ROS-I Training Class

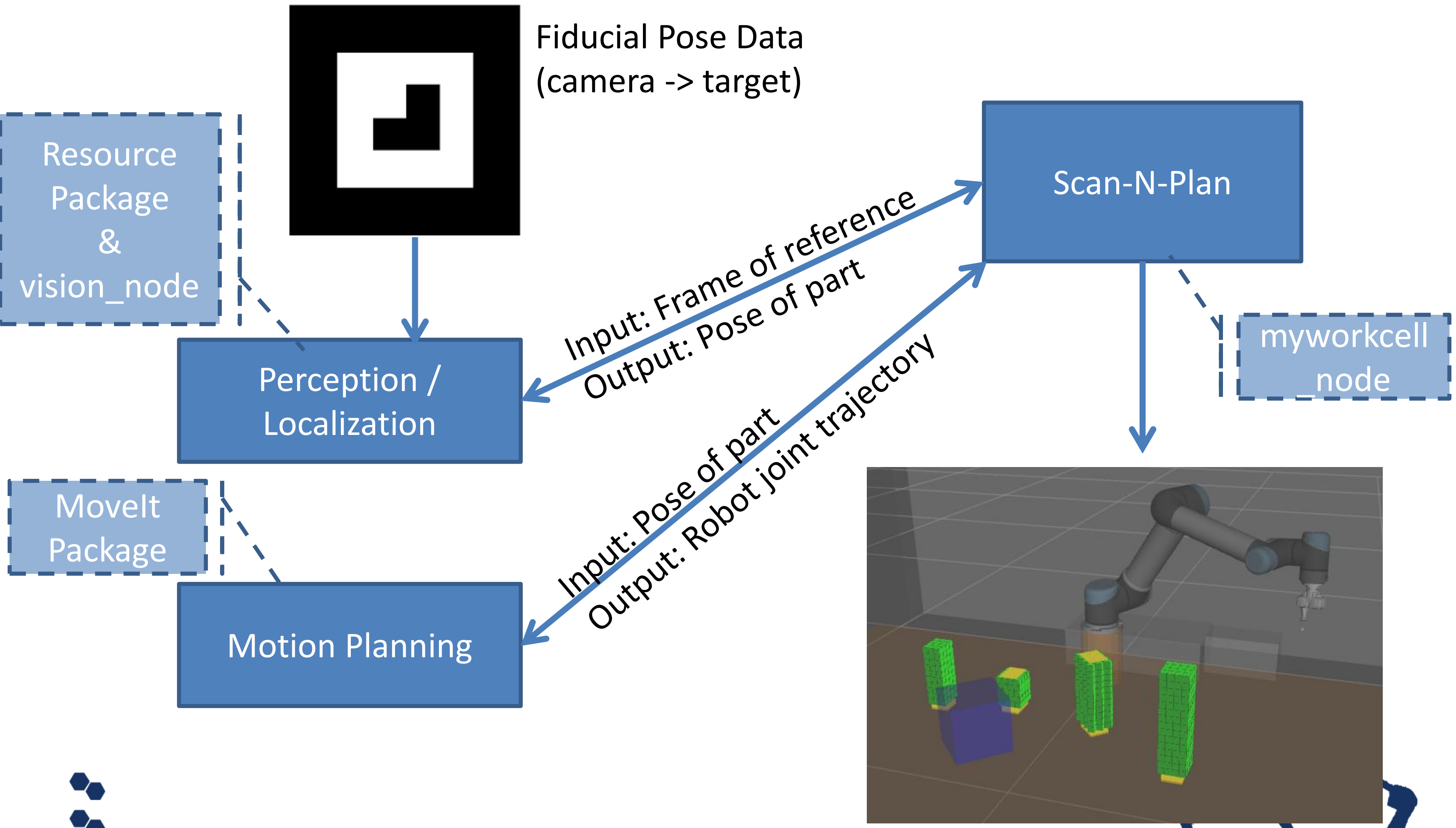
Basic Developers' Training Class

Day 3

October 2023

