for simulating the broader environment.

If I had another three-credit unit semester, I would also dive deeper into improving model deployment for GKNet. In particular, GKNet requires a GPU with CUDA, which limits experimentation inside a simulation. Spending about a third of the time working on model inference optimization and distillation would be interesting. We can deploy modern vision models to single-board computers. It would also be helpful to deploy Handy/GKNet modules to a small computer. In addition, I would also bootstrap other subgroups with ROS/Gazebo boilerplate that they can use for development. For example, the Factory Automation team has a few scripts for controlling TurtleBots that are limited to execution on a physical robot. The lack of simulation lengthens the development loop and makes it more challenging to realize more exciting demonstrations.

What If: Redo Semester

If I had to redo the semester, I could have been more efficient about progress by spending more time on boilerplate code for other teams and dropping the in-progress work I had from last semester. My critical juncture this semester was the time I spent in Atlanta in February. This time let me see the state of the lab and interact with students in the VIP and the lab. It was valuable for knowing the robot's environment and software stack and helped me figure out the most impactful work.

Knowing what I learned in the lab, I would drop experimental work following the "VisMan Learning Adventures". The tutorial was a great starting place to understand ROS and Gazebo, but it was divorced from practical code bases. I spent the first four weeks of the semester generally aimless in this regard.

The second thing I would do upfront to be more efficient is to set up various repositories that other students can work out of. For example, the Factory Automation team started a small repository with some scripts for ArUco tag detection. It would have been helpful to configure this with what I consider helpful boilerplate:

- A ROS module for the main functionality
- A testing harness for unit tests
- A Dockerfile and docker-compose.yml configuration for testing

It's hard to see how else I could have been more efficient in choosing impactful work. This semester has been fruitful, if not ending on a somewhat unsatisfying conclusion.

Research Journal

Trello

See the Trello card for Anthony's Spring 2023 Work for general progress this semester.

Progress Presentations

Progress presentations effectively capture most of the work in the research journal in two-week increments from the semester's midpoint. These presentations provide a more comprehensive overview of my work in the larger team context.

You can find the presentations in the appendix or linked below:

- 2023-03-14 - VisMan Progress Presentation 1
- 2023-04-04 - VisMan Progress Presentation 2
- 2023-04-18 - VisMan Progress Presentation 3

Appendix

Slides for the progress presentations are attached to the end of this report.

Autobots VIP Spring 2023
VisMan Progress Presentation 1

Anthony Miyaguchi
acmiyaguchi@gatech.edu

2023-03-14

Outline

- VisMan Subteam Progress
- Manipulators in the Lab
  - Handy, Mary, and Edy
  - Theory of Operation
  - Changes to Handy/Mary Software
- Reproducing GKNet Benchmarks
  - Overview of grasp detection
  - Datasets: Cornell and abridged Jacquard
  - Benchmark results
  - Ideas for further exploration
- Future Work
  - Simulating pick and place with Gazebo and GKNet
  - TSRB Gazebo world

VisMan Subteam Progress

Progress

- Established meetings and general plan for the semester
- Scene graph ROS packages for object detection/segmentation/labeling
  - Functional detector on live webcam feed via YOLO
- Pick and place simulation with Handy and GKNet
  - Simulated RGBD camera tests and model imports

Current Members

Nicholas Gilpin, Anthony Miyaguchi, Calvin Truong, Ashlynn Zheng, Ruinian Xu (advisory)

Join our meetings!

We meet on Teams every Monday at 5:30pm EST, under the *Manipulation and Understanding* channel.

Manipulators in the Lab

Overview

- Handy, Mary, and Edy
- Theory of Operation
- Changes to Handy/Mary Software

Manipulators in the Lab: Handy

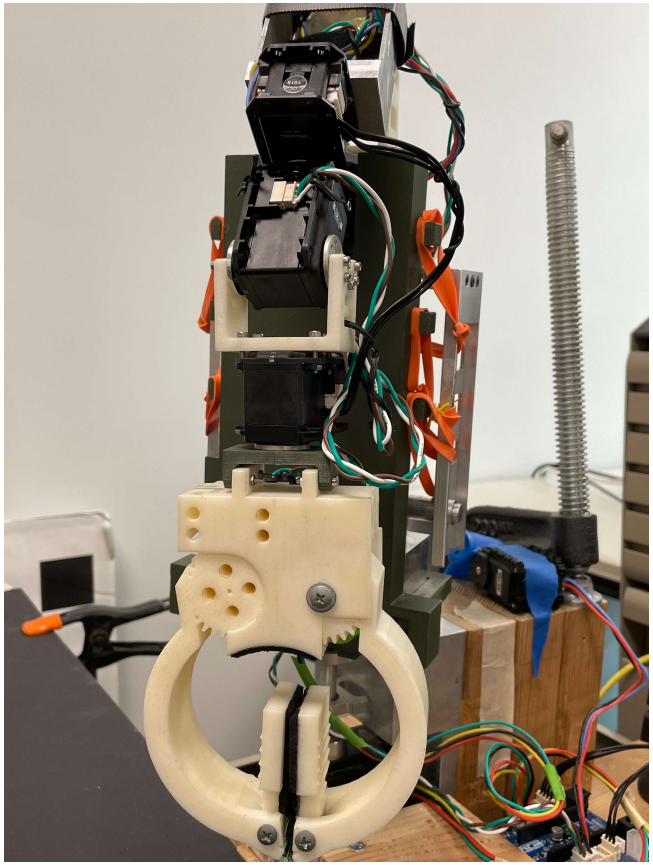


Figure 1: Handy manipulator

- Video of Handy in action

Manipulators in the Lab: Mary

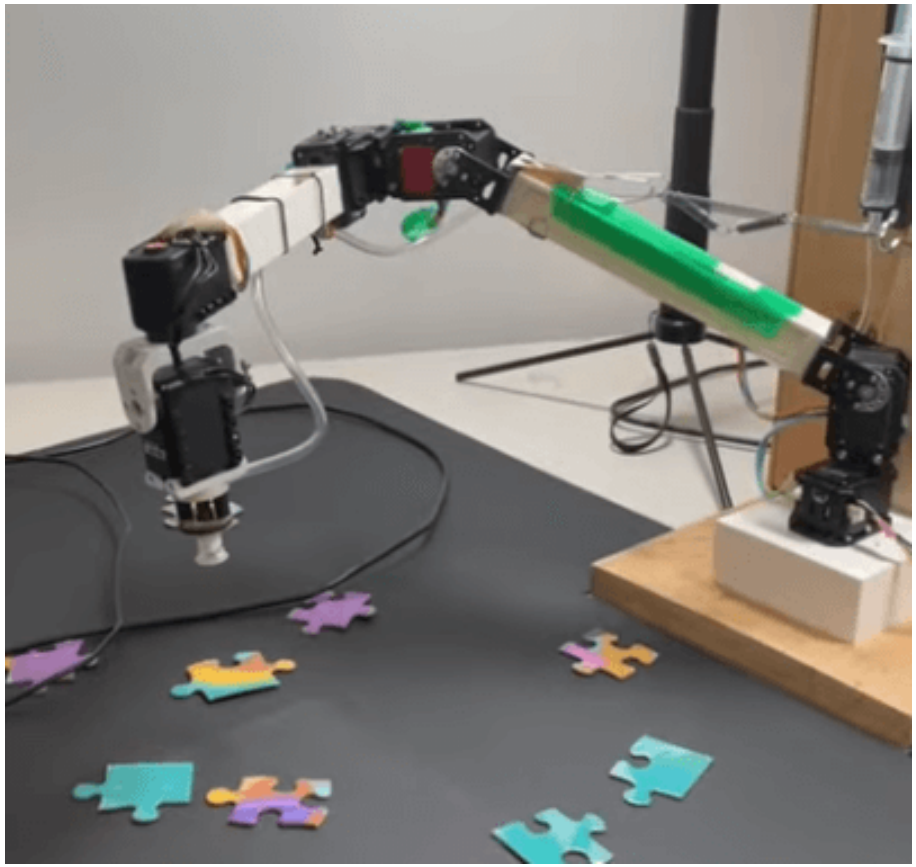


Figure 2: Mary manipulator

- Video of Mary in action

Manipulators in the Lab: Edy

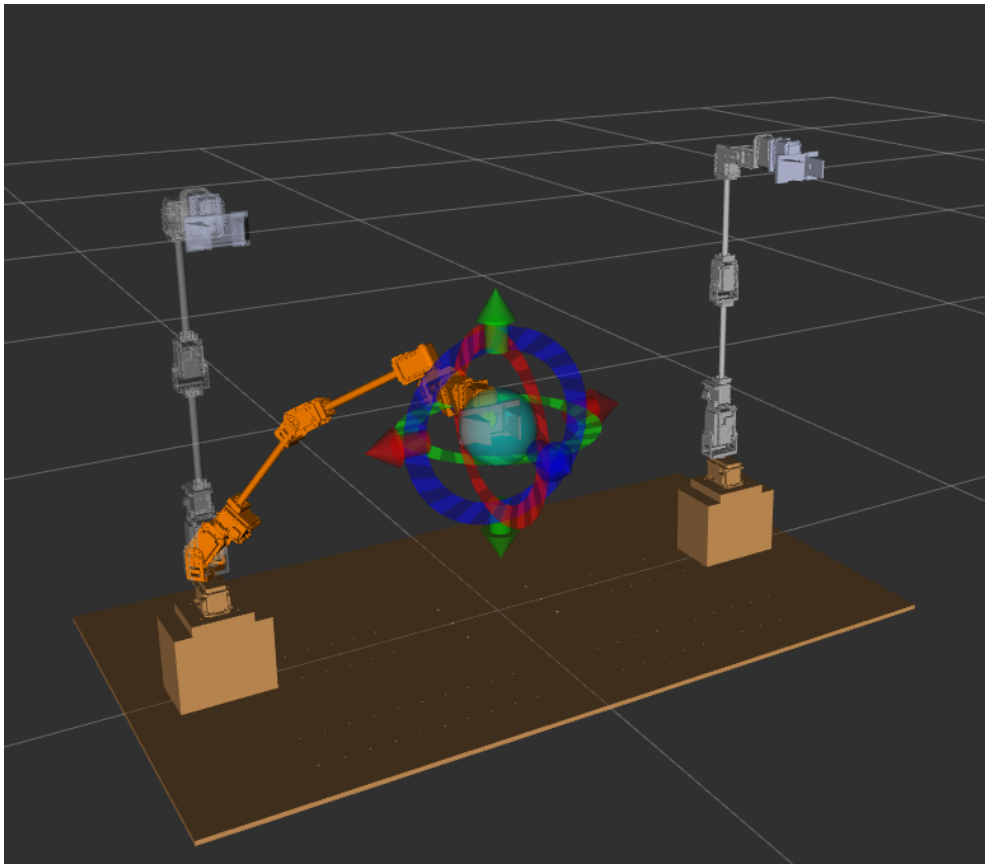


Figure 3: Edy manipulator

Edy is currently awaiting reconstruction. . .

