

ROSpec: A Domain-Specific Language for ROS-based Robot Software



Paulo Canelas

Carnegie Mellon University

University of Lisbon



Bradley Schmerl

Carnegie Mellon University



Alcides Fonseca

University of Lisbon



Christopher S. Timperley

Carnegie Mellon University

This talk is about **(preventing)** robots crashing



PORTUGAL

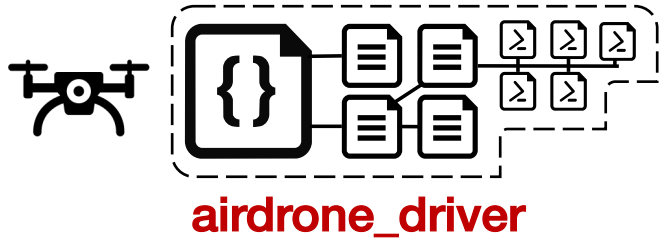
MARINHA PORTUGUESA
Novo drone ao serviço da armada

The Robot Operating System (ROS) improves development by focusing on **component reuse**

*“We have designed ROS to support our **philosophy of modular**, tools-based software development”*
[Quigley et al, 2009]

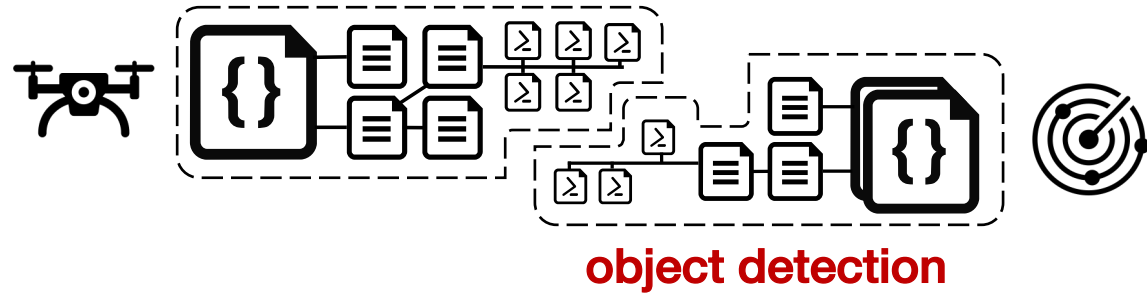
The Robot Operating System (ROS) improves development by focusing on **component reuse**

*“We have designed ROS to support our **philosophy of modular**, tools-based software development”*
[Quigley et al, 2009]



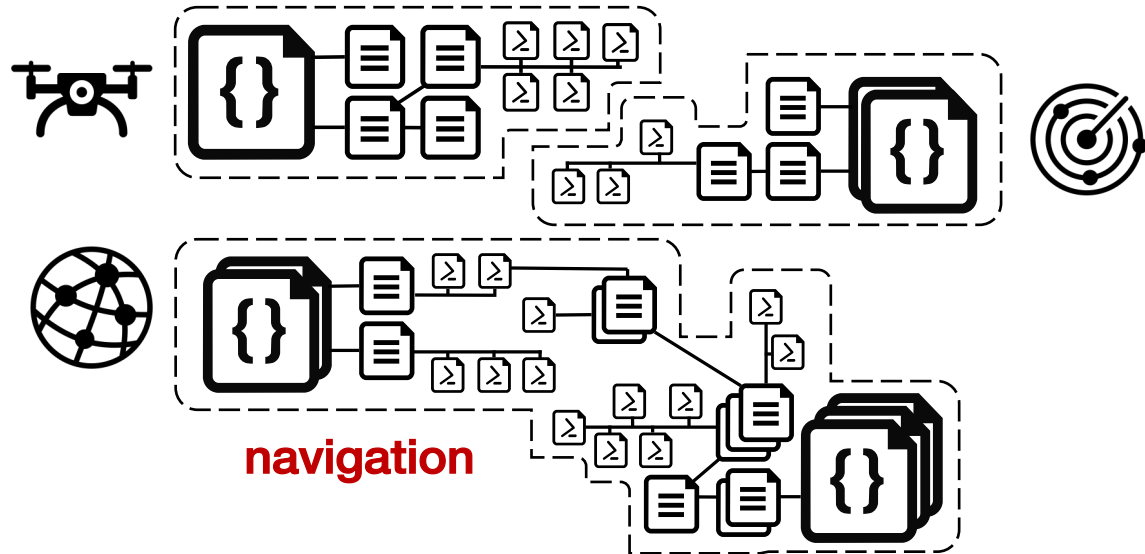
The Robot Operating System (ROS) improves development by focusing on **component reuse**

*“We have designed ROS to support our **philosophy of modular**, tools-based software development”*
[Quigley et al, 2009]



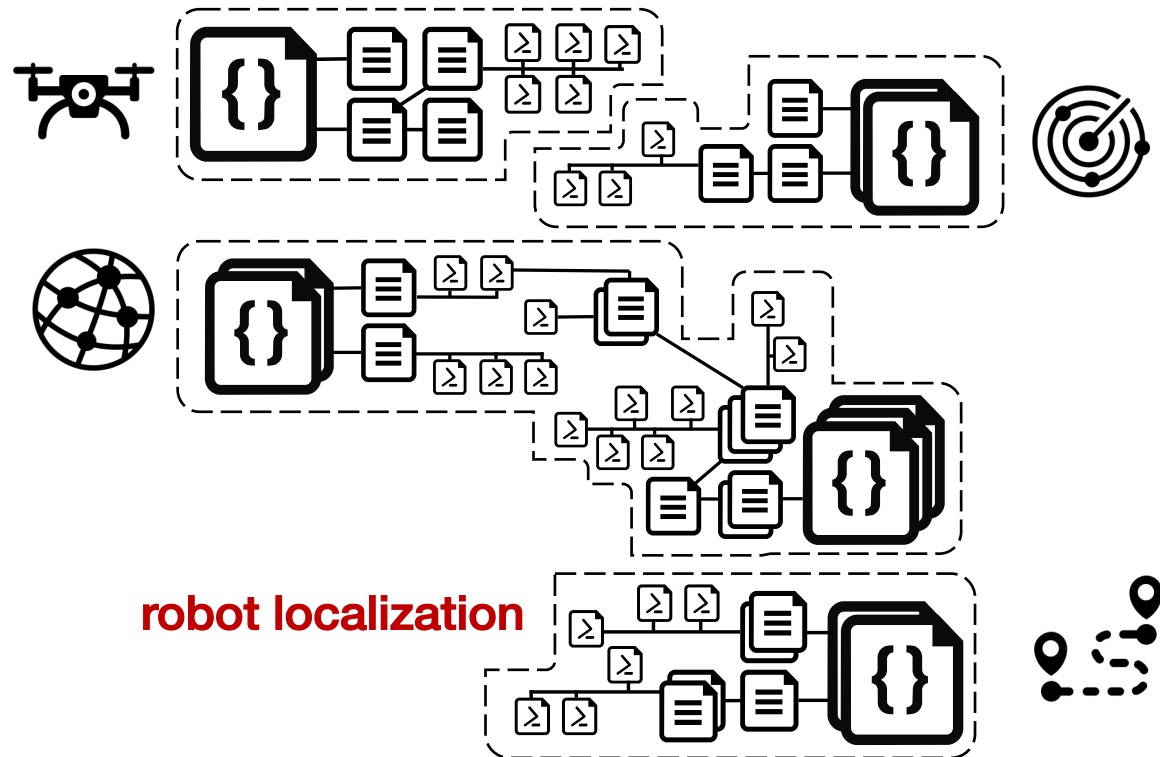
The Robot Operating System (ROS) improves development by focusing on **component reuse**

*“We have designed ROS to support our **philosophy of modular**, tools-based software development”*
[Quigley et al, 2009]



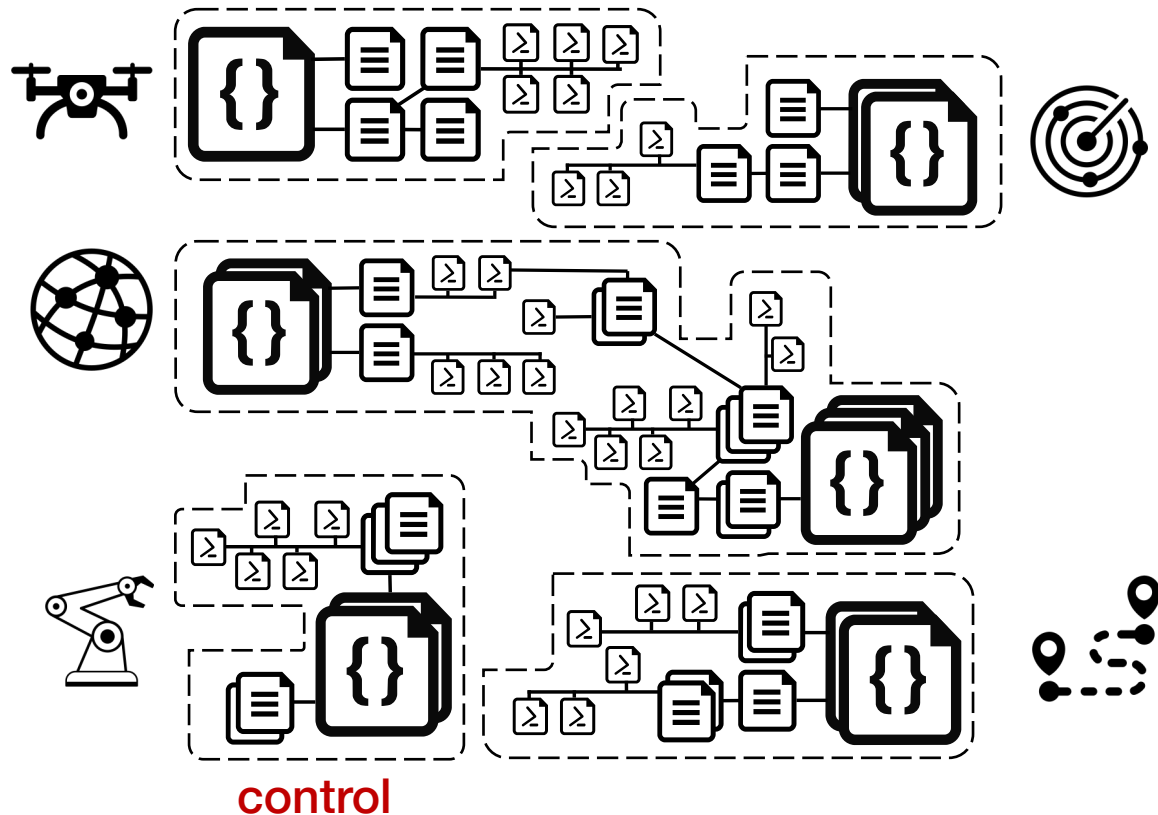
The Robot Operating System (ROS) improves development by focusing on **component reuse**