Presented by Southwest Research Institute

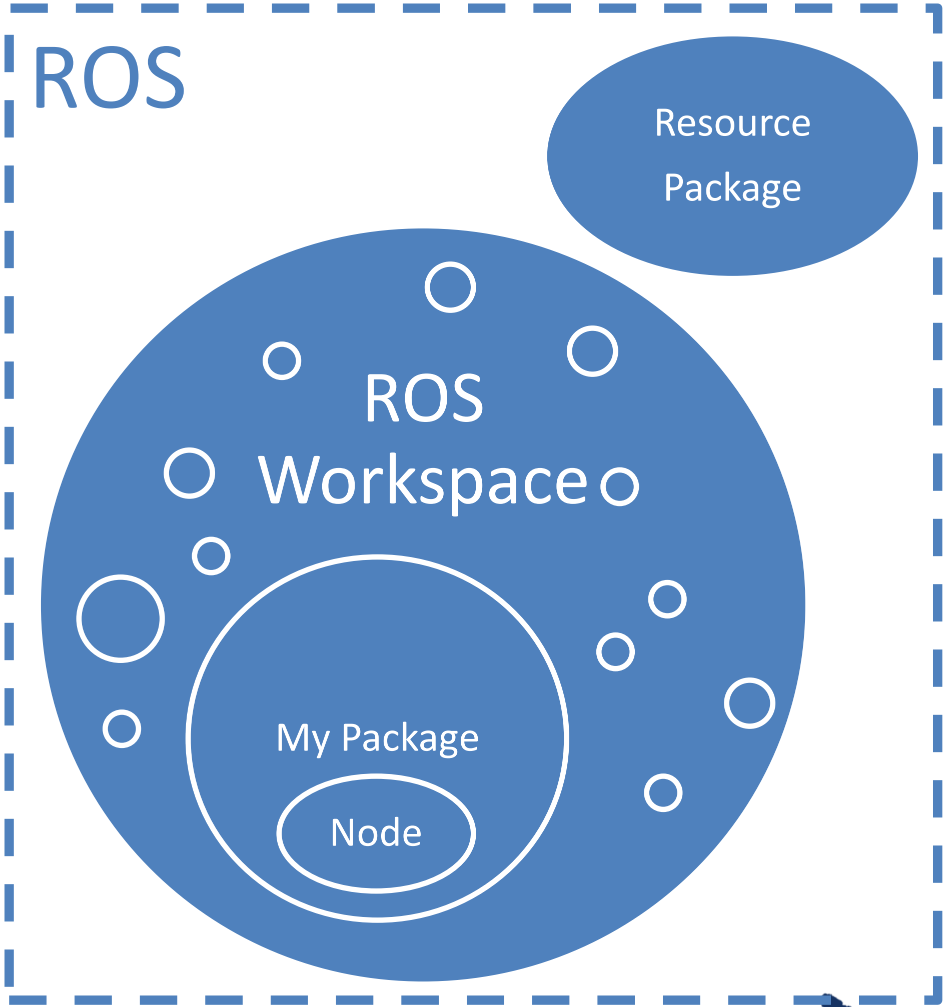
Day 3: Scan-N-Plan App



RECAP!