Manipulators in the Lab: Theory of Operation

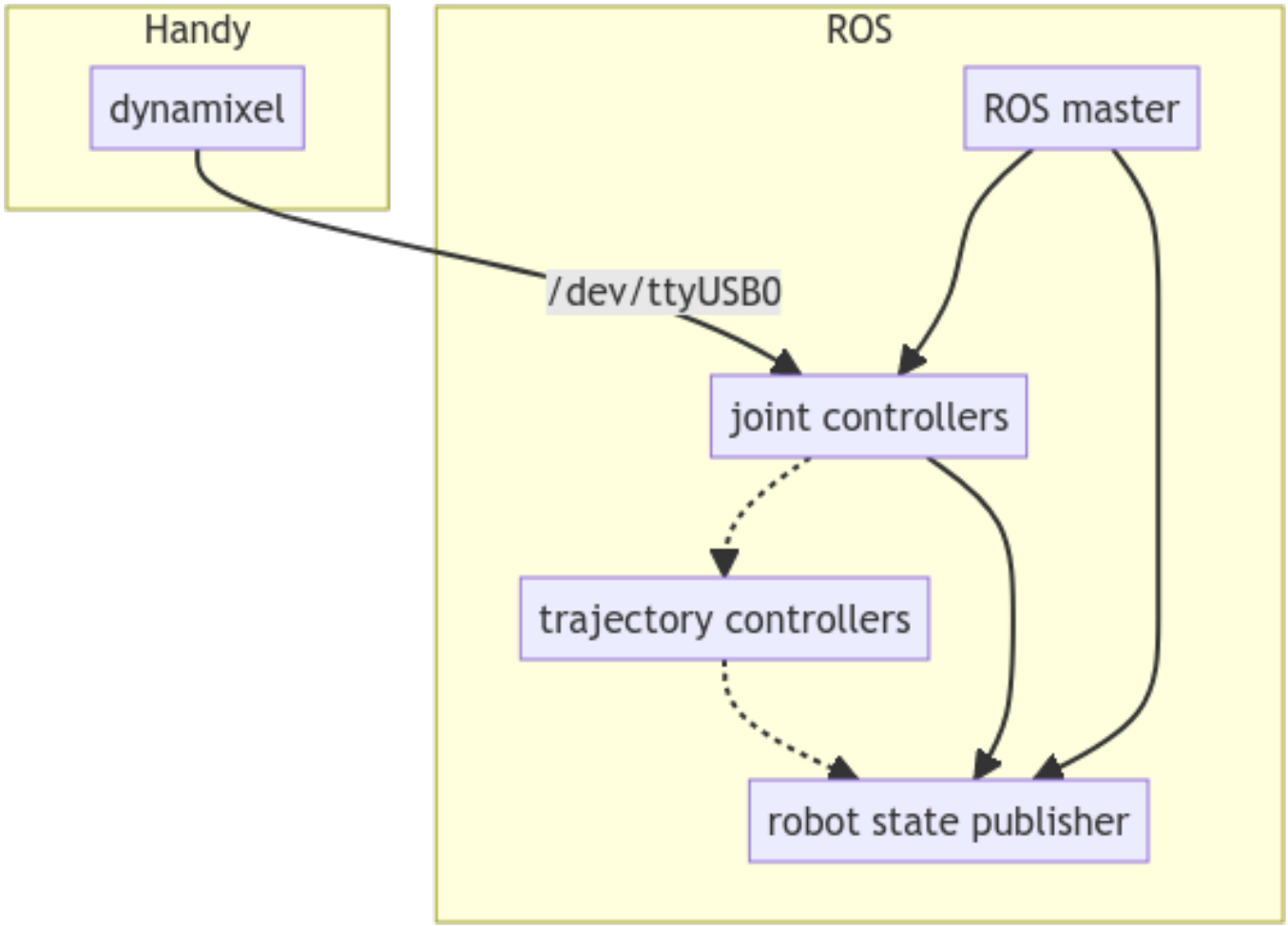


Figure 4: Handy ROS Nodes

Manipulators in the Lab: Theory of Operation (cont.)

Booting up Handy requires starting a number of ROS nodes.

Booting up Handy

```
# start a node to manage controllers
roslaunch finalarm_control controller_manager.launch
# start nodes that correspond to each of the motors
roslaunch finalarm_control start_controller.launch
# publish controllers and link transformations
roslaunch finalarm_description robot_state_pub.launch
# start node for motion planning, collision checking, etc.
roslaunch finalarm_moveit_config move_group.launch
# rviz for visualization
roslaunch finalarm_moveit_config moveit_rviz.launch
```

Manipulators in the Lab: Theory of Operation (cont.)

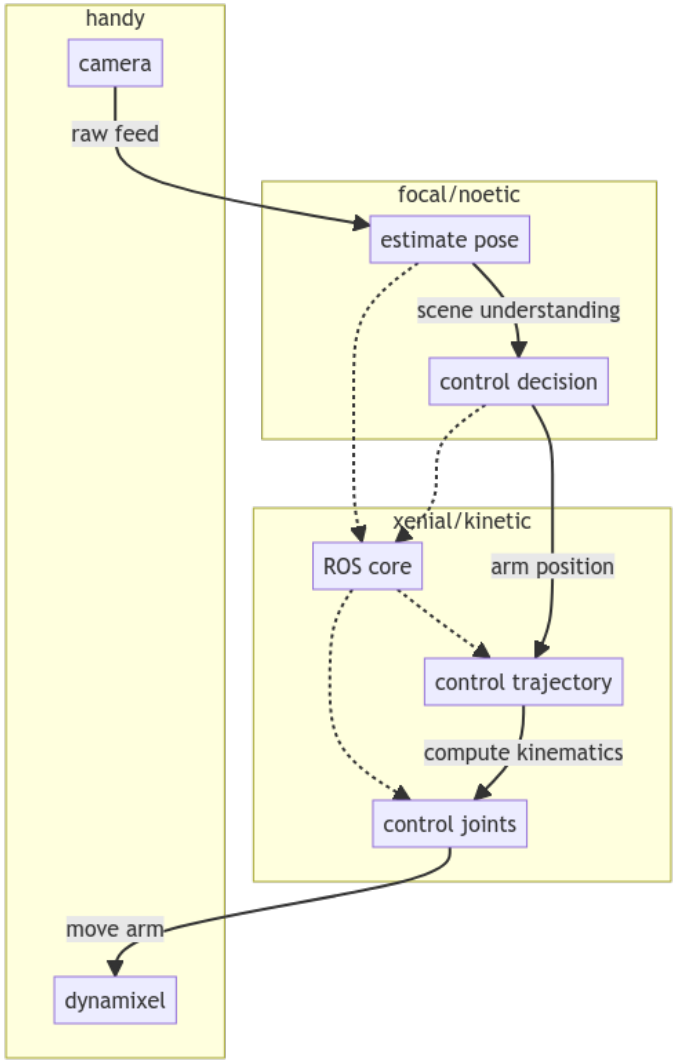


Figure 5: Handy split across networked machines for operation.

Manipulators in the Lab: Handy in Practice

Operational Overhead

- Each ROS node needs to be launched in a particular order in separate processes/terminals.
- Hard dependency on Python 2 with Dynamixel ROS, while vision code is written in Python 3.
- Installing dependencies requires a lot of manual work and knowledge of the Linux/ROS/Catkin.
- Pick and place experiments not easily reproducible; many parameters are hard-coded (camera extrinsics, directory paths, etc.)

Manipulators in the Lab: Changes to Handy/Mary Software

Changelog

- Update ivaDynamixel to support Python 3; fixed broken serial messaging
- Add Dockerfile for Ubuntu Xenial/ROS Kinetic environment
- Add Docker Compose file for launching ROS nodes in networked containers
- Upgrade Handy to run on Ubuntu Focal/ROS Noetic
- Fix various Python packages for pip installs via `git+https` protocol e.g. ivapylibs/camera.
- Upgrade Mary to run on Ubuntu Focal/ROS Noetic.

Manipulators in the Lab: Docker

Figure 6: Docker Logo

Docker - Industry Standard for Deploying Software

What is Docker?

- Tooling to create "containers" with self-contained software
- Runs the same kernel as the host, supported by all major OSes
- Not a virtual machine (at least on Linux/WSL): boots in seconds, uses less resources, and implemented via cgroups/namespace isolation instead of hypervisor

Manipulators in the Lab: ROS Nodes via Docker Compose

Running Handy no longer requires ROS or a Catkin workspace on the host machine.

Running Handy via Docker Compose

```
# build the handy container
docker compose build handy
# start handy ROS nodes
docker compose up

# node for motion planning, collision checking, etc. jobs
roslaunch finalarm_moveit_config move_group.launch
# rviz for visualization
roslaunch finalarm_moveit_config moveit_rviz.launch
```

Reproducing GKNet Benchmarks

Overview

- Overview of grasp detection
- Datasets: Cornell and abridged Jacquard
- Docker builds with GPU support
- Benchmark results
- Ideas for further exploration

Reproducing GKNet Benchmarks: Grasp Detection

Grasp detection is the task of **detecting graspable objects** in an image and **predicting the grasp pose** of the object.

Grasping detection task can be simplified by finding **keypoint pairs** $(x, y, \theta, w)^T$ instead of bounding boxes.

GKNet Paper

Xu, Ruinian, Fu-Jen Chu, and Patricio A. Vela. "Gknet: grasp keypoint network for grasp candidates detection." The International Journal of Robotics Research 41, no. 4 (2022): 361-389.

Reproducing GKNet Benchmarks: Grasp Detection (cont.)

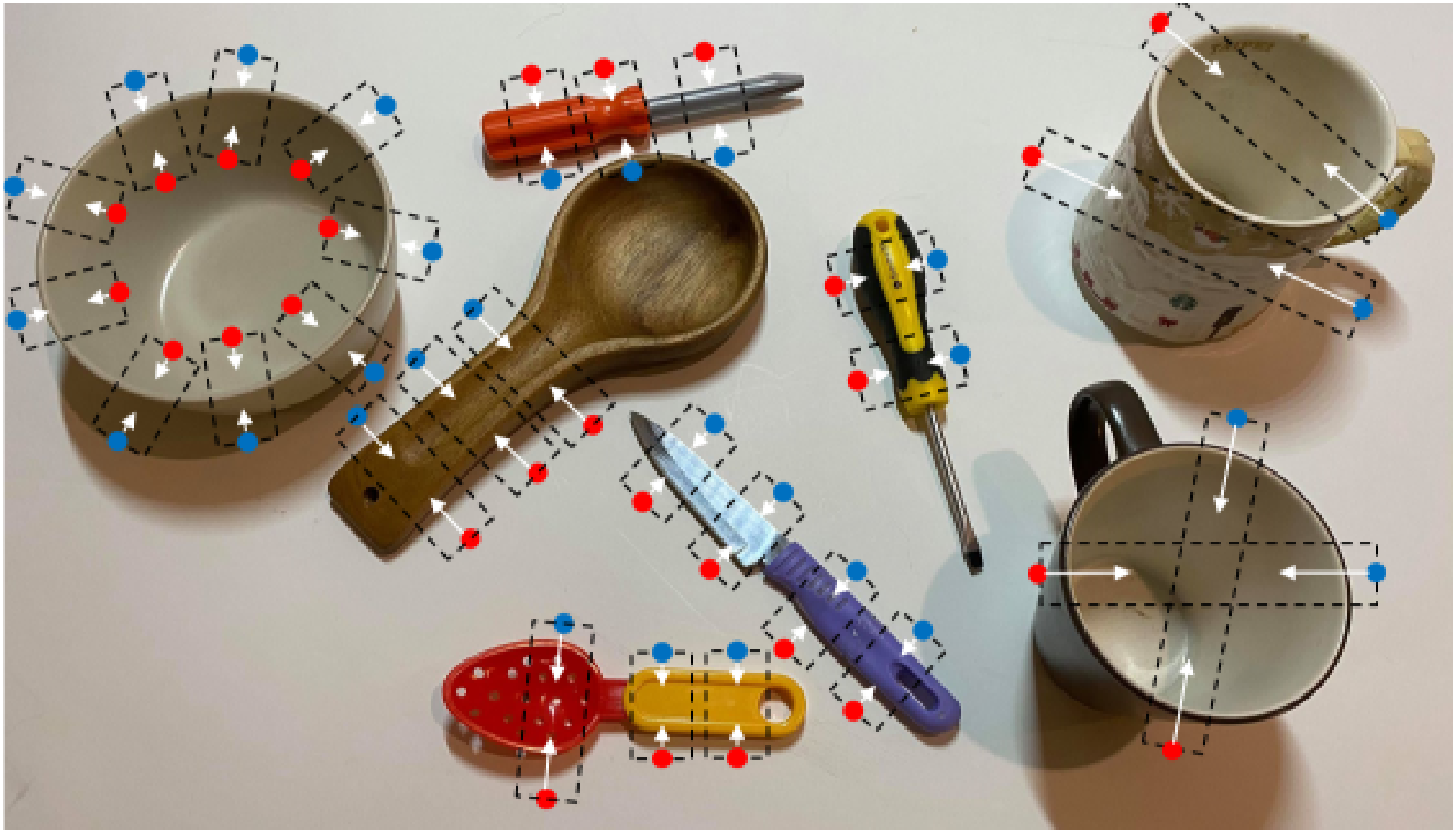


Figure 7: GKNet demonstration

Reproducing GKNet Benchmarks: Datasets

Datasets contain annotated objects in various poses.

```
datasets
|-- Cornell
|   `-- rgd_5_5_5_corner_p_full
|       `-- data
|           |-- Annotations
|           |-- ImageSets
|           `-- Images
`-- Jacquard
    `-- coco
        `-- 512_cnt_angle
            |-- test
            `-- train
```

Training datasets are linked from the ivalab/GraspKpNet GitHub repository.