*“We have designed ROS to support our **philosophy of modular**, tools-based software development”*
[Quigley et al, 2009]



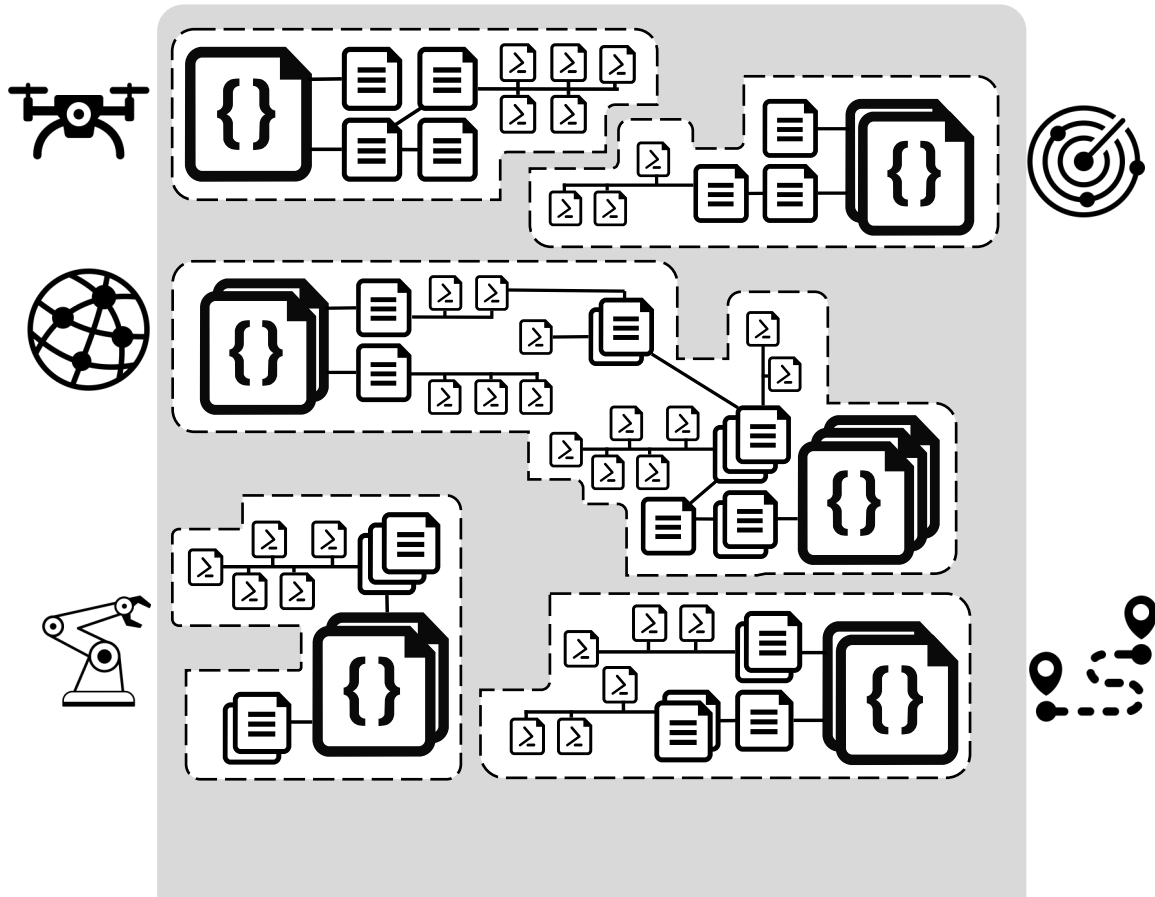
The Robot Operating System (ROS) improves development by focusing on **component reuse**

*“We have designed ROS to support our **philosophy of modular**, tools-based software development”*
[Quigley et al, 2009]



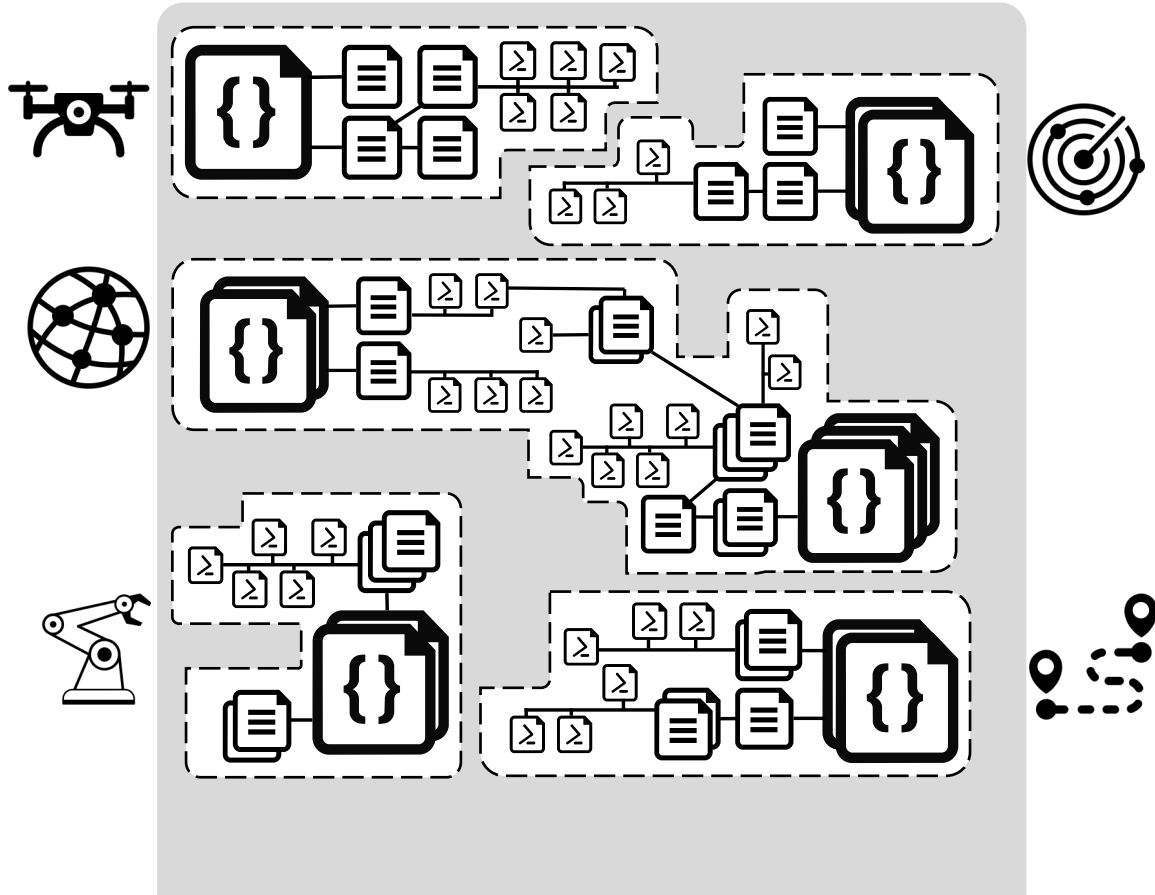
The Robot Operating System (ROS) improves development by focusing on **component reuse**

*“We have designed ROS to support our **philosophy of modular**, tools-based software development”*
[Quigley et al, 2009]

