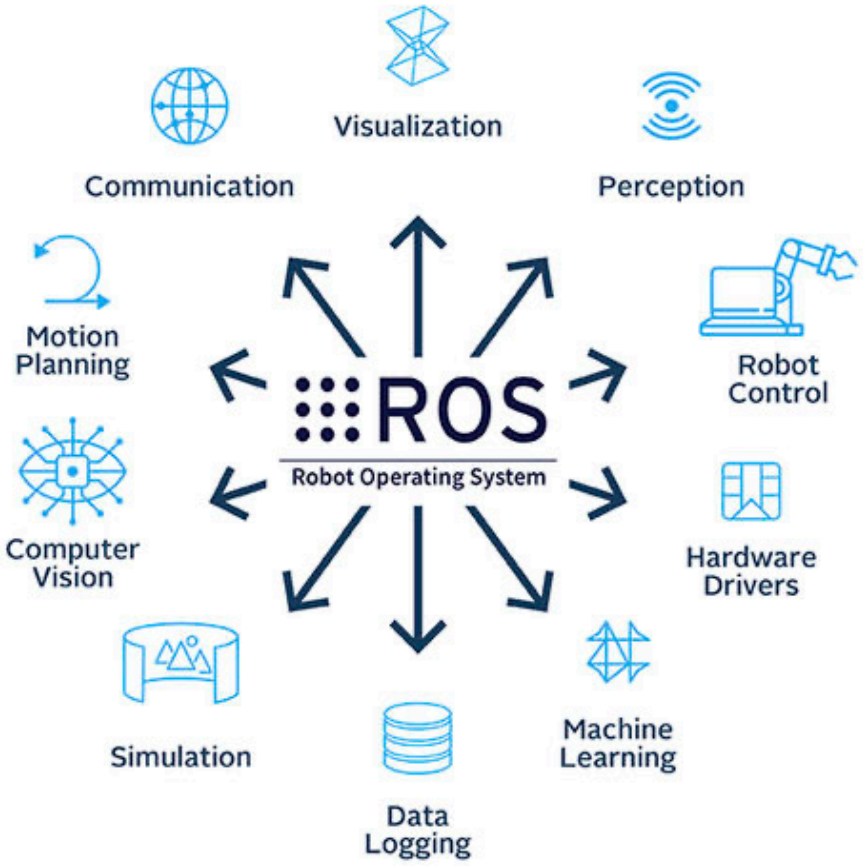
Introduction to ROS (Robot Operating System)

Jikai Wang

What is ROS?

- ROS (Robot Operating System) is an open source software development kit for robotics applications.
- ROS offers a standard software platform to developers across industries that will carry them from research and prototyping all the way through to deployment and production.