Reproducing GKNet Benchmarks: Cornell Dataset

885 RGB-D images of 244 objects with 5,110 positive and 2,909 negative grasp annotations.

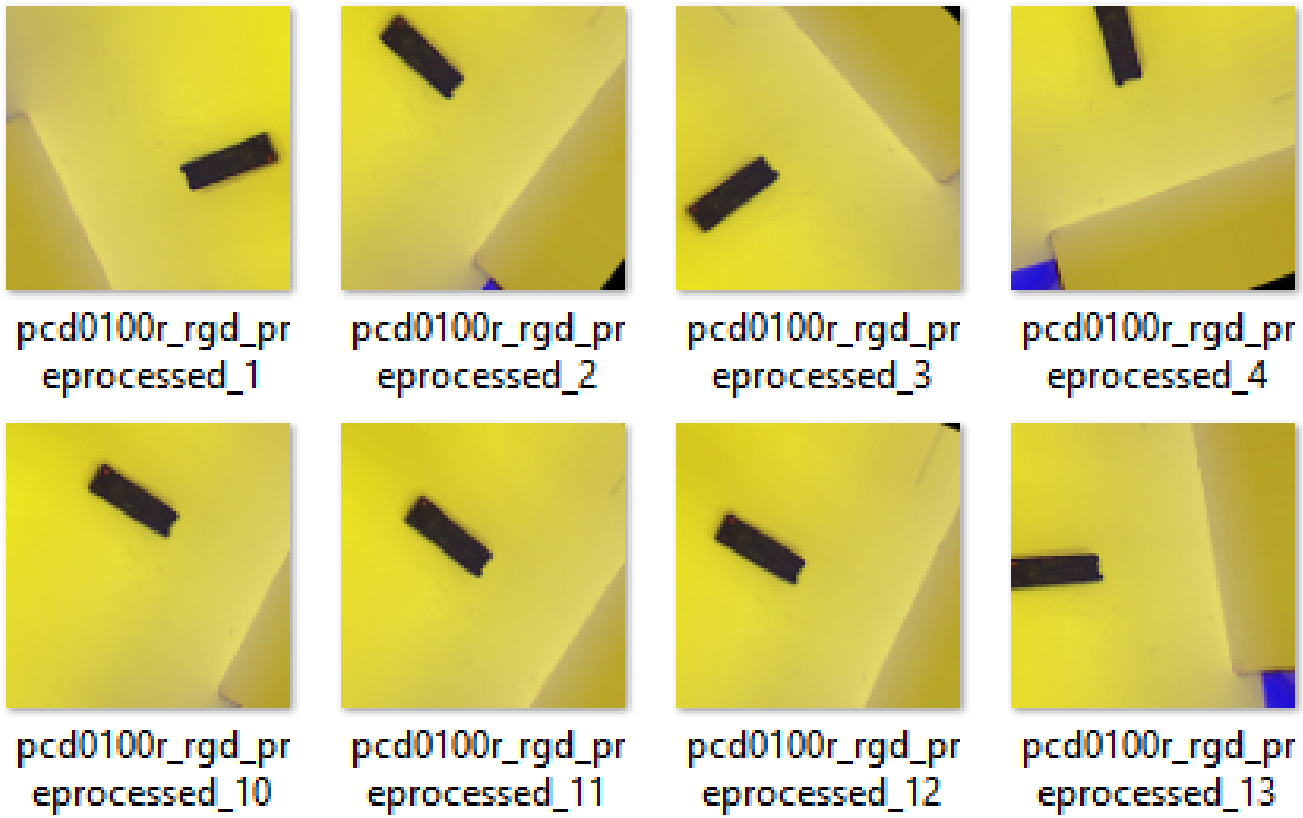


Figure 8: Images for `pcd0100r_rgd_preprocessed_1`

Reproducing GKNet Benchmarks: Cornell Dataset (cont.)

Annotation for `pcd0100r_rgd_preprocessed_1`

```
2 177.842467 110.189953 217.684406 128.688189 -1.193646
2 161.668984 111.392738 199.603883 137.878183 -1.250606
2 150.723138 118.692707 187.143203 137.258827 -1.232340
2 131.461796 126.084075 175.492871 140.997830 -1.164588
```

Reproducing GKNet Benchmarks: Jacquard Dataset

More than 50,000 RGB-D images of 11,000 objects with 1,000,000 positive grasp annotations. Simulated.

Figure 9: Example image from the Jacquard dataset

Reproducing GKNet Benchmarks: Jacquard Dataset (cont.)

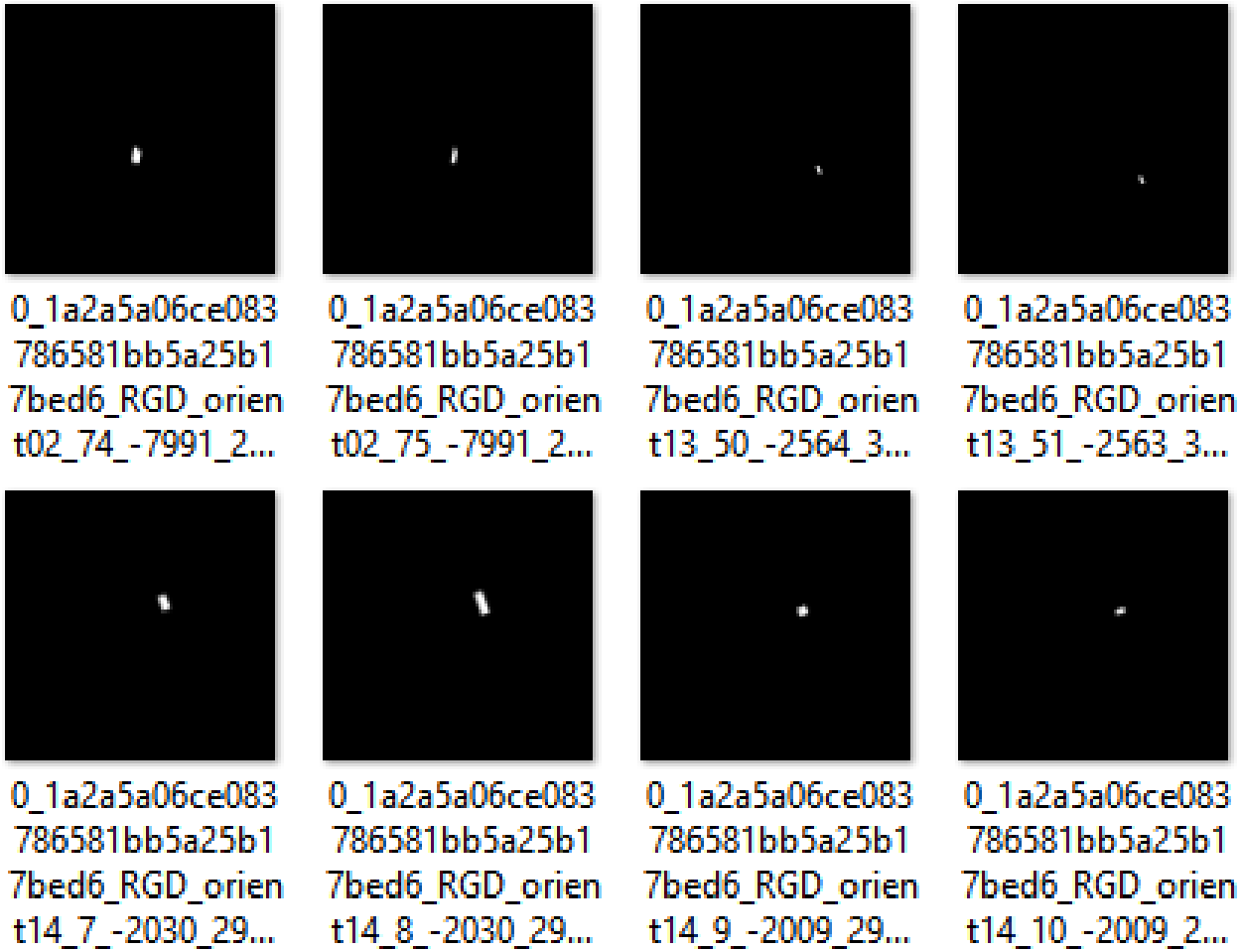


Figure 10: Example annotations from the Jacquard dataset

Reproducing GKNet Benchmarks: Docker Container

See ivalab/GraspKpNet PR #3

Changelog

- Dockerfile with Ubuntu 20.04, CUDA 11.7, and PyTorch 1.13
- Install ROS Noetic core libraries
- Fix build process of Deformable Convolutional Networks (DCNv2)
- Refactor GKNet as an importable Python package
- Add Docker Compose configuration for development and testing
- Mirror models and datasets on public Backblaze B2 bucket

Reproducing GKNet Benchmarks: Docker Container (cont.)