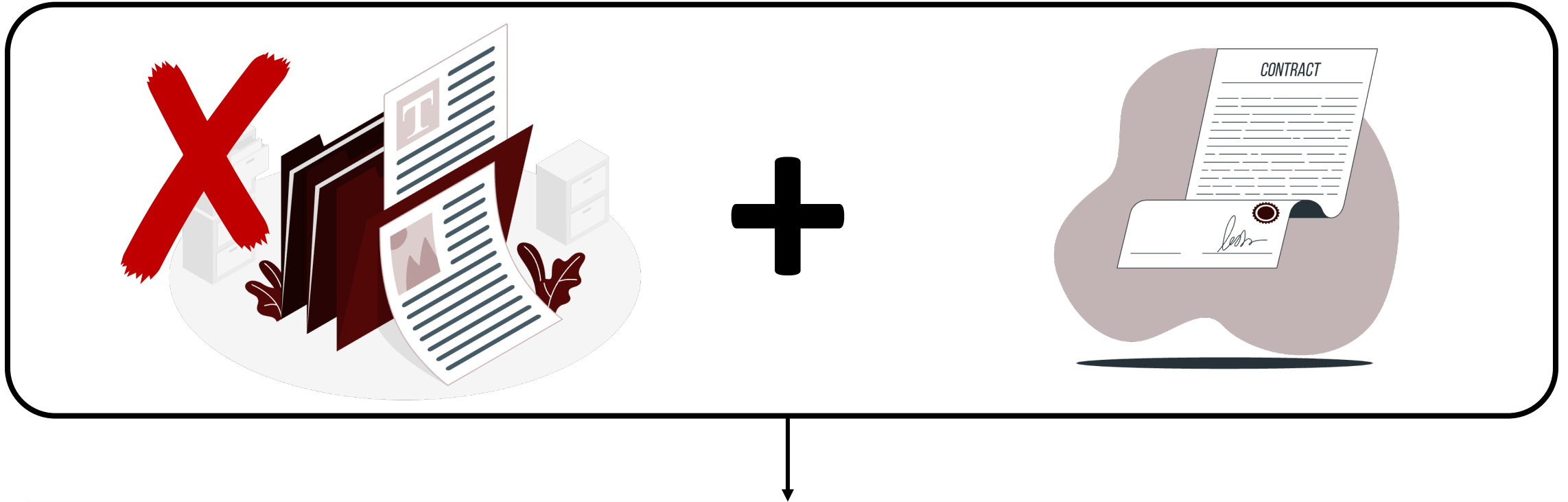


The Robot Operating System (ROS) improves development by focusing on **component reuse**

*“We have designed ROS to support our **philosophy of modular**, tools-based software development”*
[Quigley et al, 2009]



However, unchecked component assumptions and lack of documentation lead to misconfigurations



Misconfigurations arise from mismatched expectations and guarantees when configuring and integrating components

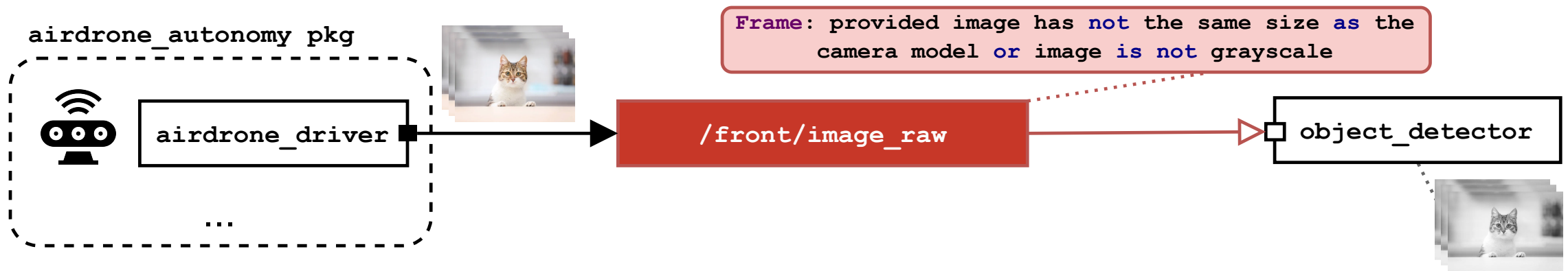
The airdrone_driver sends sensor data, while the object_detector receives and processes it



Legend



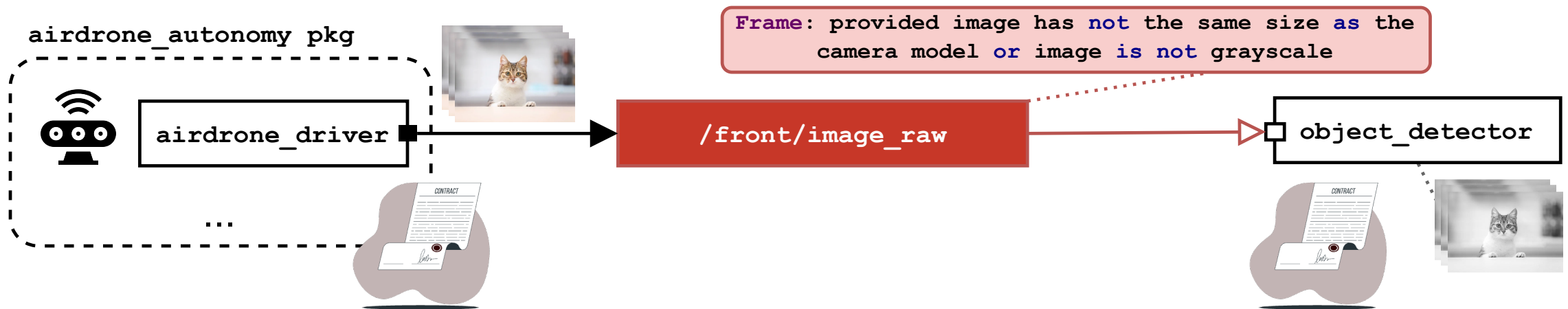
Missing semantic information regarding image format leads to image color mismatches



Legend



Missing semantic information regarding image format leads to image color mismatches