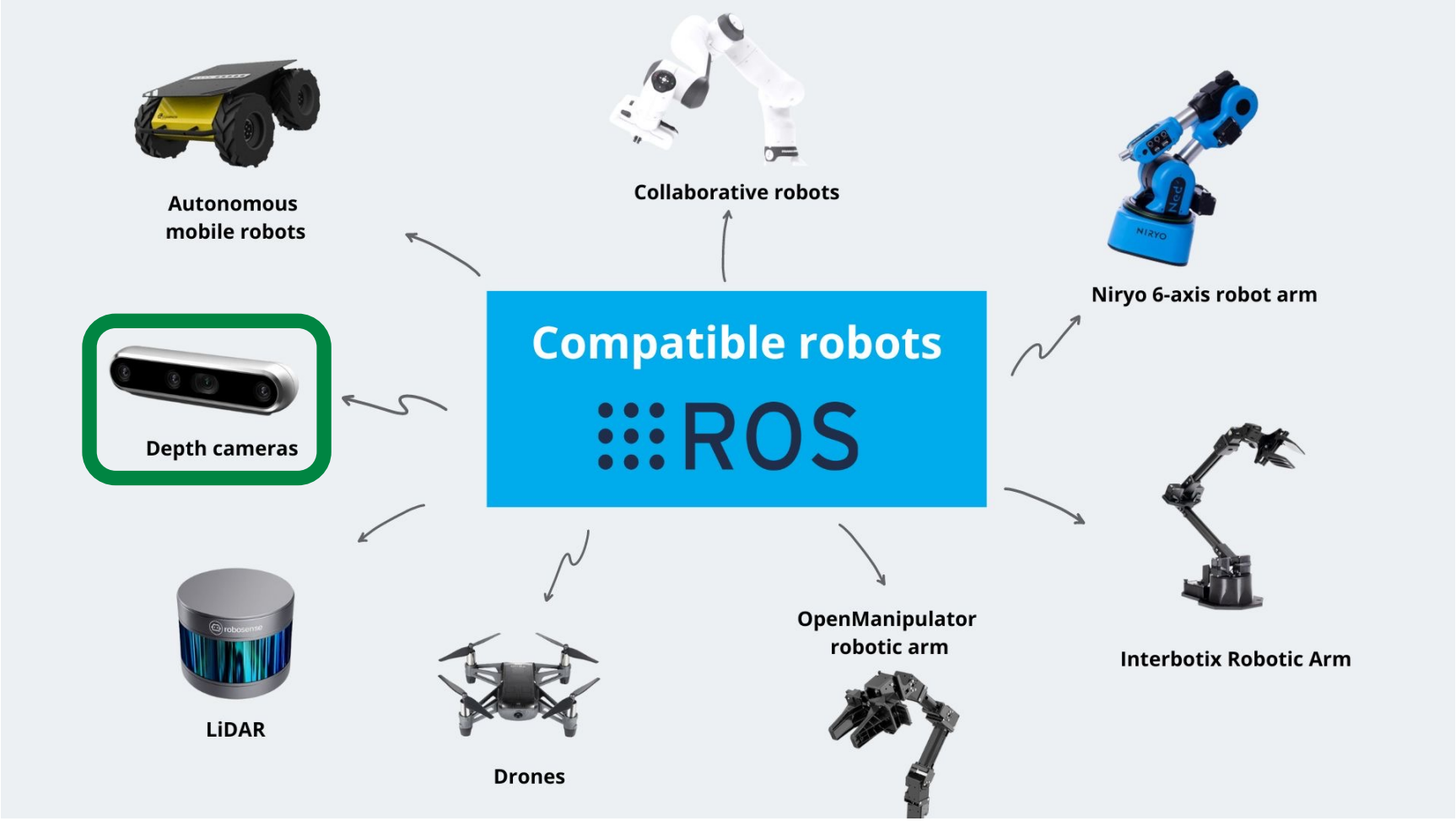
Why ROS?

- **Standardization**:
  - Offers a common platform for robotics research and development.
- **Flexibility**:
  - Supports a wide range of robot hardware and software.
- **Community**:
  - Large and active community contributing to its development and improvement.

Key Features of ROS

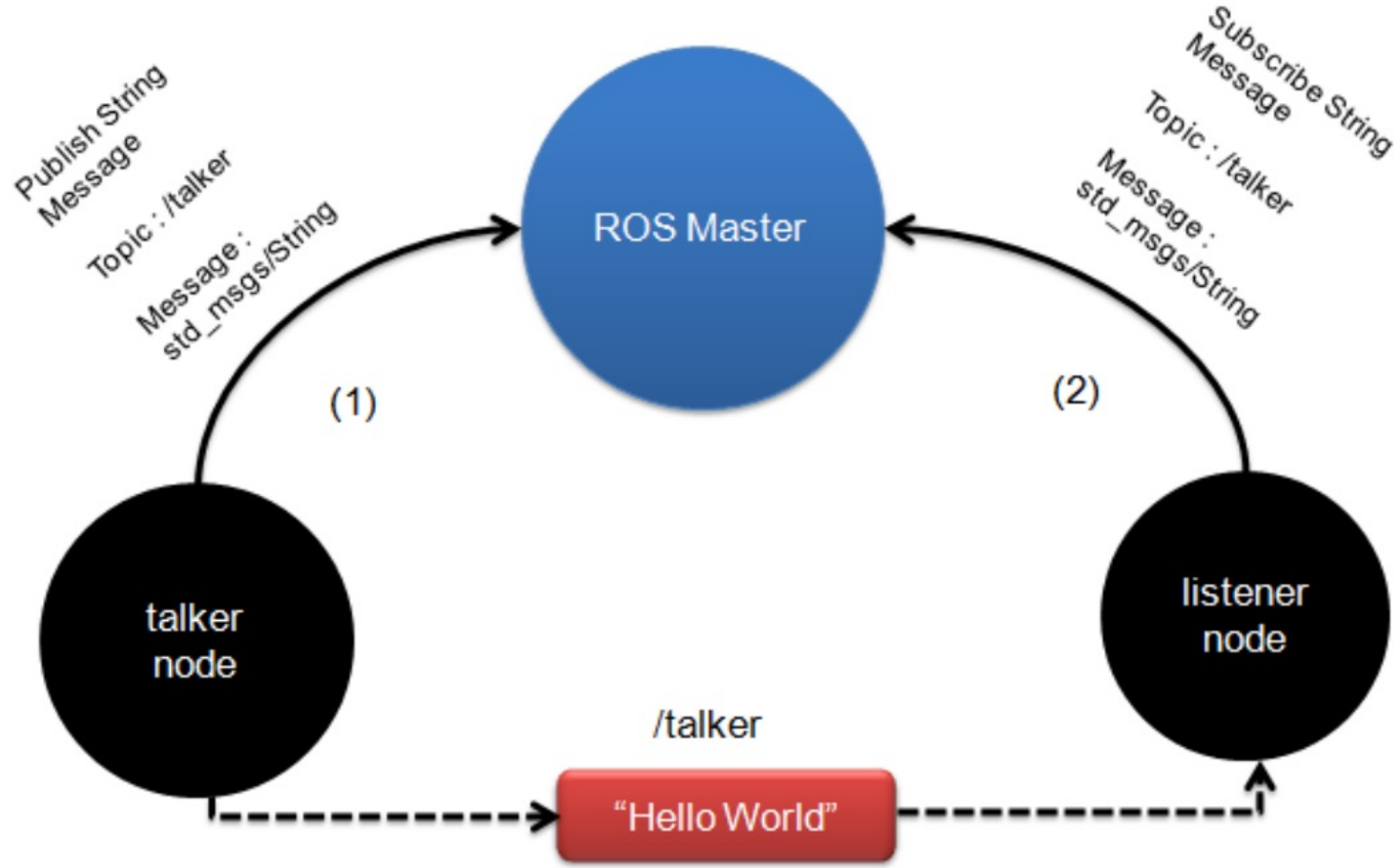
- Middleware:
  - Facilitates communication between different components of a robot.
- Reusable Code:
  - Enables code sharing and reuse through a package system.
- Simulation:
  - Allows testing and development in virtual environments before deploying on actual robots.