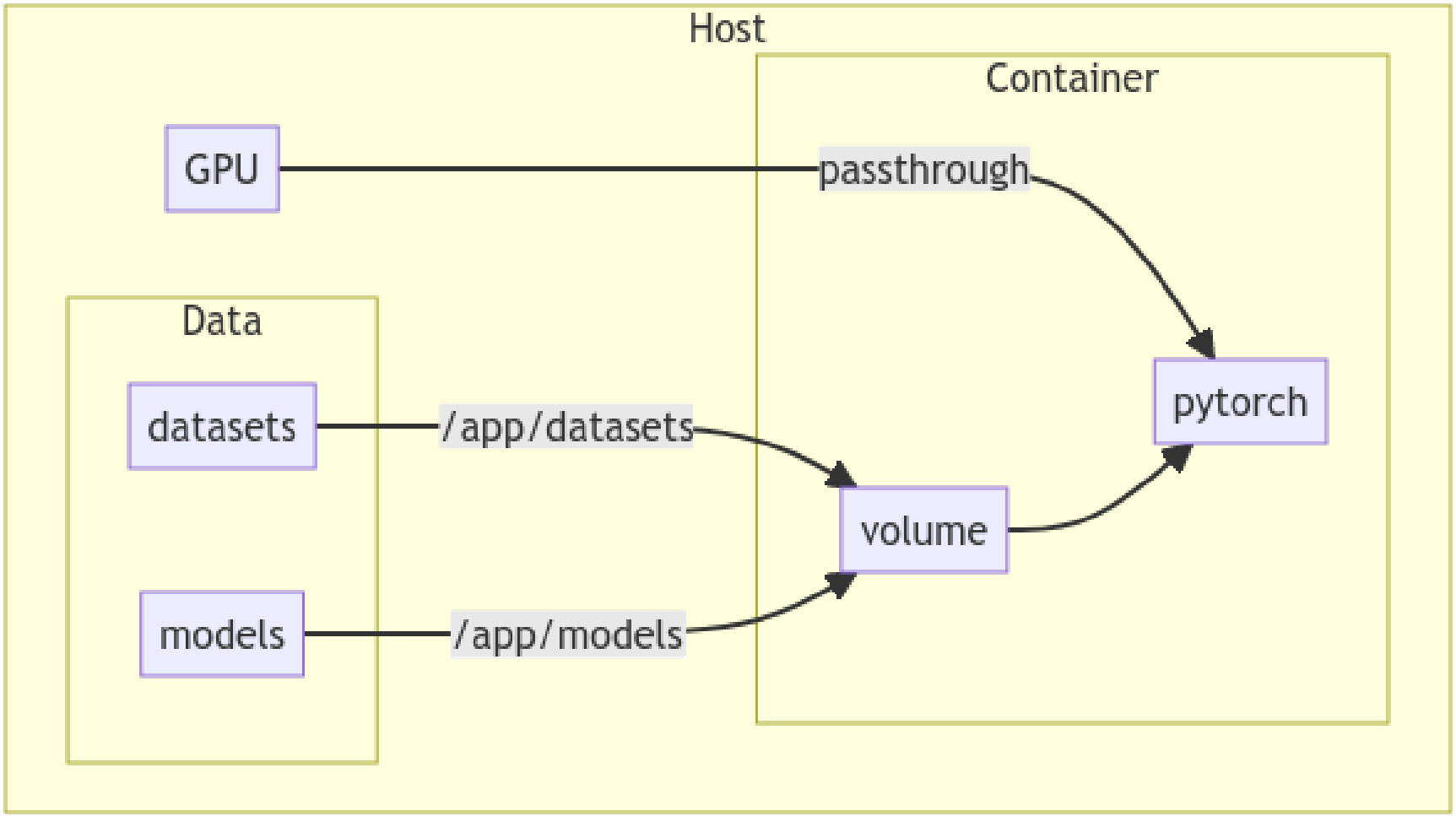


Figure 11: Docker container mounts and devices

Reproducing GKNet Benchmarks: Docker Container (cont.)

Windows 11 and a NVIDIA 1080 Ti (CUDA 11.7) on Docker Desktop via WSL2.

Docker Compose command

```
docker compose run --rm gpu \
  python scripts/test.py dbmctdet_cornell \
    --exp_id dla34_test \
    --arch dla_34 \
    --dataset cornell \
    --fix_res \
    --flag_test \
    --load_model models/model_dla34_cornell.pth \
    --ae_threshold 1.0 \
    --ori_threshold 0.24 \
    --center_threshold 0.05
```

Reproducing GKNet Benchmarks: Results

exp_id	dataset	accuracy	fps
model_alexnet_ajd	jac_coco_36	0.973701	82.4871
model_dla34_ajd	jac_coco_36	0.983857	75.2168
model_resnet18_ajd	jac_coco_36	0.979474	75.3912
model_resnet50_ajd	jac_coco_36	0.98236	76.1191
model_vgg16_ajd	jac_coco_36	0.983643	81.8511
model_alexnet_cornell	cornell	0.94663	281.826
model_dla34_cornell	cornell	0.967843	156.066
model_resnet18_cornell	cornell	0.957123	284.024
model_resnet50_cornell	cornell	0.961556	280.632
model_vgg16_cornell	cornell	0.964224	280.728

Results have similar accuracy; FPS achieved on NVIDIA 1080 Ti is higher than reported in the paper.

Reproducing GKNet Benchmarks: Further Ideas

- Pose estimation for other grasping orientations e.g. items from a shelf
- Training and evaluating GKNet on a primitive shapes dataset
  - Lin, Yunzhi, Chao Tang, Fu-Jen Chu, and Patricio A. Vela. "Using synthetic data and deep networks to recognize primitive shapes for object grasping." In 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 10494-10501. IEEE, 2020.
- Neural Radiance Fields (NeRFs) for training augmentation or direct pose estimation
  - Dellaert, Frank, and Lin Yen-Chen. "Neural volume rendering: Nerf and beyond." arXiv preprint arXiv:2101.05204 (2020).
  - Yen-Chen, Lin, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. "inerf: Inverting neural radiance fields for pose estimation." In 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1323-1330. IEEE, 2021.
  - [GitHub] awesome-NeRF/awesome-NeRF

Future Work

Simulating pick and place with Gazebo, Handy, and GKNet

- Goal is for simulation closely match lab procedures with physical operation of robots.
- Provide a platform for higher level scene understanding and natural language work. Tie in with Factory Automation and Assistive Robotics.
- ETA 2-4 weeks for a working prototype.

TSRB Gazebo world

- Configure ivaROS/customWorlds as a ROS package with Gazebo model exports.
- Generate a subset of the world local to TSRB 444.
- Find a volunteer to help flesh out the world in context of Factory Automation task.

Thank you!

Figure 12: short.acmiyaguchi.me/autobots-spr23-1

Autobots VIP Spring 2023

VisMan Progress Presentation 2

Anthony Miyaguchi
acmiyaguchi@gatech.edu

2023-04-04

Outline

Pick and Place Simulation

- Lights, Handy, Action
- Camera Calibration Package
- GKNet in Action

Image Detection and Segmentation

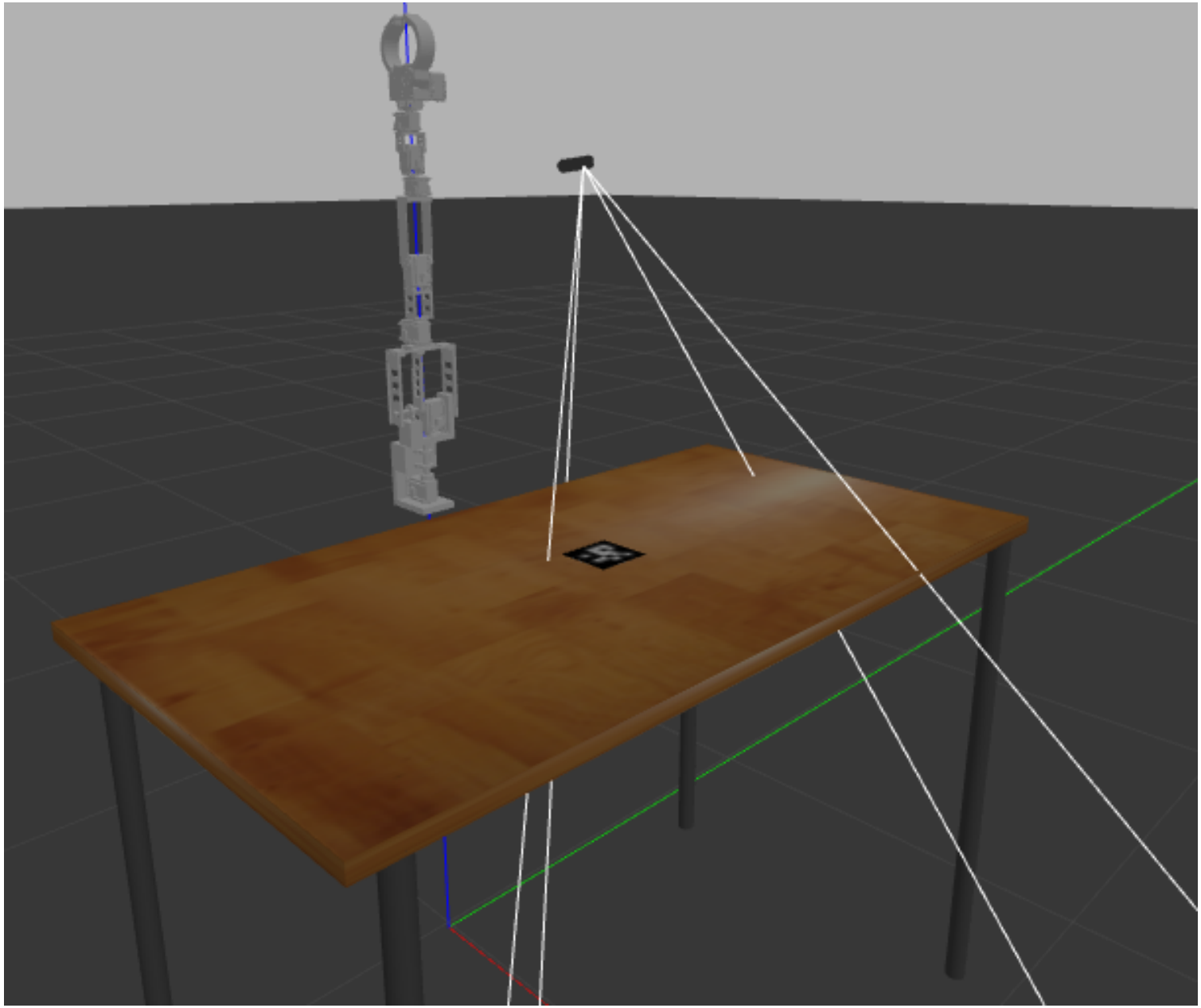
- Presented by Nick after (or before) this presentation

Lights, Handy, Action

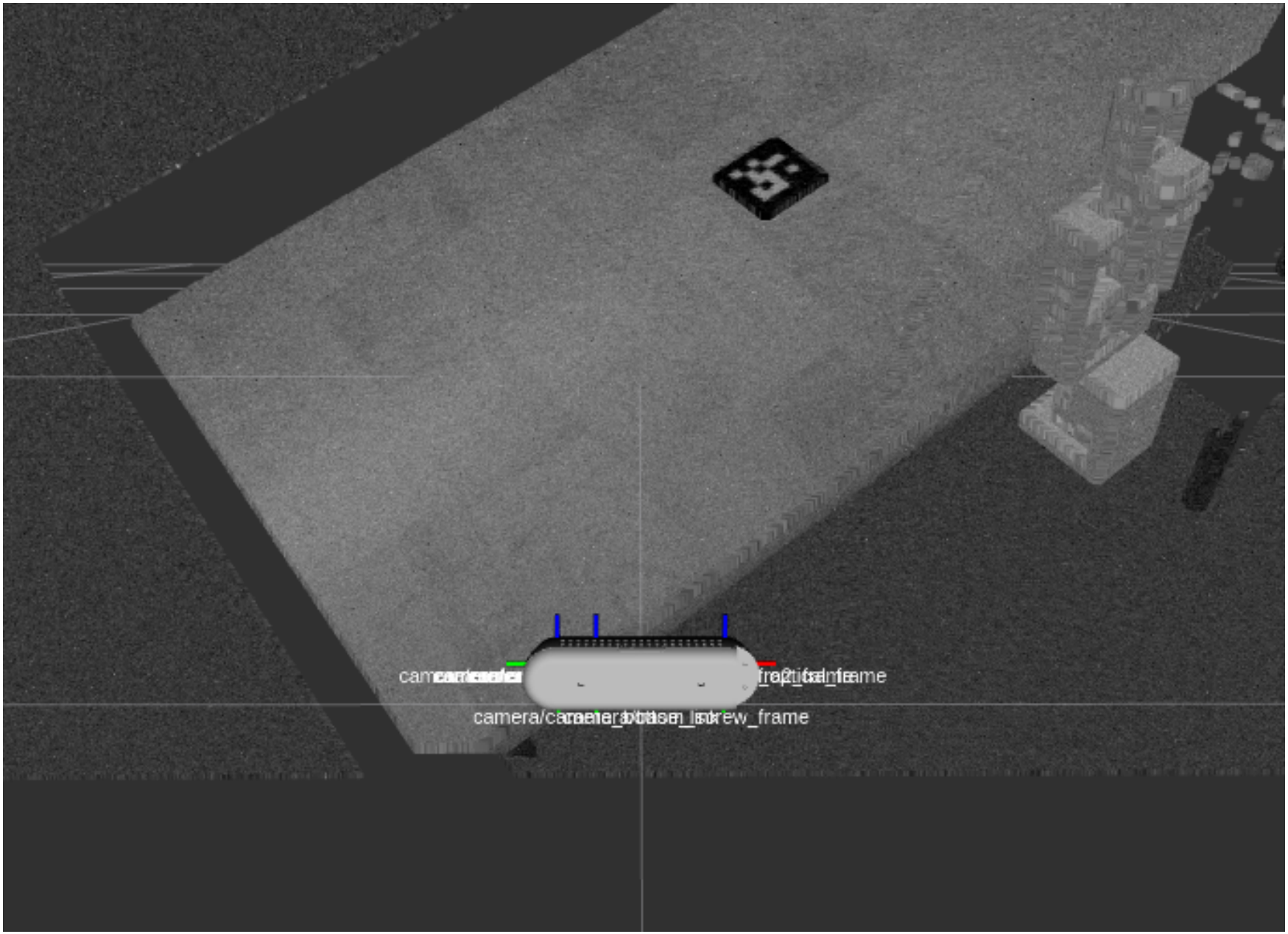
Progress

- Gazebo world with a Realsense D435, Handy, and a table.
- Gazebo is configured to work with MoveIt configured for Handy.