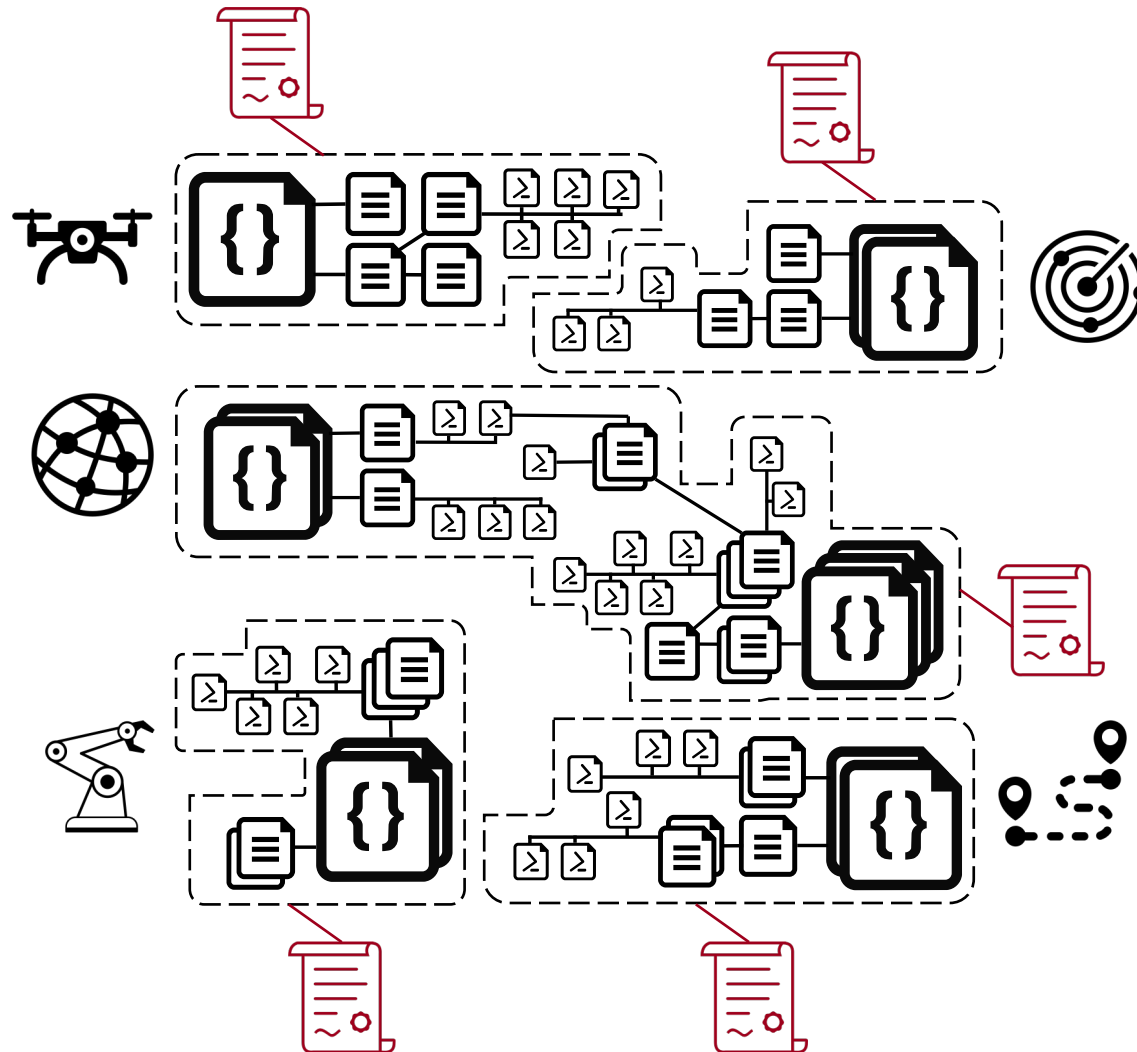


Legend

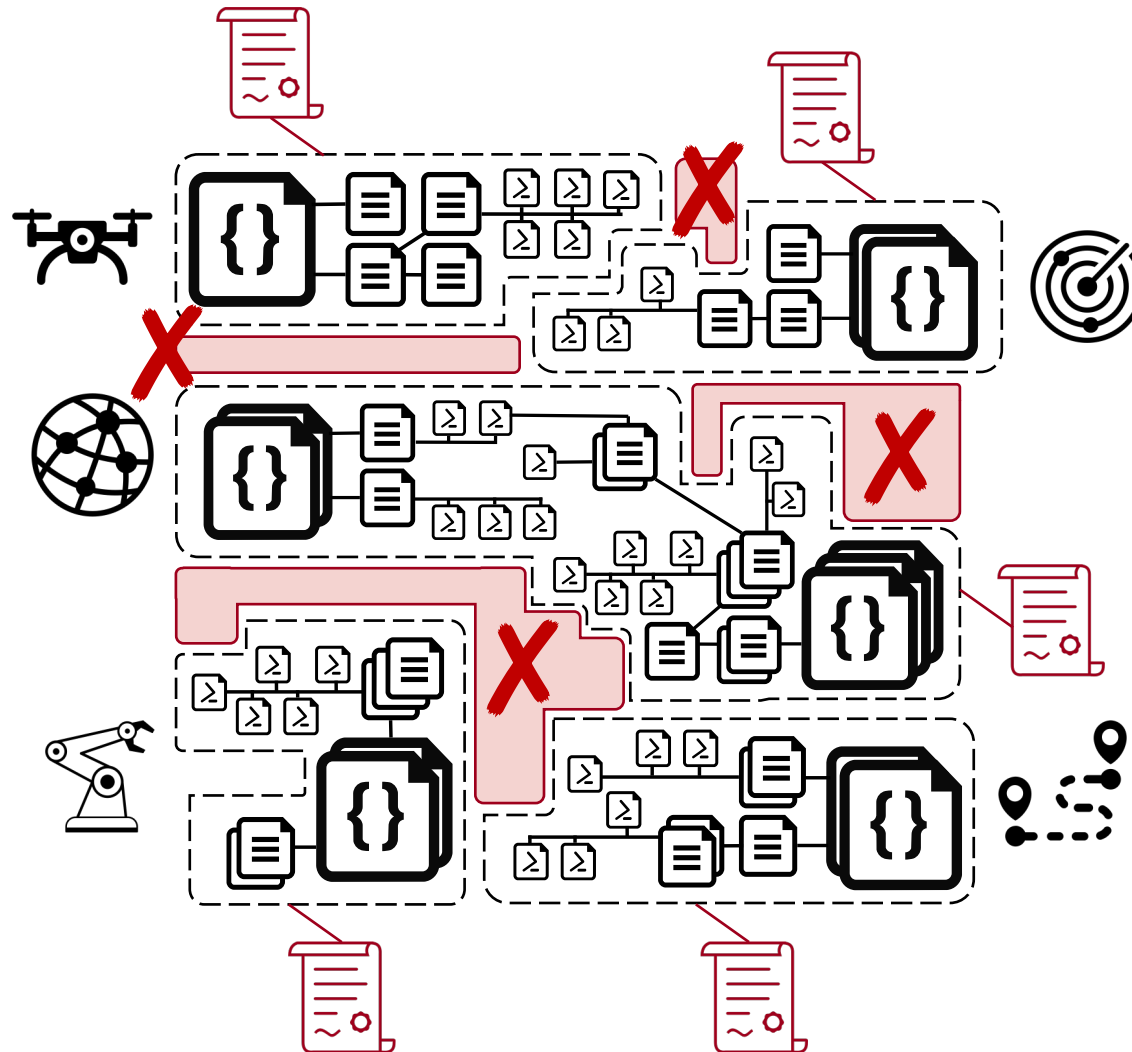


By specifying **{properties}** over component configurations and their integration, we can detect misconfigurations prior to deployment

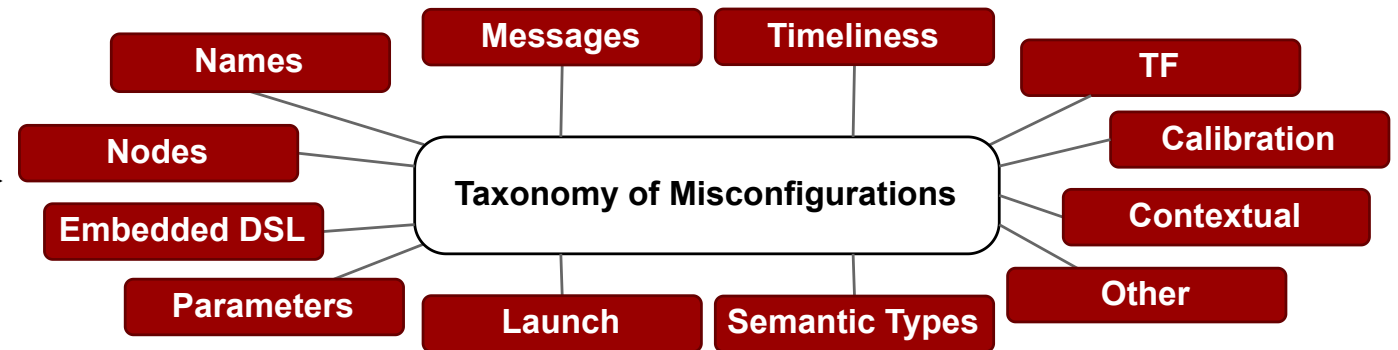
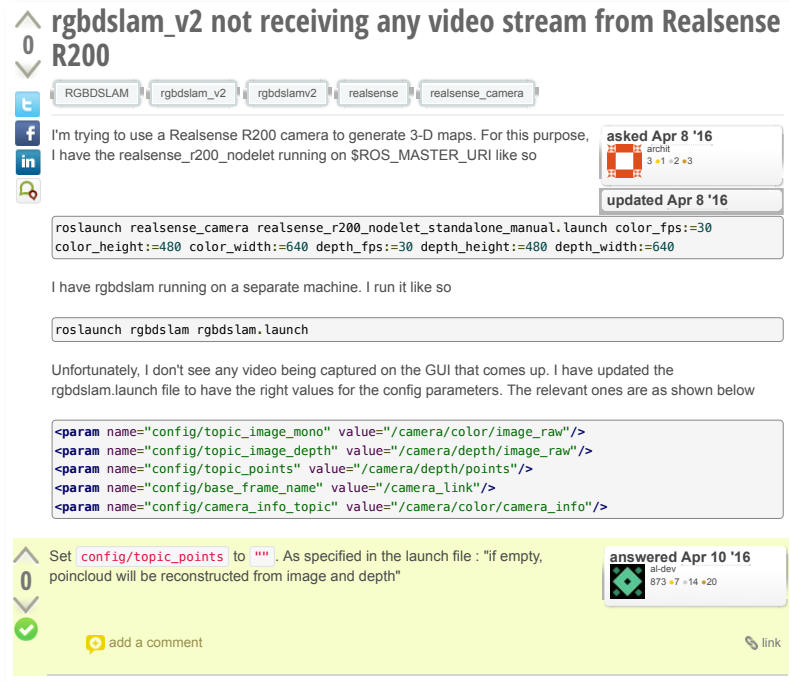
ROSpec: A specification language for refining ROS components configurations and integration



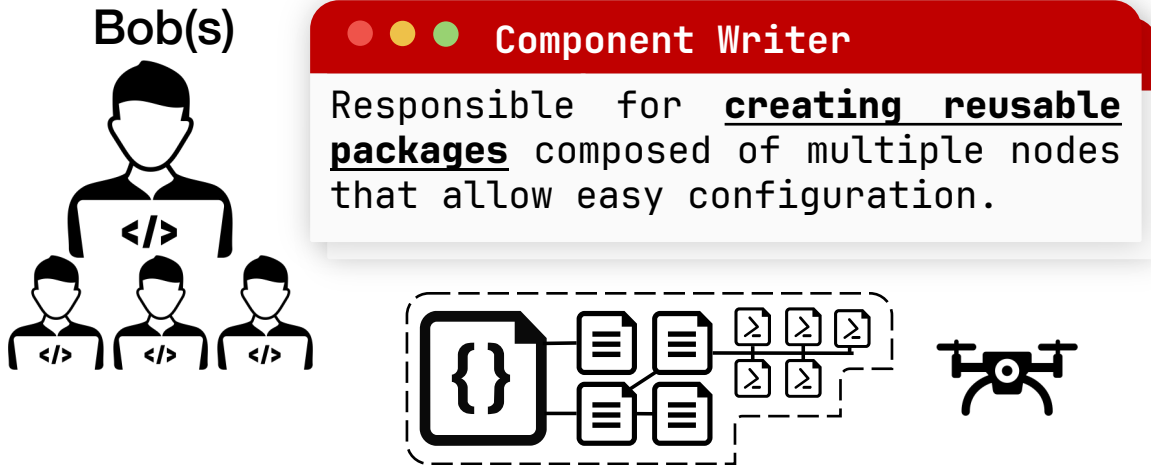
ROSpec: A specification language for refining ROS components configurations and integration



ROSpec design is based on prior work in misconfigurations [1]



ROSpec design is based on prior work in misconfigurations and two primary stakeholders



Component Source-Code

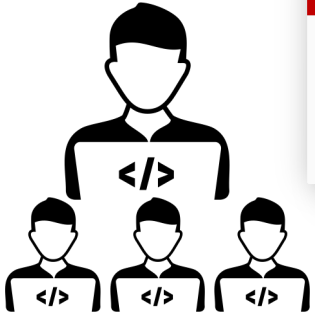
```
class VelocityPublisher:
    def __init__(self):
        rospy.init_node('velocity_publisher')

        self.min_vel_x = rospy.get_param('~min_vel_x', 0.1)
        self.max_vel_x = rospy.get_param('~max_vel_x', 0.5)

        self.pub = rospy.Publisher('/cmd_vel', Twist)
```

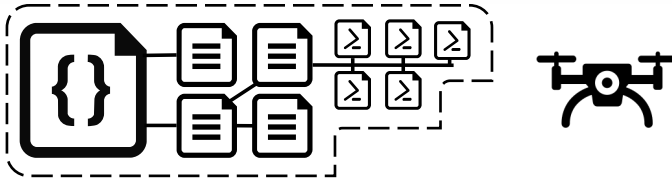
ROSpec design is based on prior work in misconfigurations and two primary stakeholders

Bob(s)



Component Writer

Responsible for creating reusable packages composed of multiple nodes that allow easy configuration.