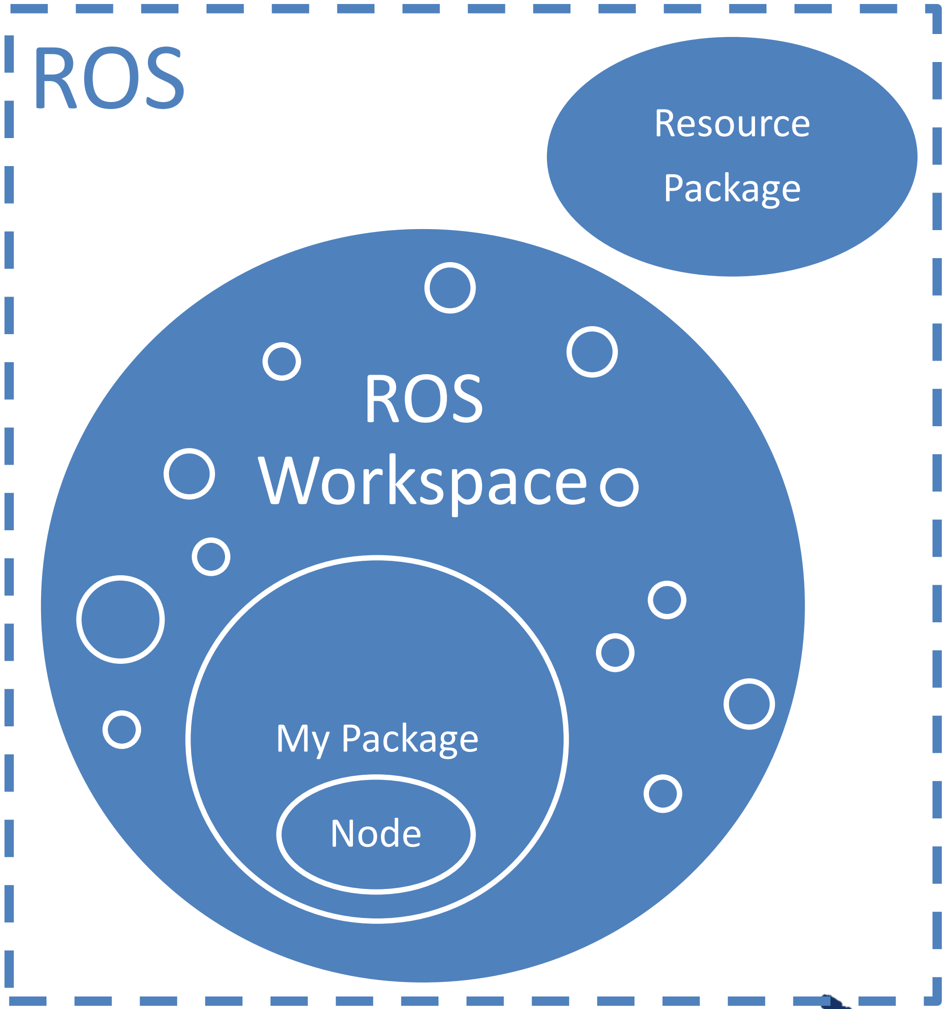
Day 1 Progression

- ❑ Install ROS
- ❑ Create Workspace
- ❑ Add “resources”
- ❑ Create Package
- ❑ Create Node
  - ❑ Basic ROS Node
  - ❑ Interact with other nodes
    - ❑ Messages
    - ❑ Services
- ❑ Run Node
  - ❑ ros2 run
  - ❑ ros2 launch



Day 2 Progression

- ❑ Build on Day 1 nodes
- ❑ Create Robot Model
  - ❑ Basic, URDF
  - ❑ Advanced, xacro
- ❑ Use TF
- ❑ Integrate Robot Model
  - ❑ MoveIt



Let's look at the reference material

RECAP!

ROS Workspace

- Your "system" – a high level collection of your project

- colcon build
  - Must be at workspace level

Install Existing Packages

- Where do existing packages exist?
  - Apt-get debians
  - Clone usually from github
- Remember what you did…
  - We cloned fake_ar_publisher into our workspace

Creating a Package(s)

- What is a package?
  - Container for executables, messages, other ROS items
    - Sometimes only configuration files
    - Sometimes only msg files

- Review CMakeLists.txt & package.xml

- Note special subfolders within packages:
  - msg, srv, action,

Nodes

- What is it?
  - – ROS structure for a process
  - – Runs independently of other processes
  - – Fundamental unit of re-usable code

- Let’s see which ones are running…
  - – What information can we get about them?
    - ros2 node list, rqt_graph

Messages

- What is it?
  - Over the wire data structure
  - This specific data travels via a “topic”