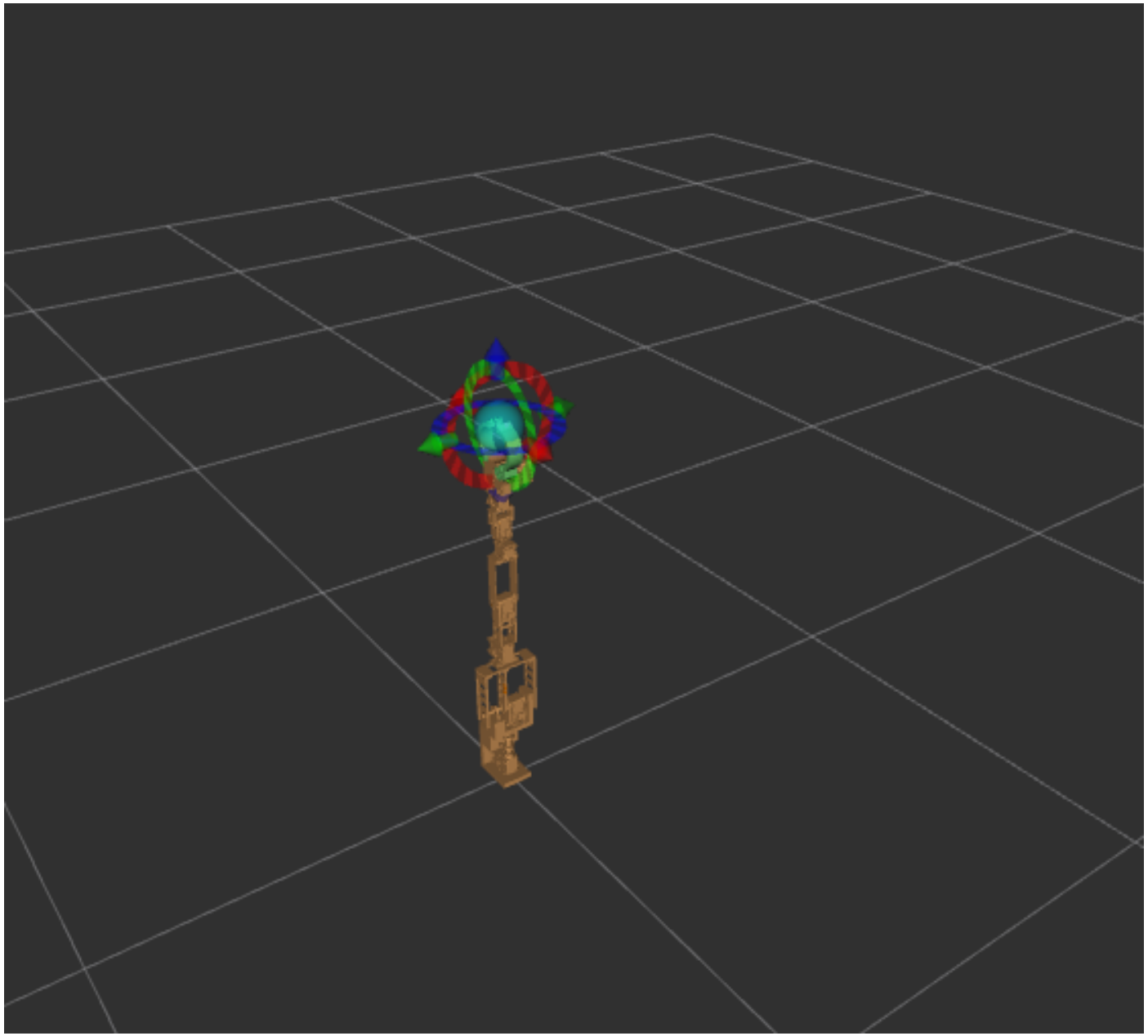
Gazebo World



Realsense D435 Point Cloud in RViz



Handy MoveIt in RViz



Demo Video

YouTube

Realsense, Handy, and Gazebo Demo Video

Camera Calibration Package

Progress

Mostly functional, reuses ivapylib/camera to determine camera extrinsics via OpenCV and a single ArUco tag. Needs more tests to ensure robustness.

Procedure

Spawn an ArUco tag with fixed world location. Then:

1. Republish the camera info topic
2. Wait until an ArUco tag is detected
3. Perform calibration; wait until transformation matrix is stable
4. Publish the transformation matrix
5. Publish calibration status

Calibration Extrinsic Publisher

Latched topic that publishes the transformation matrix from the camera to the world.

Echoing the Topic

```
$ rostopic echo /calibration/camera/camera_extrinsics
header:
  seq: 1
  stamp:
    secs: 0
    nsecs: 0
  frame_id: ''
M_CL: [
  0.9999001035866886, 0.011796350567605439, 0.007786460082420
  0.013191277641741089, -0.976675609903027 8, -0.21431458936509
  0.00507671562208 1074, 0.2143958934630858, -0.976733652448126
  0.0, 0.0, 0.0, 1.0
]
```

GKNet in Action

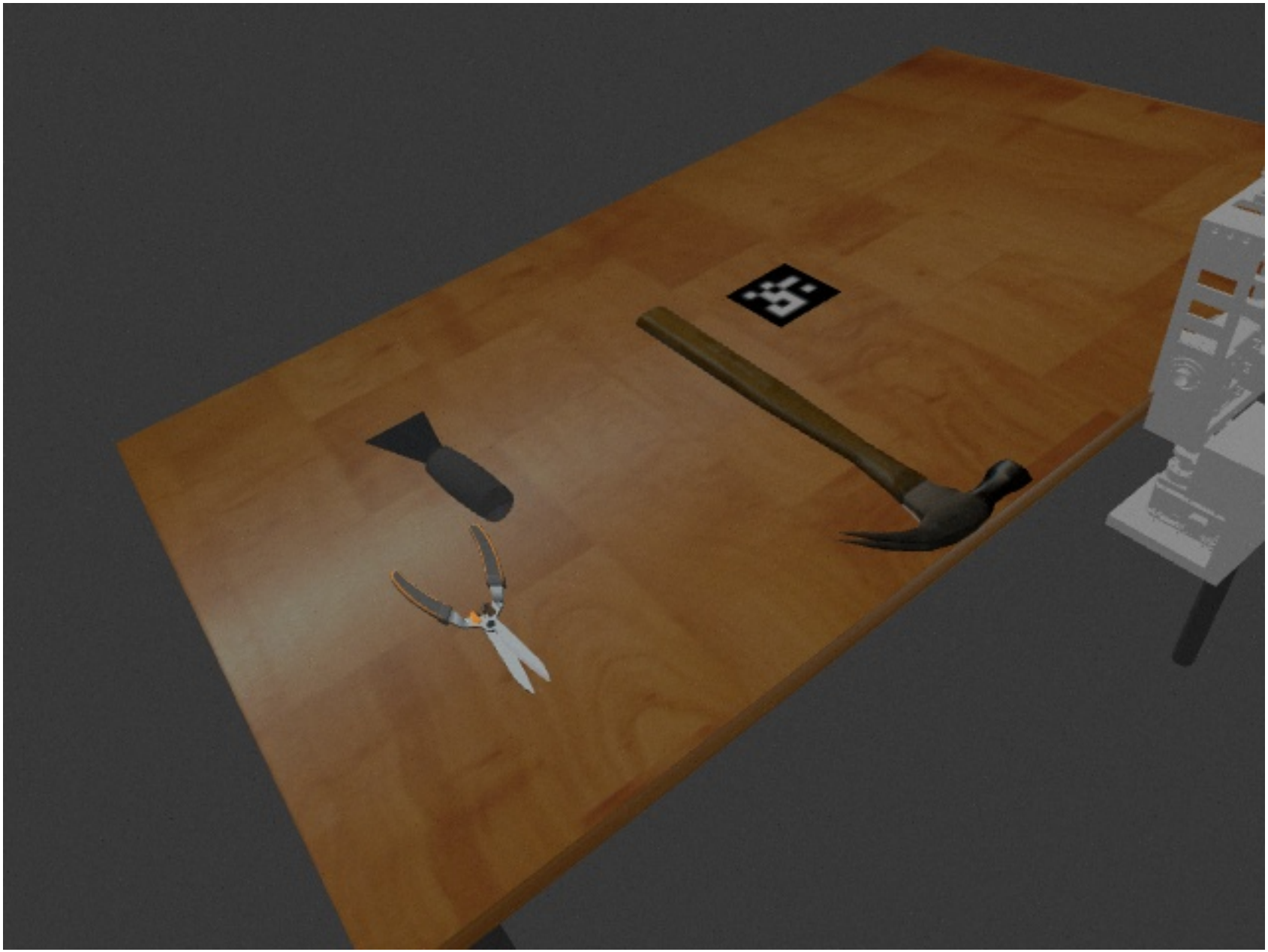
Paper Reference

Xu, Ruinian, Fu-Jen Chu, and Patricio A. Vela. "GKNet: grasp keypoint network for grasp candidates detection." The International Journal of Robotics Research 41, no. 4 (2022): 361-389.