Alice



Component Integrator

Who select and adapts components to meet system requirements, while ensuring that configurations match the component assumptions

ROS Configuration File

```
max_vel_x: 0.5
min_vel_x: 0.1
global_frame: "odom"
robot_base_frame: "base_link"
update_frequency: 5.0
static_map: true
```

Component Source-Code

```
class VelocityPublisher:
    def __init__(self):
        rospy.init_node('velocity_publisher')

        self.min_vel_x = rospy.get_param('~min_vel_x', 0.1)
        self.max_vel_x = rospy.get_param('~max_vel_x', 0.5)

        self.pub = rospy.Publisher('/cmd_vel', Twist)
```

We enforce type restrictions and parameter dependencies using liquid and dependent types

Component Writer

```
node type move_group {  
  param max_acceleration: double where {_ >= 0};  
  
  optional param max_velocity: double = 1.2211;  
  optional param has_velocity_limits: bool = false;  
  
} where {  
  exists(max_velocity) -> exists(has_velocity_limits);  
}
```

Component Integrator

```
system {  
  node instance mv_gp: move_group {  
    param max_acceleration = 0.0;  
    param max_velocity = 3.14;  
  }  
}
```

We enforce type restrictions and parameter dependencies using liquid and dependent types

```
Component Writer

node type move_group {
  param max_acceleration: double where {_ >= 0};

  optional param max_velocity: double = 1.2211;
  optional param has_velocity_limits: bool = false;
} where {
  exists(max_velocity) -> exists(has_velocity_limits);
}
```

Liquid Type: Only allow positive max accelerations

```
Component Integrator

system {
  node instance mv_gp: move_group {
    param max_acceleration = 0.0;
    param max_velocity = 3.14;
  }
}
```

In-Value Dependent Type: If max_velocity is set, then has_velocity_limits must be set

We enforce type restrictions and parameter dependencies using liquid and dependent types

Liquid Type: Only allow positive max accelerations

Component Writer

```
node type move_group {  
  param max_acceleration: double where {_ >= 0};  
  
  optional param max_velocity: double = 1.2211;  
  optional param has_velocity_limits: bool = false;  
  
} where {  
  exists(max_velocity) -> exists(has_velocity_limits)  
}
```

Component Integrator

```
system {  
  node instance mv_gp: move_group {  
    param max_acceleration = 0.0;  
    param max_velocity = 3.14;  
  }  
}
```

Dependency exists(max_acceleration) ->
exists(has_acceleration_limits) not satisfied in move_group

In-Value Dependent Type: If max_velocity is set, then has_velocity_limits must be set

Liquid types are also used to structurally refine the architecture of the system

Component Writer

```
node type openni_node {  
  optional param depth_frame_id: string = "/openni_depth_optical_frame";  
  optional param rgb_frame_id: string = "/openni_rgb_optical_frame";  
  
  publishes to camera/depth/points: sensor_msgs/PointCloud2 where { count(publishers(_)) == 1 };  
}
```

Liquid types are also used to structurally refine the architecture of the system

```
Component Writer

node type openni_node {
  optional param depth_frame_id: string = "/openni_depth_optical_frame";
  optional param rgb_frame_id: string = "/openni_rgb_optical_frame";

  publishes to camera/depth/points: sensor_msgs/PointCloud2 where { count(publishers(_)) == 1 };
}
```

Liquid Type: There must only be one publisher to the topic



Liquid types are also used to structurally refine the architecture of the system

```
Component Writer

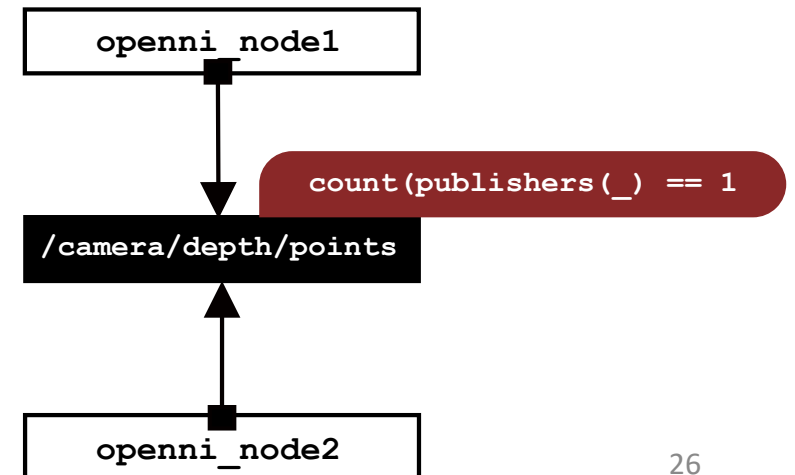
node type openni_node {
  optional param depth_frame_id: string = "/openni_depth_optical_frame";
  optional param rgb_frame_id: string = "/openni_rgb_optical_frame";

  publishes to camera/depth/points: sensor_msgs/PointCloud2 where { count(publishers(_)) == 1 };
}
```

Liquid Type: There must only be one publisher to the topic

```
Component Integrator

system {
  node instance openni_node1: openni_node { ... }
  node instance openni_node2: openni_node { ... }
}
```



Policies refine component connections by providing semantic information

Component Writer 1

```
node type openni_camera_driver {  
  @qos{effort1qos}  
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Component Writer 2

```
node type custom_node {  
  @qos{reliable5qos}  
  subscribes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Policies refine component connections by providing semantic information

Component Writer 1

```
node type openni_camera_driver {  
  @qos{effort1qos}  
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Component Writer 2

```
node type custom_node {  
  @qos{reliable5qos}  
  subscribes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Quality of Service (**QoS**) policies allow you to tune communication between nodes
(*history, depth, reliability, durability, deadline, ...*)

Policies refine component connections by providing semantic information

Component Writer 1

```
node type openni_camera_driver {  
  @qos{effort1qos}  
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Component Writer 2

```
node type custom_node {  
  @qos{reliable5qos}  
  subscribes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Quality of Service (**QoS**) policies allow you to tune communication between nodes
(history, depth, reliability, durability, deadline, ...)

Publisher	Subscription	Compatible
Best effort	Best effort	Yes
Best effort	Reliable	No
Reliable	Best effort	Yes
Reliable	Reliable	Yes

Policies refine component connections by providing semantic information

Component Writer 1

```
node type openni_camera_driver {  
  @qos{effort1qos}  
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Component Writer 2

```
node type custom_node {  
  @qos{reliable5qos}  
  subscribes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Quality of Service (**QoS**) policies allow you to tune communication between nodes
(*history, depth, reliability, durability, deadline, ...*)

Publisher	Subscription	Compatible
Best effort	Best effort	Yes
Publisher	Subscription	Compatible
Default	Default	Yes
Default	x	No
x	Default	Yes
x	x	Yes
x	y (where $y > x$)	Yes
x	y (where $y < x$)	No

Policies refine component connections by providing semantic information

●●● Component Writer 1

```
node type openni_camera_driver {
  @qos{effort1qos}
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;
}
```

●●● Component Writer 2

```
node type custom_node {
  @qos{reliable5qos}
  subscribes to /camera/rgb/image_raw: sensor_msgs/Image;
}
```

Quality of Service (QoS) policies allow you to tune communication between nodes
(history, depth, reliability, durability, deadline, ...)

Publisher		Subscription	Compatible
Default	Automatic	Automatic	Yes
	Automatic	Manual by topic	No
	Manual by topic	Automatic	Yes
	Manual by topic	Manual by topic	Yes
x	x	Yes	
x	y (where y > x)	Yes	
x	y (where y < x)	No	

Policies refine component connections by providing semantic information

Component Writer 1

```
node type openni_camera_driver {  
  @qos{effort1qos}  
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Component Writer 2

```
node type custom_node {  
  @qos{reliable5qos}  
  subscribes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Quality of Service (**QoS**) policies allow you to tune communication between nodes
(*history, depth, reliability, durability, deadline, ...*)

Publisher			
Publisher		Subscription	Compatible
Default	Default	Default	Yes
Default	x	Default	No
x	Default	Default	Yes
x	x	x	Yes
x	x	y (where y > x)	Yes
x	x	y (where y < x)	No

Policies refine component connections by providing semantic information

●●● Component Writer 1

```
node type openni_camera_driver {  
  @qos{effort1qos}  
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

●●● Component Writer 2

```
node type custom_node {  
  @qos{reliable5qos}  
  subscribes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

Quality of Service (QoS) policies allow you to tune communication between nodes
(history, depth, reliability, durability, deadline, ...)