Robotics using ROS



ROS Core Concepts

- Nodes
- Topics
- Messages
- Services



ROS Nodes

- Single-purposed executable programs
  - e.g. sensor driver(s), actuator driver(s), mapper, planner, UI, etc.
- Individually compiled, executed, and managed
- Nodes are written using a ROS client library
  - roscpp: C++ client library
  - rospy: python client library
- Nodes can publish or subscribe to a Topic
- Nodes can also provide or use a Service

ROS Topics

- A topic is a name for a stream of messages with a defined type
  - e.g., data from a laser range-finder might be sent on a topic called scan, with a message type of LaserScan
- Nodes communicate with each other by publishing messages to topics
- Publish/Subscribe model: 1-to-N broadcasting

ROS Topics


Node Graph
Nodes/Topics (active)
/
/
Group: 3
Namespaces
Actions
tf
Images
Highlight
Fit
Hide:
Dead sinks
Leaf topics
Debug
tf
Unreachable
Params
/camera
/camera/realsense2_camera_manager
/camera/realsense2_camera_manager/bond
/camera/realsense2_camera
/camera/realsense2_camera_manager
/camera/color
/camera/color/image_raw
/camera/color/image_raw/image_topics
/camera/depth
/camera/depth/image_rect_raw
/camera/depth/image_rect_raw/image_topics