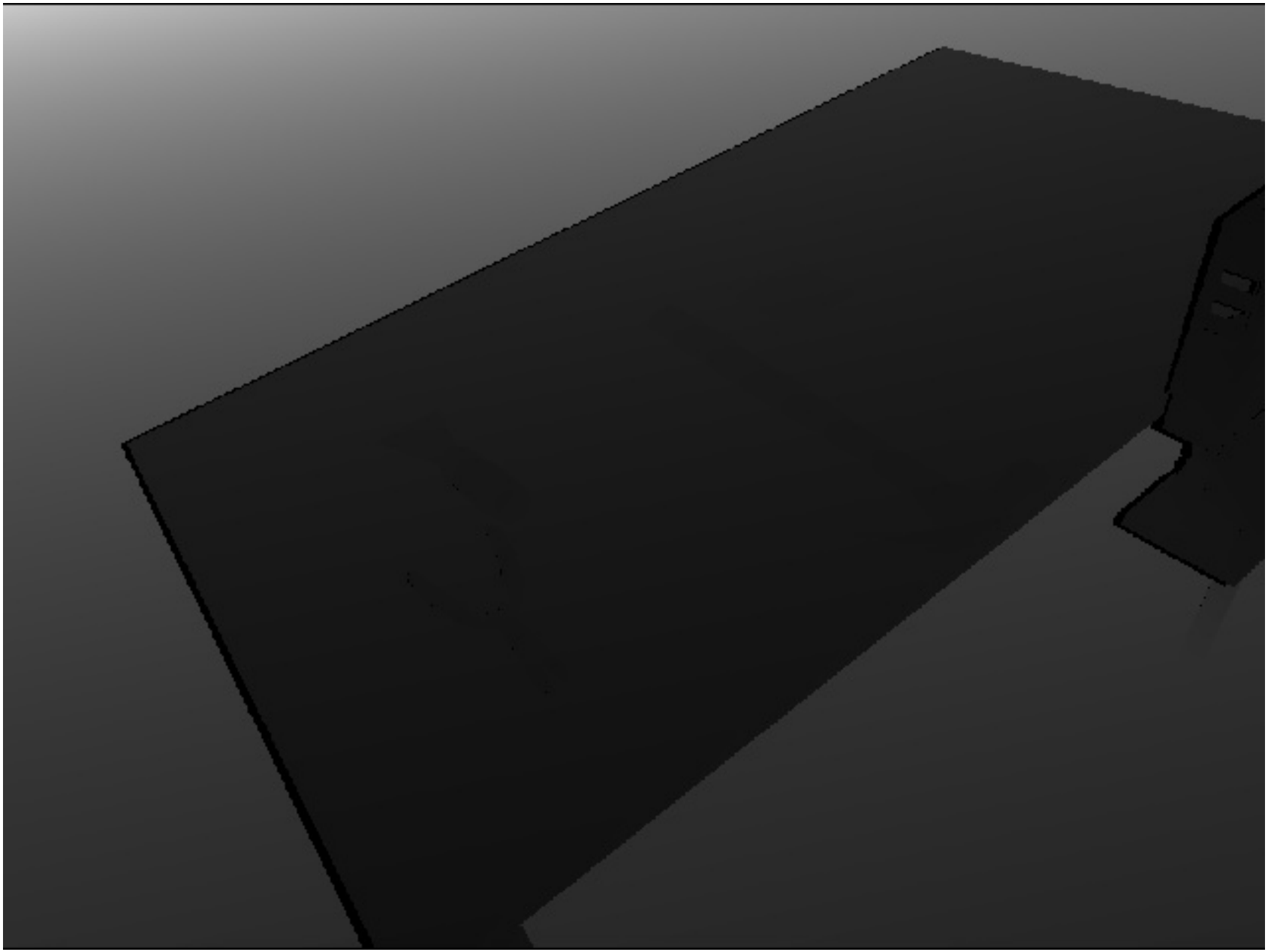
Goal

Take an image from the pick and place simulation and run it through the GKNet model.

Color Image



Depth Image



GKNet Architecture

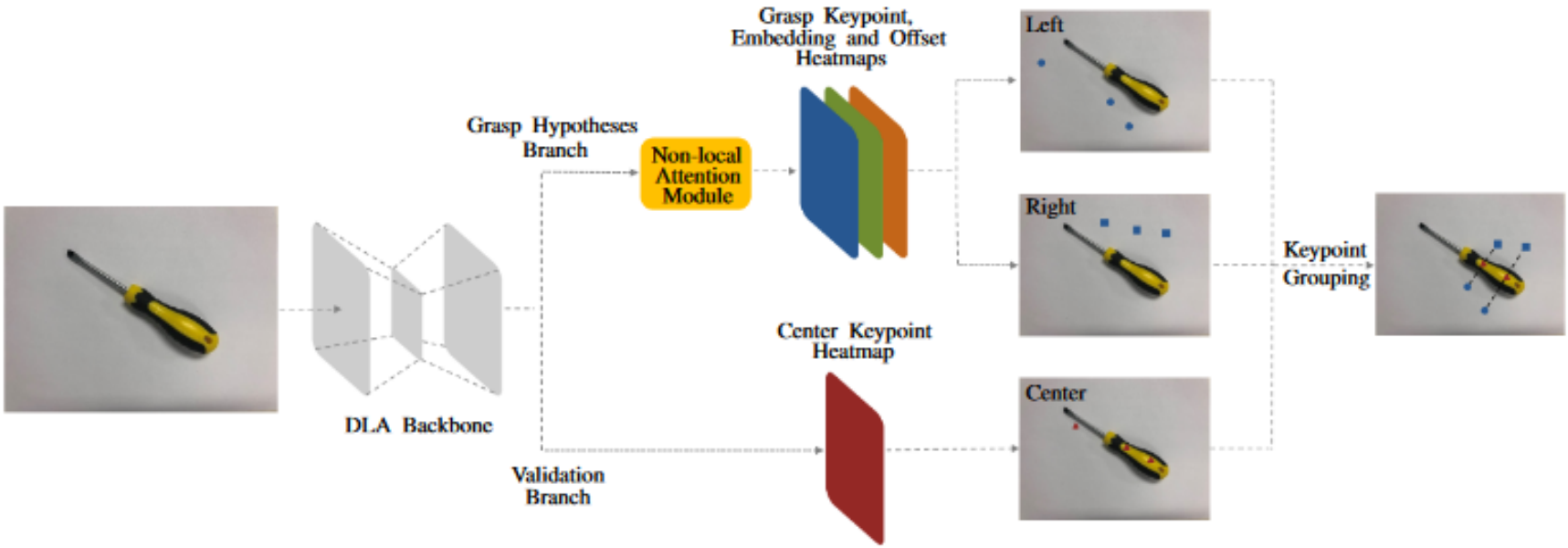
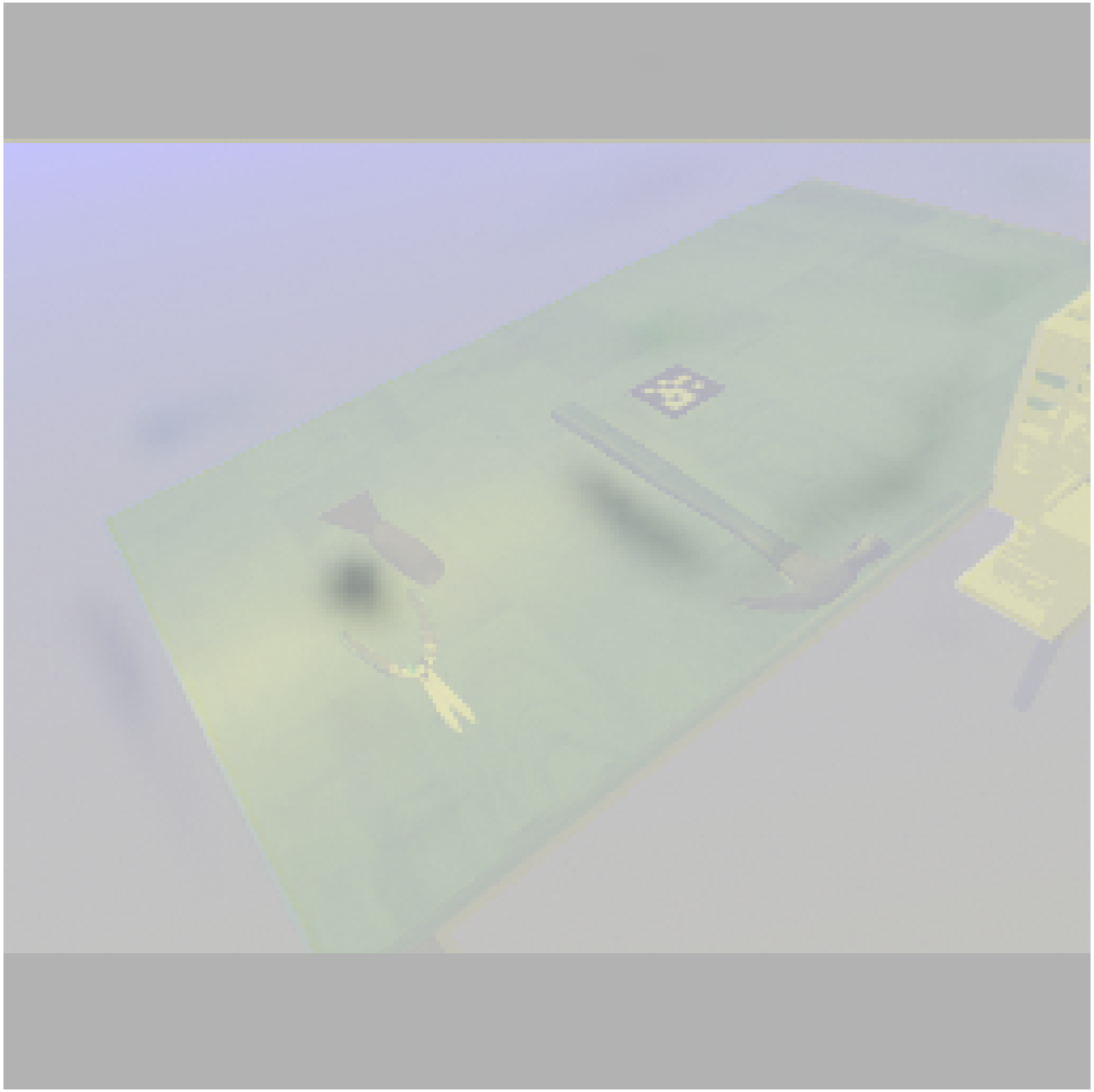


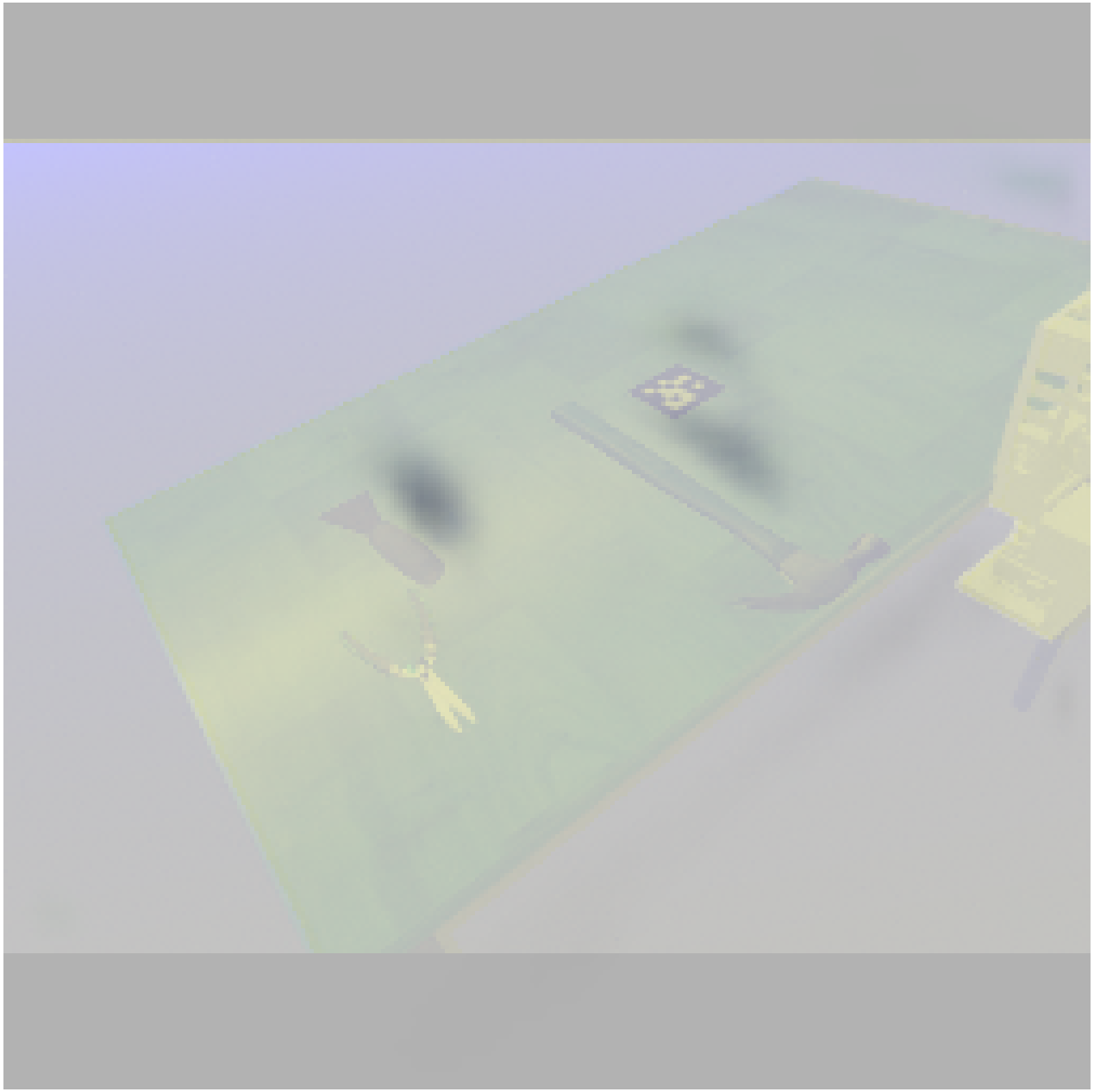
Figure 2. Architecture of GKNet. An hourglass-like backbone network is followed by two prediction branches, one for grasp keypoints and one for a center keypoint. A non-local attention module is inserted between backbone network and the prediction branch for grasp keypoints. The pair of *grasp keypoint heatmaps* with corresponding embeddings and offsets and the *center keypoint heatmap* are fed into keypoint grouping algorithm and yield the final grasp detections. Blue dots correspond to left grasp keypoints, blue rectangles correspond to right grasp keypoints and red triangles correspond to center keypoints.