Publisher	Subscription	Compatible
Default	Default	Yes
Default	x	No
x	Default	Yes
x	x	No
where y > x		Yes
where y < x		No

Publisher	Subscription	Compatible	Result
Volatile	Volatile	Yes	New messages only
Volatile	Transient local	No	No communication
Transient local	Volatile	Yes	New messages only
Transient local	Transient local	Yes	New and old messages

Policies refine component connections by providing semantic information

Component Writer 1

```
node_type openni_camera_driver {  
  @qos{effort1qos}  
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

“ Figuring out the right QoS settings to use for nodes is also a pain, since certain combinations are not compatible with others, so for every nodes, you have to consult the compatibility matrix and make sure that all subscribers and publishers agree with each other. [2]

Quality of Service (QoS) policies allow you to tune communication between nodes (history, depth, reliability, durability, deadline, ...)

	Publisher	Subscription	Compatible
Default			

	Publisher	Subscription	Compatible
Default			

	Publisher	Subscription	Compatible
Default			

	Publisher	Subscription	Compatible
Default			

	Publisher	Subscription	Compatible
Default			

	Publisher	Subscription	Compatible
Default			

	Publisher	Subscription	Compatible
Default			

Policies refine component connections by providing semantic information

Component Writer 1

```
node type openni_camera_driver {  
  @qos{effort1qos}  
  publishes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

```
policy instance effort1qos: qos {  
  setting reliability = BestEffort;  
  setting depth = 1;  
}
```

Component Writer 2

```
node type custom_node {  
  @qos{reliable5qos}  
  subscribes to /camera/rgb/image_raw: sensor_msgs/Image;  
}
```

```
policy instance reliable5qos: qos {  
  setting reliability = Reliable;  
  setting depth = 1;  
}
```



Component Integrator

```
system {  
  node instance node: custom_node { }  
  node instance openni_node: openni_camera_driver { }  
}
```

Evaluation

We evaluate ROSpec on a medium-sized robot and a dataset of real-world misconfigurations



0 rgbdsлам_v2 not receiving any video stream from Realsense R200

RGBDSLAM rgbdsлам_v2 rgbdsламv2 realsense realsense_camera

I'm trying to use a Realsense R200 camera to generate 3-D maps. For this purpose, I have the realsense_r200_nodelet running on \$ROS_MASTER_URI like so

asked Apr 8 '16
archit
3 * 1 = 2 * 3

updated Apr 8 '16

```
roslaunch realsense_camera realsense_r200_nodelet_standalone_manual.launch color_fps:=30  
color_height:=480 color_width:=640 depth_fps:=30 depth_height:=480 depth_width:=640
```

I have rgbdsлам running on a separate machine. I run it like so

```
roslaunch rgbdsлам rgbdsлам.launch
```

Unfortunately, I don't see any video being captured on the GUI that comes up. I have updated the rgbdsлам.launch file to have the right values for the config parameters. The relevant ones are as shown below

```
<param name="config/topic_image_mono" value="/camera/color/image_raw"/>  
<param name="config/topic_image_depth" value="/camera/depth/image_raw"/>  
<param name="config/topic_points" value="/camera/depth/points"/>  
<param name="config/base_frame_name" value="/camera_link"/>  
<param name="config/camera_info_topic" value="/camera/color/camera_info"/>
```

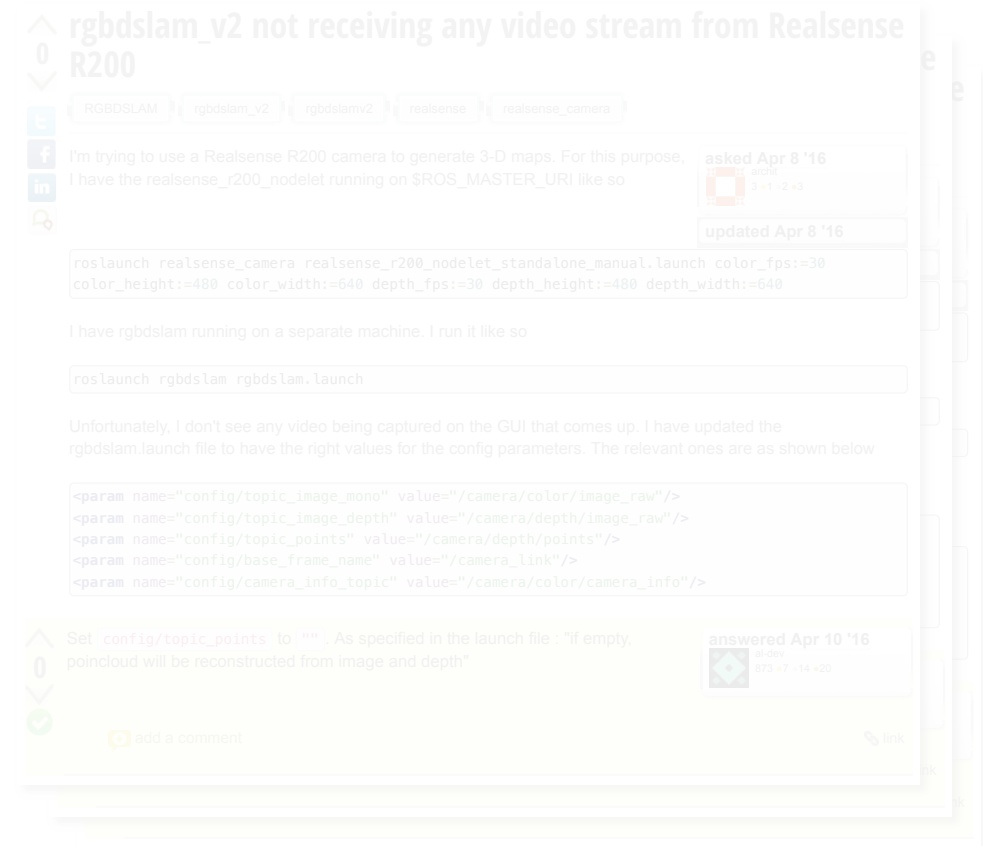
0 Set config/topic_points to "". As specified in the launch file : "if empty, poincloud will be reconstructed from image and depth"