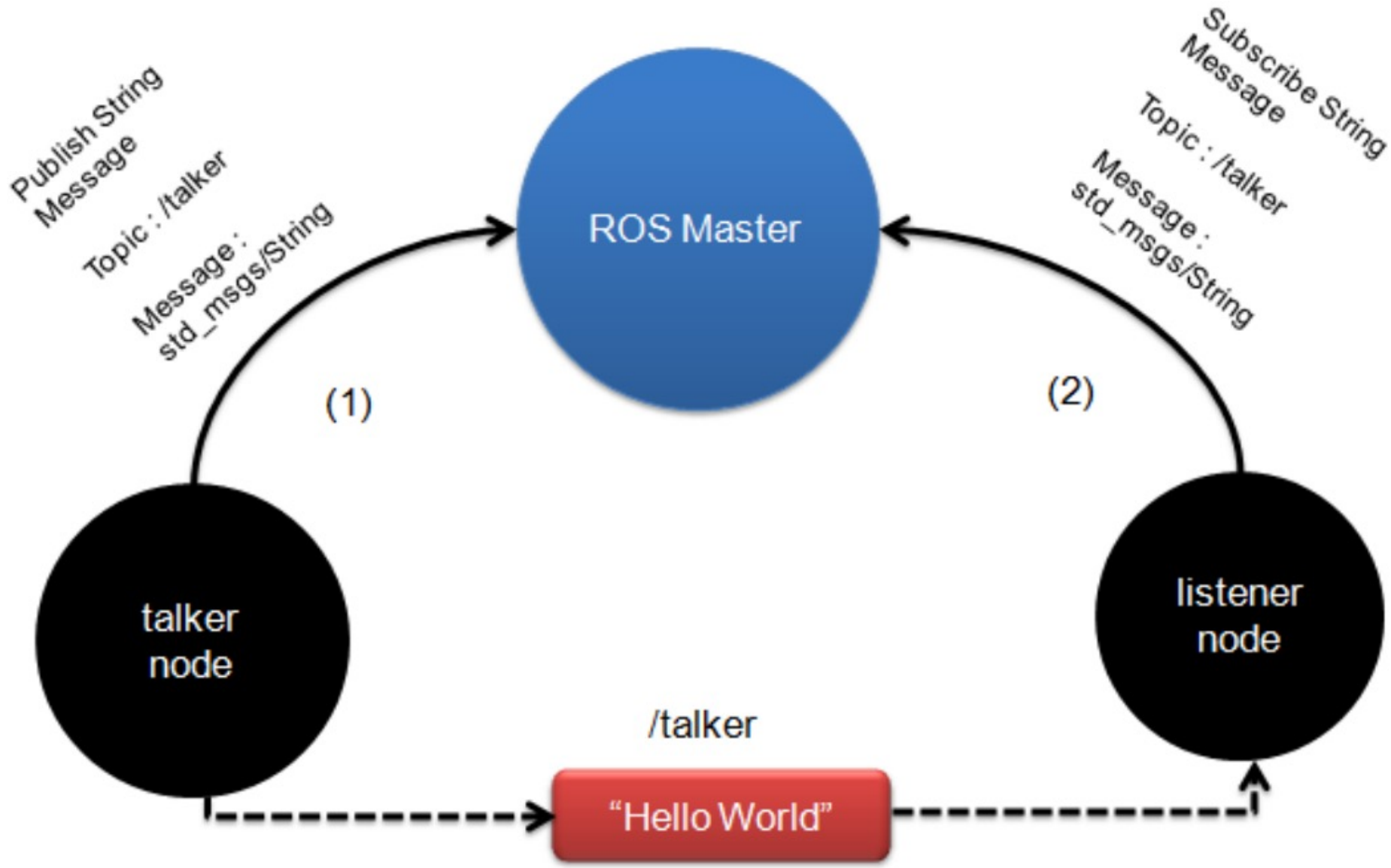
ROS Messages

- Data sent between nodes.
- Strictly-typed data structures for inter-node communication.
- Example:
  - ***sensor_msgs/Image***

```
Header header        # Header timestamp should be acquisition time of image
                     # Header frame_id should be optical frame of camera
                     # origin of frame should be optical center of camera
                     # +x should point to the right in the image
                     # +y should point down in the image
                     # +z should point into to plane of the image
                     # If the frame_id here and the frame_id of the CameraInfo
                     # message associated with the image conflict
                     # the behavior is undefined

uint32 height         # image height, that is, number of rows
uint32 width          # image width, that is, number of columns

# The legal values for encoding are in file src/image_encodings.cpp
# If you want to standardize a new string format, join
# ros-users@lists.sourceforge.net and send an email proposing a new encoding.

string encoding       # Encoding of pixels -- channel meaning, ordering, size
                      # taken from the list of strings in include/sensor_msgs/image_encodings.h

uint8 is_bigendian    # is this data bigendian?
uint32 step           # Full row length in bytes
uint8[] data          # actual matrix data, size is (step * rows)
```

ROS Services

- Request/reply communication between nodes.
- Synchronous inter-node transactions
- Service/Client model: 1-to-1 request-response
- Service roles:
  - carry out remote computation
  - trigger functionality / behavior
- Example:
  - ***map_server/static_map***: retrieves the current grid map used by the robot for navigation


Publish String
Message
Topic : /talker
Message :
std_msgs/String
ROS Master
Subscribe String
Message
Topic : /talker
Message :
std_msgs/String
(1)
(2)
talker
node
listener
node
/talker
"Hello World"