Figure 6: GKNet architecture (see paper)

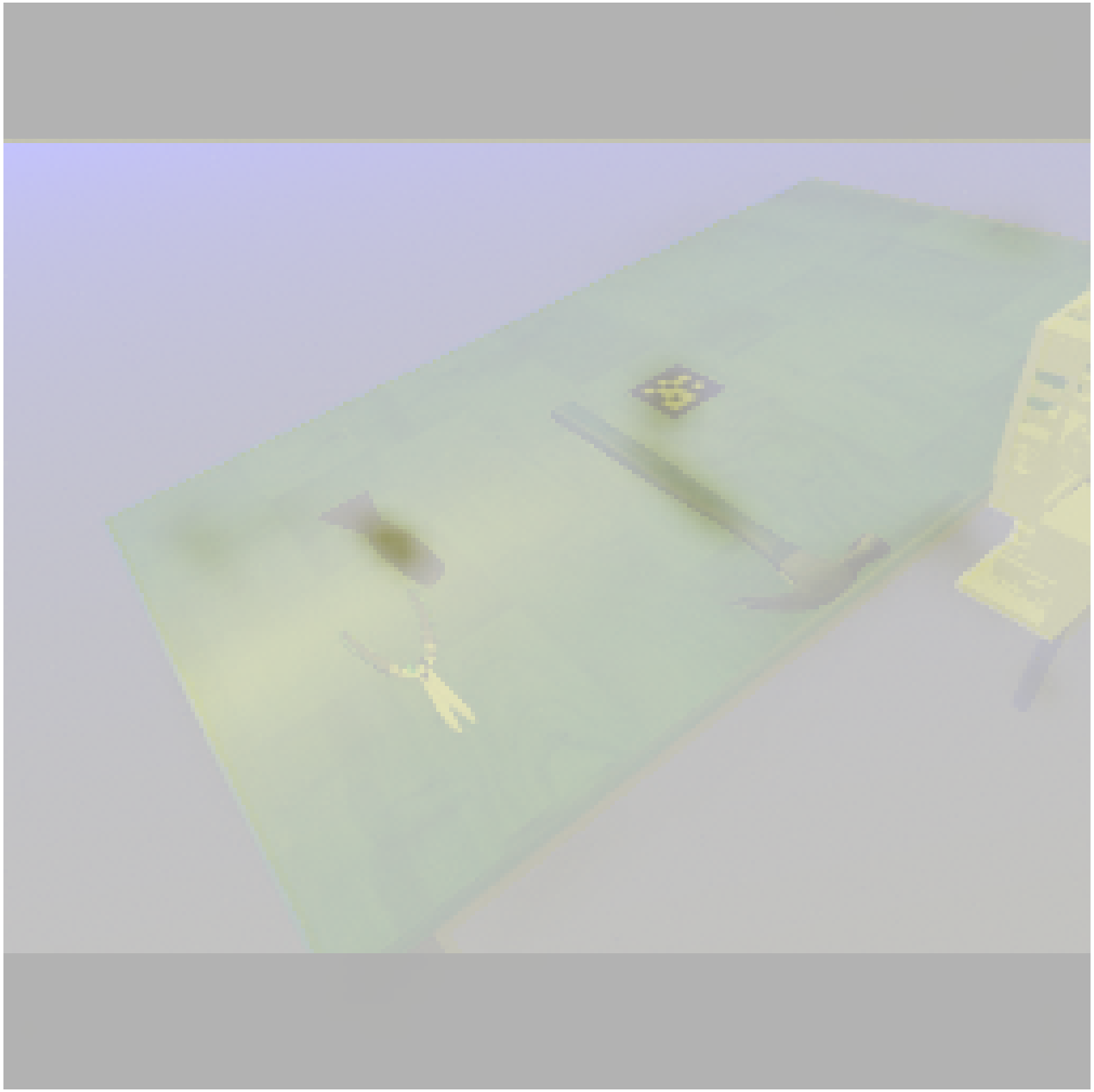
Heatmap - Left Means



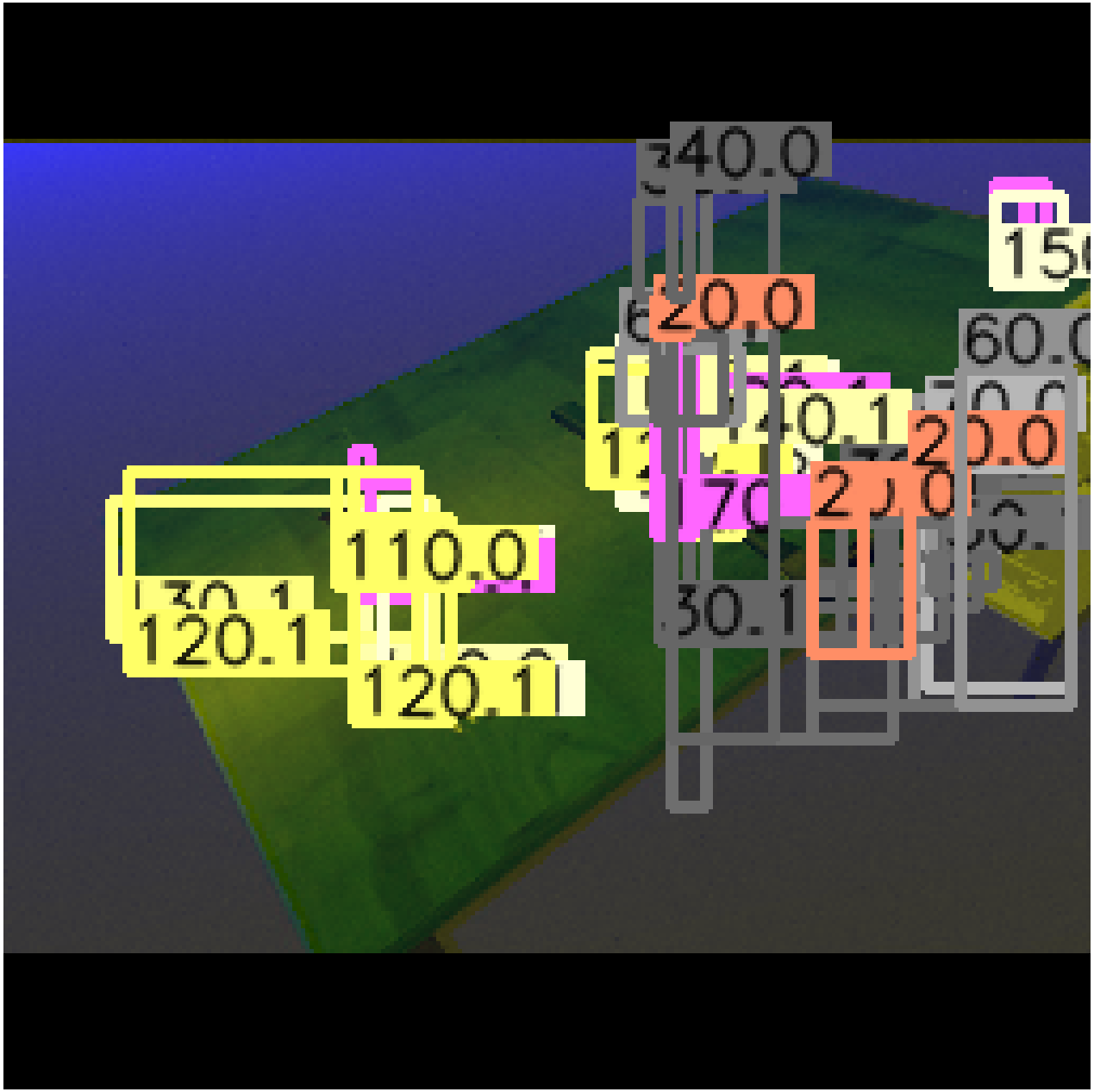
Heatmap - Right Means



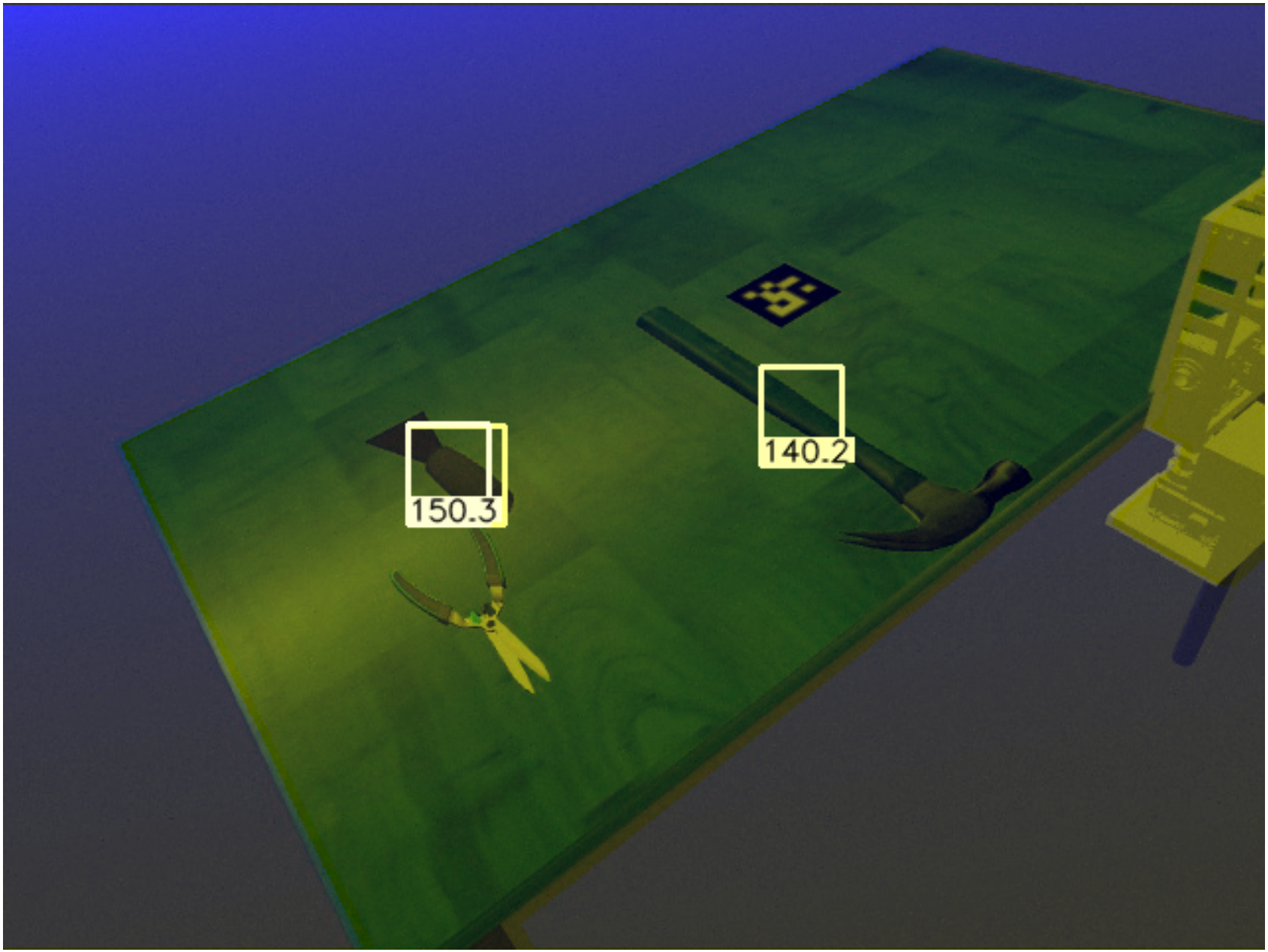
Heatmap - Center Means



Ranked Predictions



Ranked Predictions with Center Thresholding



What next?

Instead of a one-off script, the GKNet model will be integrated into a ROS node that can be used in the pick and place simulation.

Inputs

- `/camera/color/image_raw`
- `/camera/aligned_depth_to_color/image_raw`
- `/camera/color/camera_info`
- `/camera/aligned_depth_to_color/camera_info`
- `/calibration/camera/camera_extrinsics`

Outputs (TBD)

- `/gknet/predictions`
  - ranked list of keypoint pairs in camera coordinates
- `/gknet/image_annotated`
  - image with keypoints and bounding boxes drawn on it

Thank you!

Questions?

Autobots VIP Spring 2023

VisMan Progress Presentation 3

Anthony Miyaguchi
acmiyaguchi@gatech.edu

2023-04-18

Outline

Pick and Place Simulation

- GKNet ROS Module
- Current/Future Work

Handy and D435 in the Lab

Presented by Ashlynn and Calvin.

GKNet ROS Module

GKNet model will be integrated into a ROS node that can be used in the pick and place simulation. Visual demo runs from a Docker container.