- Let’s see which ones we have…
  - ros2 interface “package”
  - ros2 interface show “package”
- What else is available: common_msgs
  - sensor_msgs, geometry_msgs, control_msgs, trajectory_msgs, odometry_msgs, std_msgs,

Topics

- What is it?
  - Channels that route messages
  - Typical for streams of messages/data
  - One way communication (publisher->subscriber)
- Let’s see which ones are running…
    - ros2 topic list
    - ros2 topic info “topic”
    - ros2 topic echo “topic”
    - rqt_graph

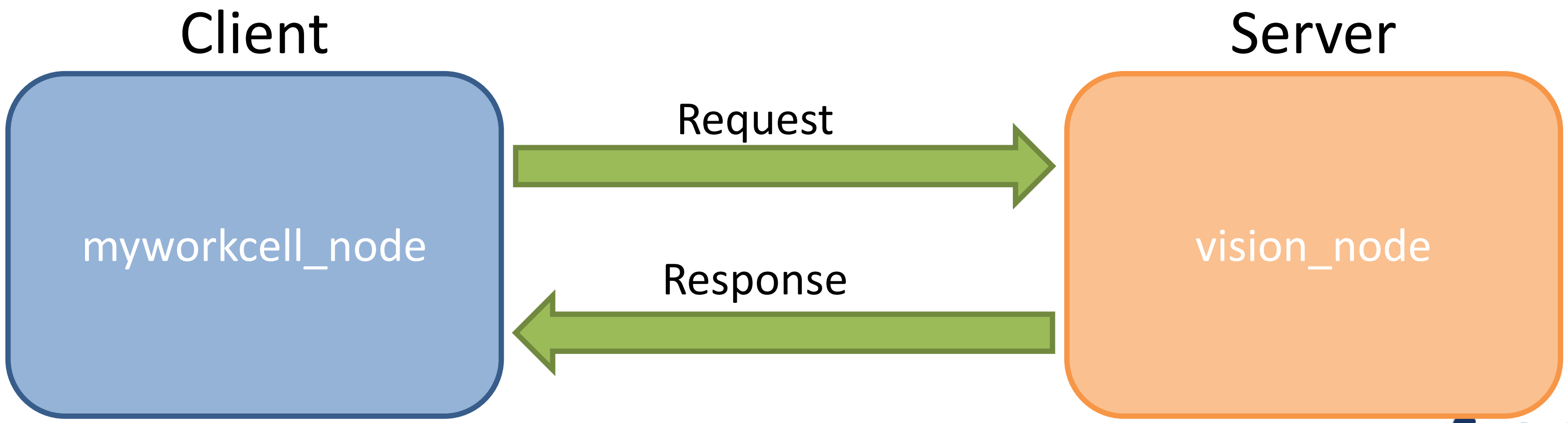
Topics

- Let’s see what we wrote…
  - Callbacks – event driven

```
ar_sub_ = this->create_subscription<fake_ar_publisher::msg::ARMarker>(
                 "ar_pose_marker",
                 rclcpp::QoS(1),
                 std::bind(&Localizer::visionCallback, this,
                 std::placeholders::_1));

 void visionCallback(fake_ar_publisher::msg::ARMarker::SharedPtr msg)
{
last_msg_ = msg;
}
```

Services

- What is it?
  - Two way communication (client request, server response)
  - "Blocking" calls, usually completed quickly



Services

- What is the .srv file?
  - Text file with request/response definition
    - Can contain any ROS msg type
  - Needs to be generated by ament
    - Requires changes in CMakeLists.txt & package.xml
    - Same/similar changes required to generate msgs, actions