answered Apr 10 '16
all-dev
873 * 7 = 14 * 20

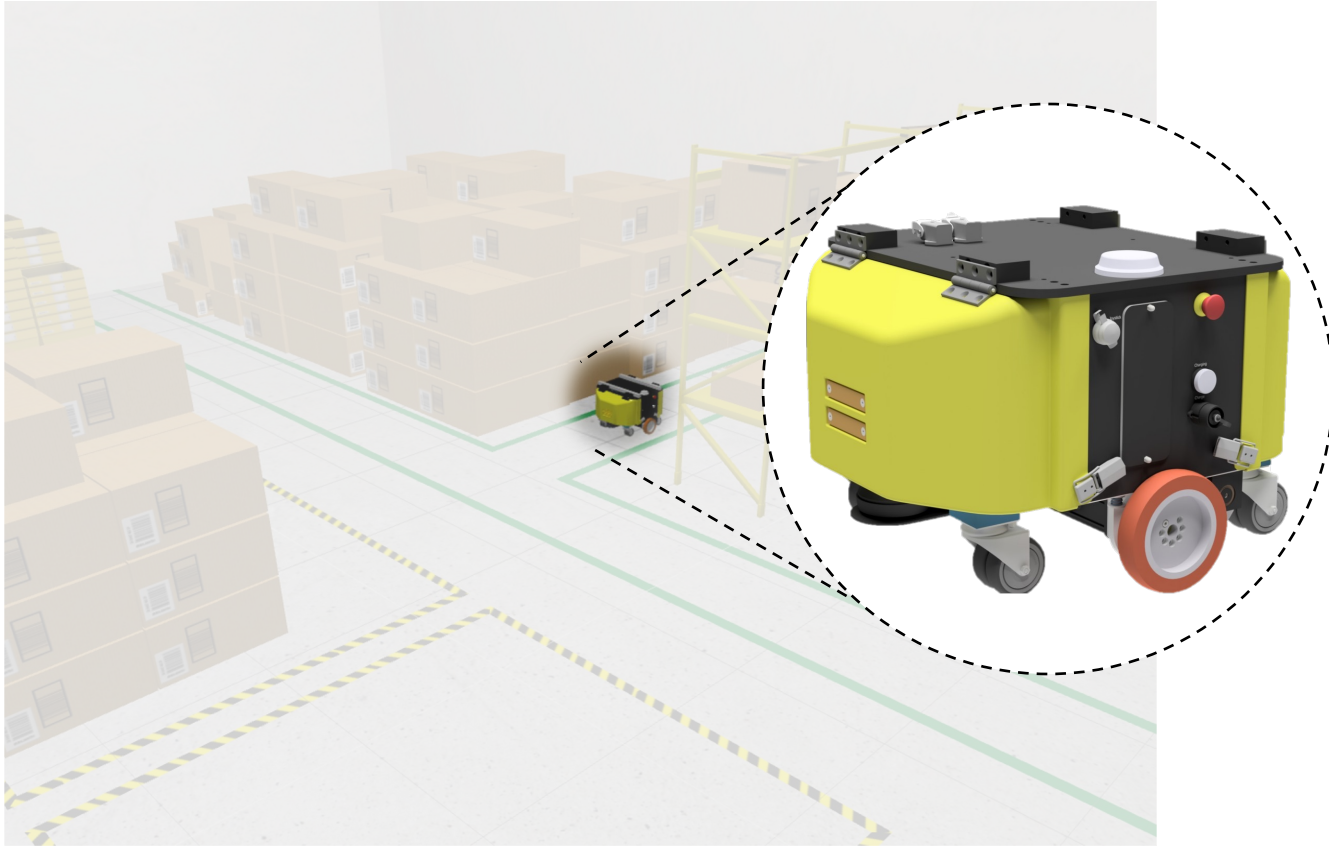
add a comment

link

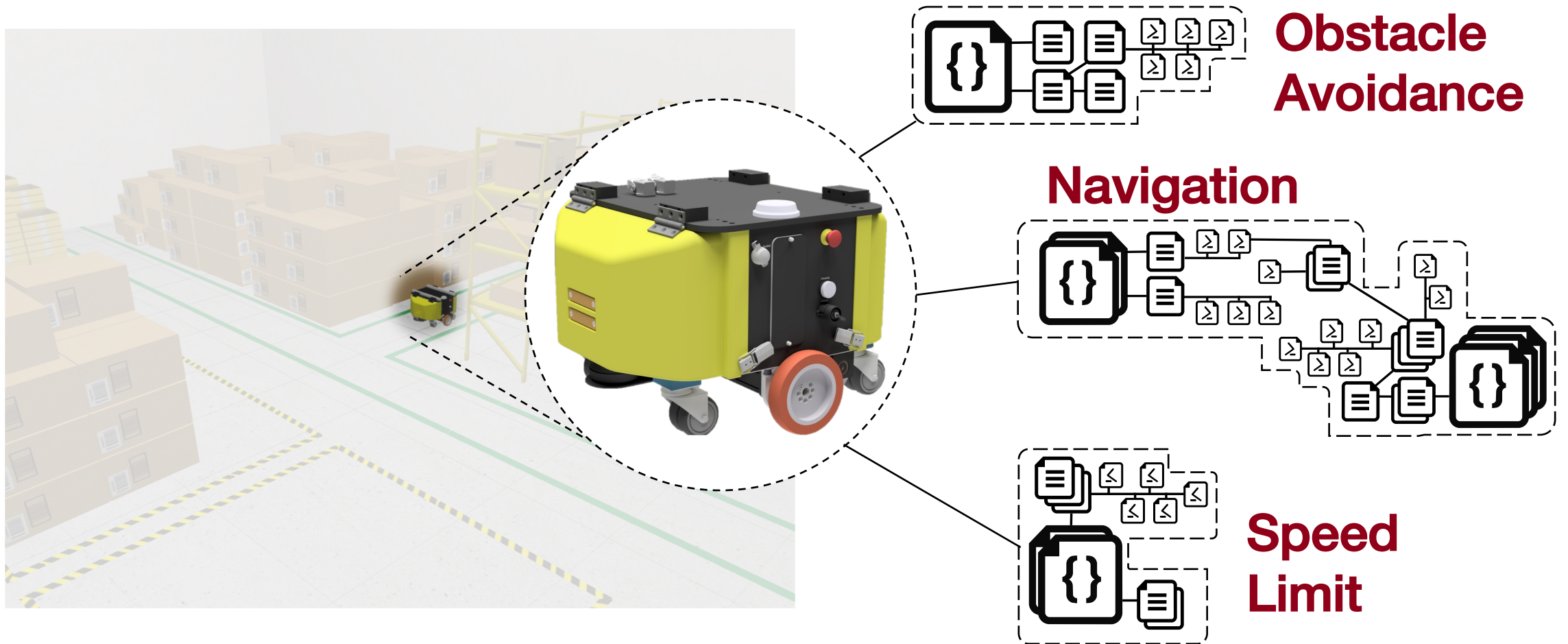
We evaluate ROSpec on a medium-sized robot and a dataset of real-world misconfigurations



Responsible for following a path, avoiding dangerous areas, and respecting speed limits



Responsible for following a path, avoiding dangerous areas, and respecting speed limits



We specified 19 components, using both liquid and dependent types to refine configurations

```
Component: amcl

node type amcl {
  context distribution: AfterHumbleVersion;

  optional param max_beams: int = 60;
  optional param max_particles: int = 2000;
  optional param min_particles: int = 500;
  optional param laser_max_range: double where {_ >= 0} = 100.0;
  optional param set_initial_pose: bool = false;

  optional param initialpose: geometry_msgs/Pose2D = geometry_msgs/Pose2D {
    x = 0.0,
    y = 0.0,
    theta = 0.0,
  };

  @qos{sensor_data_profile}
  subscribes to content(scan_topic): RestrictedLaserScan;

  @qos{sensor_data_profile}
  publishes to particle_cloud: nav2_msgs/ParticleCloud;

  provides service request_nomotion_srv: std_srvs/Empty;
} where {
  min_particles < max_particles;
  set_initial_pose -> exists(initial_pose);
}
```

We specified 19 components, using both liquid and dependent types to refine configurations

```
Component: amcl

node type amcl {
  context distribution: AfterHumbleVersion;

  optional param max_beams: int = 60;
  optional param max_particles: int = 2000;
  optional param min_particles: int = 500;
  optional param laser_max_range: double where {_ >= 0} = 100.0;
  optional param set_initial_pose: bool = false;

  optional param initialpose: geometry_msgs/Pose2D = geometry_msgs/Pose2D {
    x = 0.0,
    y = 0.0,
    theta = 0.0,
  };

  @qos{sensor_data_profile}
  subscribes to content(scan_topic): RestrictedLaserScan;

  @qos{sensor_data_profile}
  publishes to particle_cloud: nav2_msgs/ParticleCloud;

  provides service request_nomotion_srv: std_srvs/Empty;
} where {
  min_particles < max_particles;
  set_initial_pose -> exists(initialpose);
}
```

— Total: 498 lines

✓ Liquid types

✓ QoS policies

✓ Dependencies

We evaluate ROSpec on a medium-sized robot and a dataset of real-world misconfigurations



↑ 0 ↓ rgbdsлам_v2 not receiving any video stream from Realsense R200

RGBDSLAM rgbdsлам_v2 rgbdsламv2 realsense realsense_camera

I'm trying to use a Realsense R200 camera to generate 3-D maps. For this purpose, I have the realsense_r200_nodelet running on \$ROS_MASTER_URI like so

asked Apr 8 '16
archit
3 * 1 = 2 * 3

updated Apr 8 '16

```
roslaunch realsense_camera realsense_r200_nodelet_standalone_manual.launch color_fps:=30  
color_height:=480 color_width:=640 depth_fps:=30 depth_height:=480 depth_width:=640
```

I have rgbdsлам running on a separate machine. I run it like so

```
roslaunch rgbdsлам rgbdsлам.launch
```

Unfortunately, I don't see any video being captured on the GUI that comes up. I have updated the rgbdsлам.launch file to have the right values for the config parameters. The relevant ones are as shown below

```
<param name="config/topic_image_mono" value="/camera/color/image_raw"/>  
<param name="config/topic_image_depth" value="/camera/depth/image_raw"/>  
<param name="config/topic_points" value="/camera/depth/points"/>  
<param name="config/base_frame_name" value="/camera_link"/>  
<param name="config/camera_info_topic" value="/camera/color/camera_info"/>
```

↑ 0 ↓ Set config/topic_points to "". As specified in the launch file : "if empty, poincloud will be reconstructed from image and depth"

answered Apr 10 '16
all-dev
873 * 7 = 14 * 20

add a comment

link

We evaluate ROSpec on a medium-sized robot and a dataset of real-world misconfigurations

0 ^ v

rgbdslam_v2 not receiving any video stream from Realsense R200

RGBDSLAM rgbdslam_v2 rgbdslamv2 realsense realsense_camera

I'm trying to use a Realsense R200 camera to generate 3-D maps. For this purpose, I have the realsense_r200_nodelet running on \$ROS_MASTER_URI like so

asked Apr 8 '16
archit
3 * 1 = 2 * 3

updated Apr 8 '16

```
roslaunch realsense_camera realsense_r200_nodelet_standalone_manual.launch color_fps:=30 color_height:=480 color_width:=640 depth_fps:=30 depth_height:=480 depth_width:=640
```

I have rgbdslam running on a separate machine. I run it like so

```
roslaunch rgbdslam rgbdslam.launch
```

Unfortunately, I don't see any video being captured on the GUI that comes up. I have updated the rgbdslam.launch file to have the right values for the config parameters. The relevant ones are as shown below

```
<param name="config/topic_image_mono" value="/camera/color/image_raw"/>
<param name="config/topic_image_depth" value="/camera/depth/image_raw"/>
<param name="config/topic_points" value="/camera/depth/points"/>
<param name="config/base_frame_name" value="/camera_link"/>
<param name="config/camera_info_topic" value="/camera/color/camera_info"/>
```

0 ^ v

Set config/topic_points to "". As specified in the launch file : "if empty, poincloud will be reconstructed from image and depth"

answered Apr 10 '16
all-dev
873 * 7 = 14 * 20

add a comment link

We manually inspect 182 questions from a Q&A platform, and specify components & systems

 0 

rgbdslam_v2 not receiving any video stream from Realsense R200

RGBDSLAM

rgbdslam_v2

rgbdslamv2

realsense

realsense_camera

I'm trying to use a Realsense R200 camera to generate 3-D maps. For this purpose, I have the realsense_r200_nodelet running on \$ROS_MASTER_URI like so

asked Apr 8 '16

 archit
3 • 1 • 2 • 3

updated Apr 8 '16

```
roslaunch realsense_camera realsense_r200_nodelet_standalone_manual.launch color_fps:=30  
color_height:=480 color_width:=640 depth_fps:=30 depth_height:=480 depth_width:=640
```

I have rgbdslam running on a separate machine. I run it like so

```
roslaunch rgbdslam rgbdslam.launch
```