Inputs

- `/camera/color/image_raw`
- `/camera/aligned_depth_to_color/image_raw`
- `/gknet/object_filter`
  - A list of bounding boxes to perform per-object grasp ranking

Outputs

- `/gknet/keypoints`
  - ranked list of keypoint pairs in camera coordinates
- `/gknet/annodated_image`
  - image with keypoints and bounding boxes drawn on it

GKNet ROS Module Demo

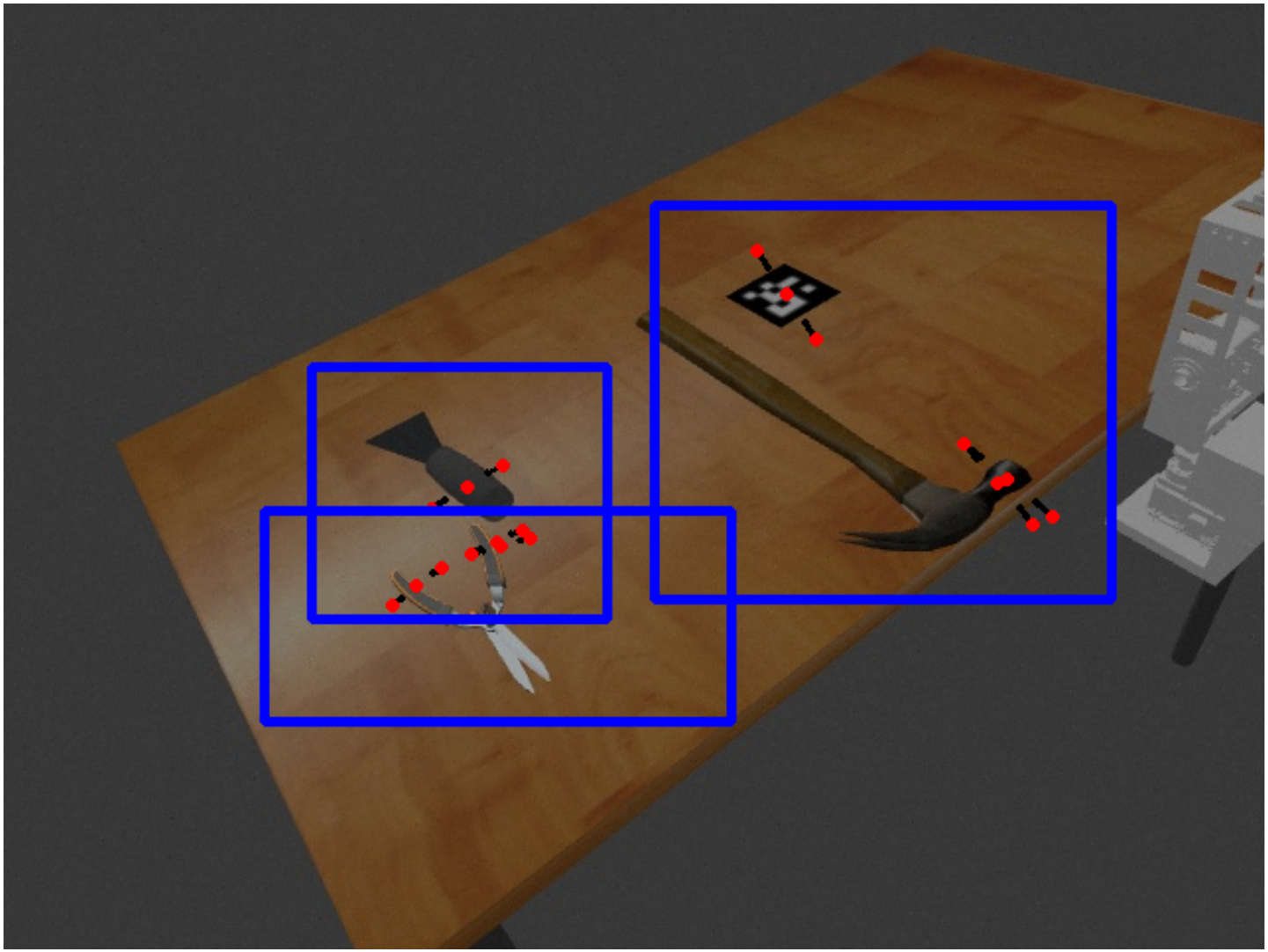


Figure 1: GKNet ranking grasping keypoints per bounding box

Integration: Architecture Overview

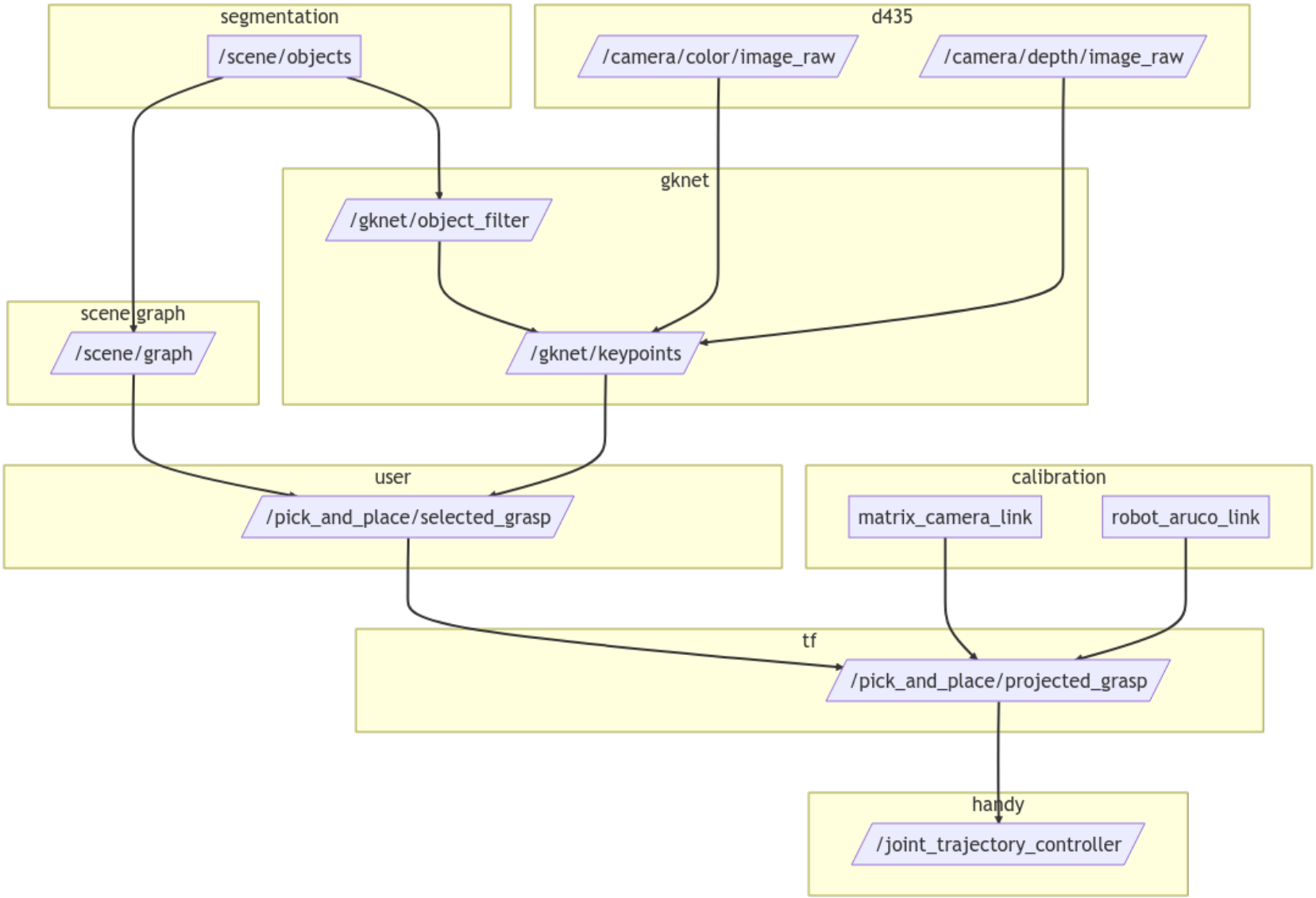


Figure 2: Architecture overview of pick and place topics.

Current/Future Work

Integration across all the components is still required, but the perception side is mostly complete.

TODO for pick-and-place demo

- Bug with ApproximateTimeSynchronizer in GKNet module
- SimData models lack mass/physics properties needed for grasping
- TF transform nodes for grasping poses from camera to world frame
- Implementation of segmentation/YOLO bounding boxes into GKNet
- Implementation of MoveIt planning and execution on selected grasps
- Verification of YOLO/GKNet on a real D435 camera
- Verification of calibration code on a real D435 camera