- Let's see what ours look like…
  - Open .srv, CMakeLists.txt, and package.xml

Services

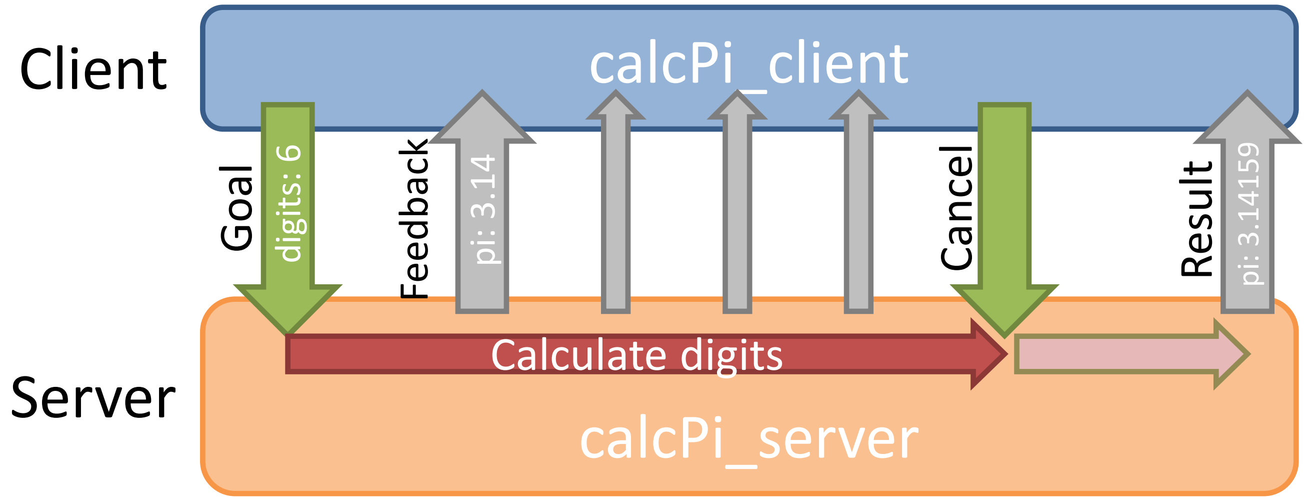
- Let’s see what we wrote…
  - vision_node acts as server

```
server_ = this->create_service<LocalizePart>("localize_part", …)

void localizePart(LocalizePart::Request, LocalizePart::Response)
{…}
```

  - workcell_node acts as a client

```
vision_client_ = this->create_client<LocalizePart>("localize_part");

auto request = std::make_shared<LocalizePart::Request>();

request->base_frame = base_frame;
auto future = vision_client_->async_send_request(request);

rclcpp::spin_until_future_complete(future)

auto response = future.get();
```

Actions



- Feel free to check out the action tutorials…

Launch Files

- Easier way to run set of nodes
  - Multiple nodes
  - Change name of nodes
  - Set parameters at launch
- Let’s review what we wrote…

Parameters

- Parameters have directory structure
  - Be careful about namespacing
- Meant to initialize without recompiling
  - See note in slides on runtime “dynamic reconfigure”
  - Can be set in files: yaml, config, launch etc.

- Let’s review what we wrote…
  - Launch file param
  - Cpp param

URDF/Xacro

- Descriptions of links and joints
  - Links: physical objects, usually with geometry
  - Joints: Relationships between links
- Feeds into motion planners (collision checking)

- Let’s review what we wrote: workcell.urdf.xacro
- Let’s review what is running…
  - Inspect rviz (collision objects, marker types, etc.)

TF

- TF keeps track of multiple coordinate frames over time
- Let’s review what is running…
  - Inspect rviz (tf tree)
  - ros2 run tf view_frames
  - ros2 run tf tf_echo [ref_frame] [target_frame]