ROS Basic Commands

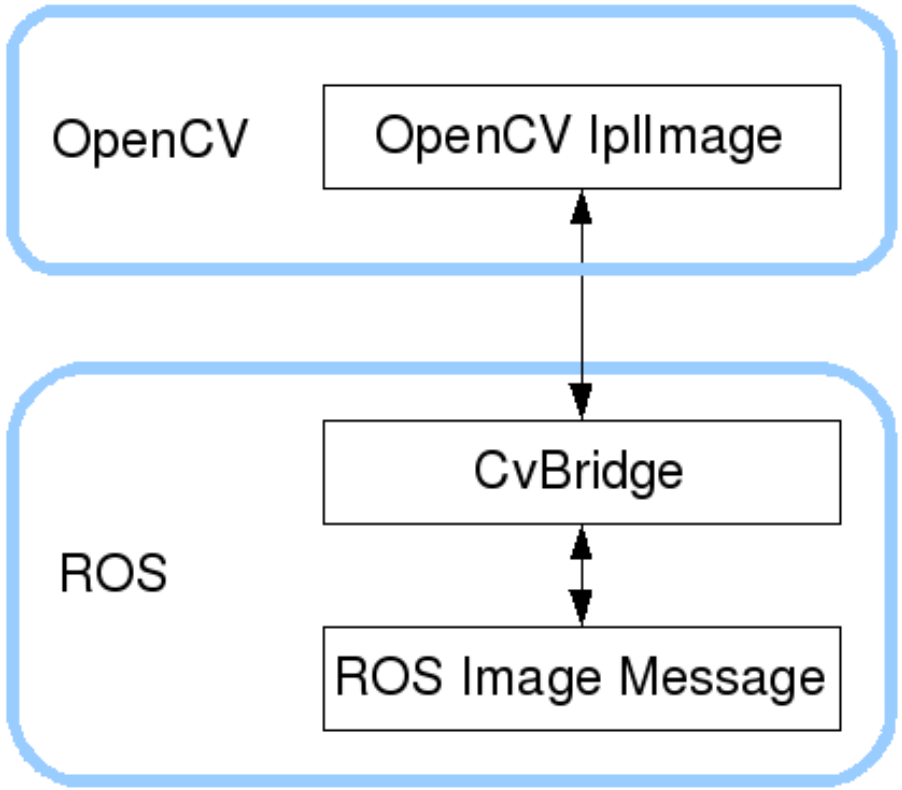
- *roscore*
  - roscore is the first thing you should run when using ROS
  - roscore will start up:
    - a ROS Master
    - a ROS Parameter Server
    - a rosout logging node
- *rosnode*
  - Displays debugging information about ROS nodes
- *rostopic*
  - Gives information about a topic and allows to publish messages on a topic
- *rosmsg*
  - Displaying information about ROS Message types

ROS Basic Commands

- *roslaunch*
  - a tool for easily launching multiple ROS nodes as well as setting parameters on the Parameter Server
- *rosbag*
  - for recording from and playing back to ROS topics
- *rviz*
  - rviz is a ROS 3D visualization tool that lets you see the world from a robot's perspective

ROS and OpenCV

- ROS passes images in its own ***sensor_msgs/Image*** message
- **cv_bridge** is a ROS package that provides functions to convert between ROS ***sensor_msgs/Image*** messages and the objects used by OpenCV