Unfortunately, I don't see any video being captured on the GUI that comes up. I have updated the rgbdslam.launch file to have the right values for the config parameters. The relevant ones are as shown below

```
<param name="config/topic_image_mono" value="/camera/color/image_raw"/>  
<param name="config/topic_image_depth" value="/camera/depth/image_raw"/>  
<param name="config/topic_points" value="/camera/depth/points"/>  
<param name="config/base_frame_name" value="/camera_link"/>  
<param name="config/camera_info_topic" value="/camera/color/camera_info"/>
```

 0 



Set `config/topic_points` to `""`. As specified in the launch file : "if empty, poincloud will be reconstructed from image and depth"

answered Apr 10 '16

 al-dev
873 • 7 • 14 • 20

 add a comment

 link

We manually inspect 182 questions from a Q&A platform, and specify components & systems

rgbdsлам_v2 not receiving any video stream from Realsense R200

RGBDSLAM rgbdsлам_v2 rgbdsламv2 realsense realsense_camera

I'm trying to use a Realsense R200 camera to generate 3-D maps. For this purpose, I have the realsense_r200_nodelet running on \$ROS_MASTER_URI like so

asked Apr 8 '16

3 +1 -2 +3

updated Apr 8 '16

```
roslaunch realsense_camera realsense_r200_nodelet_standalone_manual.launch color_fps:=30  
color_height:=480 color_width:=640 depth_fps:=30 depth_height:=480 depth_width:=640
```

I have rgbdsлам running on a separate machine. I run it like so

```
roslaunch rgbdsлам rgbdsлам.launch
```

Unfortunately, I don't see any video being captured on the GUI that comes up. I have updated the rgbdsлам.launch file to have the right values for the config parameters. The relevant ones are as shown below

```
<param name="config/topic_image_mono" value="/camera/color/image_raw"/>  
<param name="config/topic_image_depth" value="/camera/depth/image_raw"/>  
<param name="config/topic_points" value="/camera/depth/points"/>  
<param name="config/base_frame_name" value="/camera_link"/>  
<param name="config/camera_info_topic" value="/camera/color/camera_info"/>
```

Set config/topic_points to "". As specified in the launch file : "if empty, poincloud will be reconstructed from image and depth"

answered Apr 10 '16

al-dev 873 7 +14 20

add a comment

----- What packages are used?

----- What components are used?

----- What configurations are used?

----- What is the fix?



We use components documentation, and our playground to create and refine specifications



NAV 2

robot_localization 2.6.12 documentation »

previous | next | index

Writing a New Controller Plugin

- Overview
- Requirements
- Tutorial Steps

navsat_transform_node

navsat_transform_node takes as input a nav_msgs/Odometry message (usually the output of ekf_localization_node or ukf_localization_node), a sensor_msgs/Imu containing an accurate estimate of your robot's heading, and a sensor_msgs/NavSatFix message containing GPS data. It produces an odometry message in coordinates that are consistent with your robot's world frame. This value can be directly fused into your state estimate.

Note: If you fuse the output of this node with any of the state estimation nodes in robot_localization, you should make sure that the odom_differential setting is false for that input.

Parameters

~frequency

The real-valued frequency, in Hz, at which navsat_transform_node checks for new sensor_msgs/NavSatFix messages, and publishes filtered sensor_msgs/NavSatFix when publish_filtered_gps is set to true.

~delay

The time, in seconds, to wait before calculating the transform from GPS coordinates to your robot's world frame.

~magnetic_declination_radians

Enter the magnetic declination for your location. If you don't know it, see <http://www.ngdc.noaa.gov/geomag-web> (make sure to convert the value to radians). This parameter is needed if your IMU provides its orientation with respect to the magnetic north.

~yaw_offset

Table Of Contents

- State Estimation Nodes
 - navsat_transform_node
 - Parameters
 - ~frequency
 - ~delay
 - ~magnetic_declination_radians
 - ~yaw_offset
 - ~zero_altitude
 - ~publish_filtered_gps
 - ~broadcast_utm_transform
 - ~use_odometry_yaw
 - ~wait_for_datum
 - ~broadcast_utm_transform_as_parent_frame
 - ~transform_timeout
- Subscribed Topics
- Published Topics
- Published Transforms

Preparing Your Data for Use with robot_localization

Configuring robot_localization

Preparing from robot_pose_4d

Preparing from robot_pose_4d



rospec

- Home
- Getting Started
- Language
- Examples
- Evaluation
- Playground
- About us

Search rospect

rospec - A Domain-Specific Language for ROS-based Robot Software

Ensuring the correct configuration and integration of your ROS components.

[Get Started](#) [View on GitHub](#)

What is rospect?

rospect is a domain-specific language designed to specify and verify component configurations and their integration in ROS-based robot software. With rospect, you can:

- Define component specifications with explicit parameter types, constraints, and dependencies;
- Verify system integrations to ensure components are correctly configured and connected;
- Detect misconfigurations early before they lead to runtime failures or dangerous behaviors;
- Document component requirements for other developers in a formal, verifiable way.

Quick Example

Here's a simple example of a rospect specification for a camera node and its configuration:

```
node type camera_node_type {
  param frame_rate: double where {_. > 0.0 and _ <= 30.0};
  publishes to camera/image_raw: sensor_msgs/Image;
}

system {
  node instance main_camera: camera_node_type {
    param frame_rate = 15.0; # Valid configuration
  }
}
```

In this example, rospect ensures that the camera's frame rate is within the allowed range of [0, 30] frames per second, helping prevent performance issues or hardware damage from invalid configurations.

Why rospect?

rospect.pcanelas.com

ROSpec documents components and detects real misconfigurations made by developers



61

Detectable

Component specification possible



23

Documentation

Missing integration information



31

Not Supported

Cannot model the concept (e.g., URDF)

32% use of
Dependent types

17% explicit use of
Liquid types

ROSpec documents components and detects real misconfigurations made by developers



28

Not enough context
Missing information

Out of Scope

How-to use, bug in component

31



61

Detectable

Component specification possible



23

Documentation

Missing integration information



31

Not Supported

Cannot model the concept (e.g., URDF)

32%

use of
Dependent types

17%

explicit use of
Liquid types

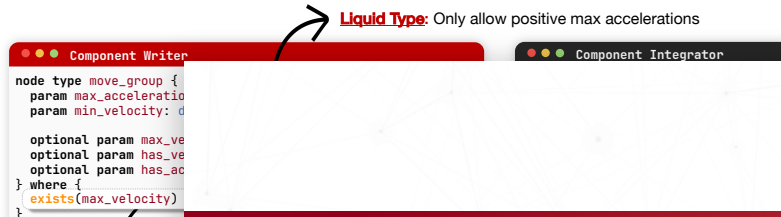
Find more on our paper, website or meet me during the coffee break! And...

ROSpec: A Domain-Specific Language for ROS-based Robot Software



Paulo Canelas
Carnegie Mellon University
University of Lisbon

We enforce type restrictions and parameter dependencies using liquid and dependent types



By specifying **{properties}** over **component configurations** and their **integration**, we can **detect misconfigurations** prior to deployment

ROSpec: A Domain-Specific Language for ROS-Based Robot Software

PAULO CANELAS, Carnegie Mellon University, USA and University of Lisbon, Portugal

BRADLEY SCHMERL, Carnegie Mellon University, USA

ALCIDES FONSECA, University of Lisbon, Portugal

CHRISTOPHER S. TIMPERLEY, Carnegie Mellon University, USA

Component-based robot software frameworks, such as the Robot Operating System (ROS), allow developers to quickly compose and execute systems by focusing on configuring and integrating reusable, off-the-shelf components. However, these components often lack documentation on how to configure and integrate them correctly. Even when documentation exists, its natural language specifications are not enforced, resulting in misconfigurations that lead to unpredictable and potentially dangerous robot behaviors. In this work, we introduce ROSpec, a ROS-tailored domain-specific language designed to specify and verify component configurations and their integration. ROSpec's design is grounded in ROS domain concepts and informed by a prior empirical study on misconfigurations, allowing the language to provide a usable and expressive way of specifying and detecting misconfigurations. At a high level, ROSpec verifies the correctness of argument and

We would like to thank Catarina Gamboa, Claire Le Goues, and Jonathan Aldrich for their feedback on this work.

This work was supported by Fundação para a Ciência e Tecnologia (FCT) and FEDER in the LASIGE Research Unit under the ref. (UID/00408/2025), RAP(EXPL/CCI-COM/1306/2021), RAP+ (LISBOA2030-FEDER-00663700), HPC (2025.00002.HPCVLAB.ISTUL, POR011PRE), the CMU Portugal Dual Degree PhD program (SFRH/BD/151469/2021), and a CMU Cylab seed grant.

Find more on our paper, website or meet me during the coffee break! And...

paulo where {**pau**lo.job_market == **True**}



pcanelas.com



pasantos@andrew.cmu.edu

Empirical Research

Program Analysis

Software Engineering

Programming Languages

We would like to thank Catarina Gamboa, Claire Le Goues, and Jonathan Aldrich for their feedback on this work.

This work was supported by Fundação para a Ciência e Tecnologia (FCT) and FEDER in the LASIGE Research Unit under the ref. (UIDB/00408/2025), RAP/EXPL/CCI-COM/1306/2021, RAP/498904/2030-FEDER-00663700, HPC (2025.00002.HPCVLAB.ISTUL, POR011PRQ), the CMU Portugal Dual Degree PhD program (SFRH/BD/151463/2021), and a CMU Cylab seed grant.