TF

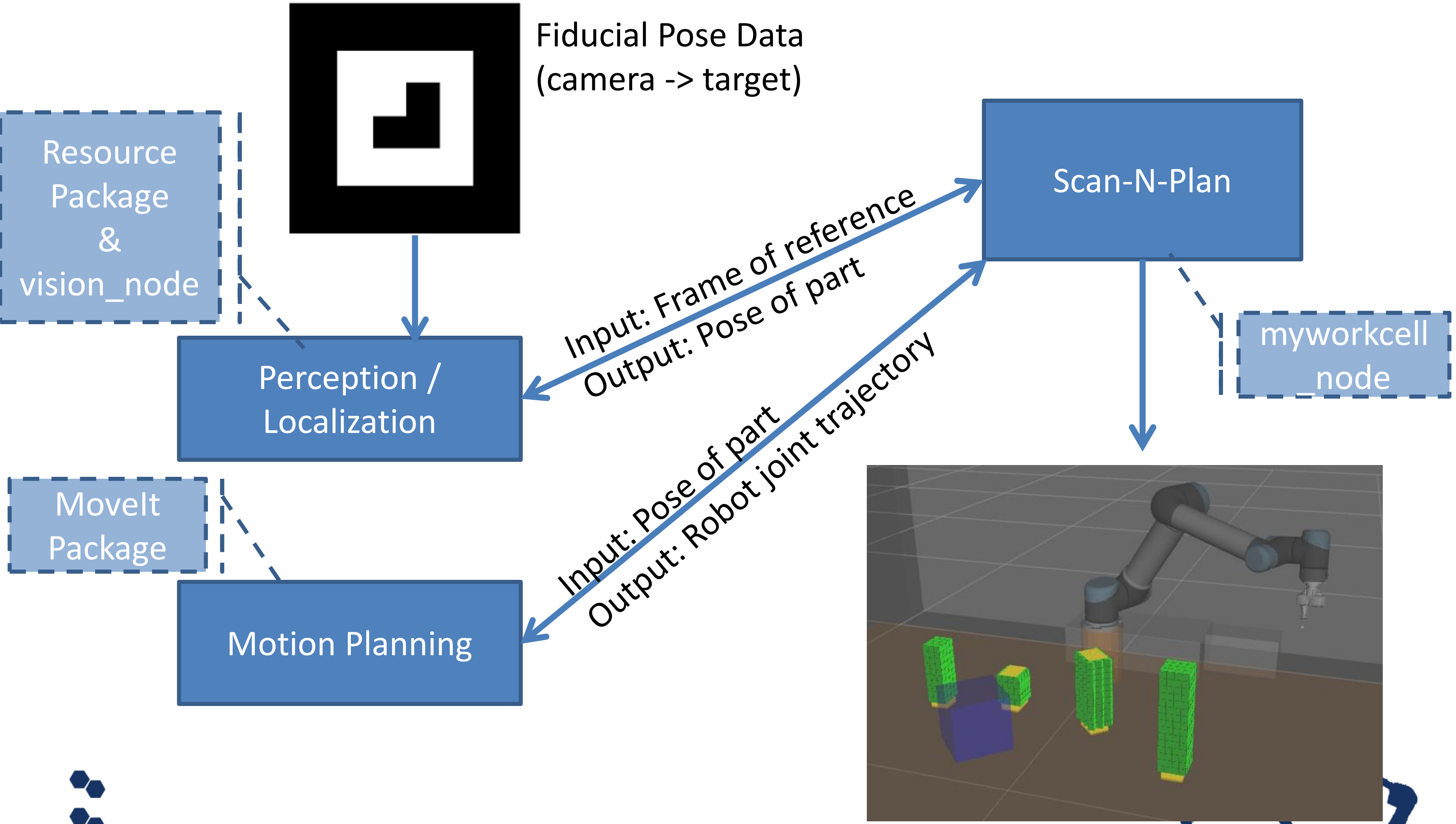
- Let’s review what we wrote…
  - Listener
  - Listener.lookupTransform
    (target_frame, source_frame, time, transform)
  - Convert from “msgs” types to tf transform
  - Perform transformation (result matrix = A * B)
  - Convert back to tf to msg to send to service client

MoveIt

- Use MoveIt Wizard to process a URDF
  - Kinematic chains organized into named groups
- Planning Interfaces
  - RVIZ
  - MoveItCpp class
  - Service Calls

FINAL DEMOS!

Day 3: Scan-N-Plan App



Day 3 Exercises

- Choose:
  - Demo1: Perception-Based Manipulation
  - Exercise 5.0: Building a perception pipeline
  - Ask the Trainers