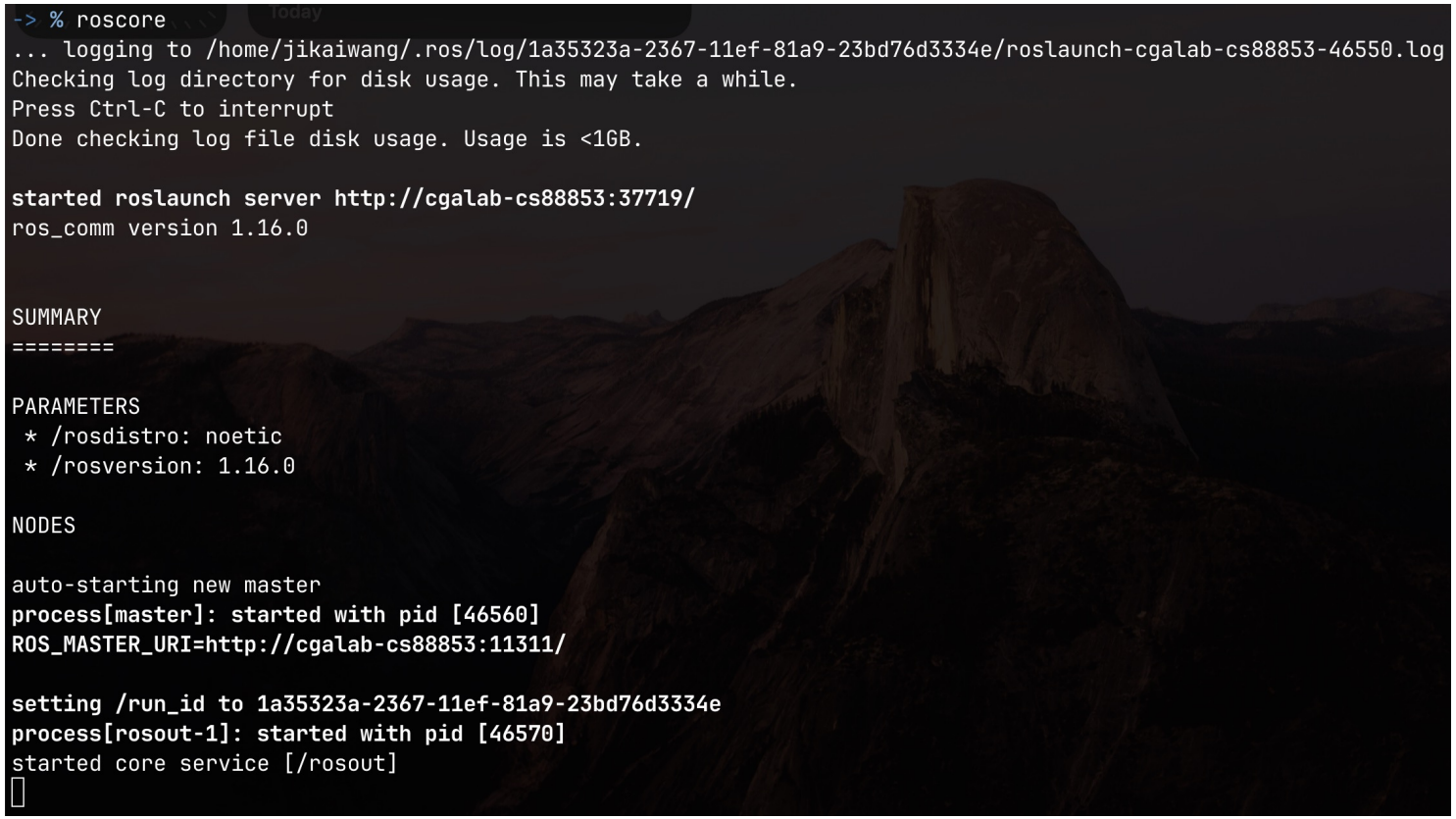


Synchronize Messages across all Cameras

- message_filters.ApproximateTimeSynchronizer()
  - Algorithms: https://wiki.ros.org/message_filters/ApproximateTime
  - Example: https://gist.github.com/zxf8665905/2d09d25da823b0f7390cab83c64d631a

Record Rosbag from Multiple Cameras

- Run ***roscore*** to start the Master Node



Record Rosbag from Multiple Cameras

- Run ***roslaunch*** to start the Camera Nodes

```
roslaunch ${RS_LAUNCH_FILE} \
    serial_no:=${CAM} camera:=${CAM} \
    depth_width:=${RS_WIDTH} depth_height:=${RS_HEIGHT} depth_fps:=${RS_FPS} \
    color_width:=${RS_WIDTH} color_height:=${RS_HEIGHT} color_fps:=${RS_FPS} \
    align_depth:=${RS_ALIGN_DEPTH}
```

- Run ***rosbag*** to start the Camera Nodes

```
rosbag record \
    --regex "(.*)/(aligned_depth_to_color|color)/image_raw" \
    --buffsize=8192 \
    --output-name=${SAVE_PATH}
```

Reference

- ROS commands
  - rosnode: https://wiki.ros.org/rosnode?distro=noetic
  - rostopic: https://wiki.ros.org/rostopic
  - rosbag: https://wiki.ros.org/rosbag?distro=noetic
  - roslaunch: https://wiki.ros.org/roslaunch?distro=noetic
  - cv_bridge: https://wiki.ros.org/cv_bridge
- Example to use ApproximateTimeSynchronizer for message synchronization
  - https://gist.github.com/zxf8665905/2d09d25da823b0f7390cab83c64d631a

THE UNIVERSITY OF TEXAS AT DALLAS

Thank You