Processing Camera Data

ROS + PR2 Training Workshop

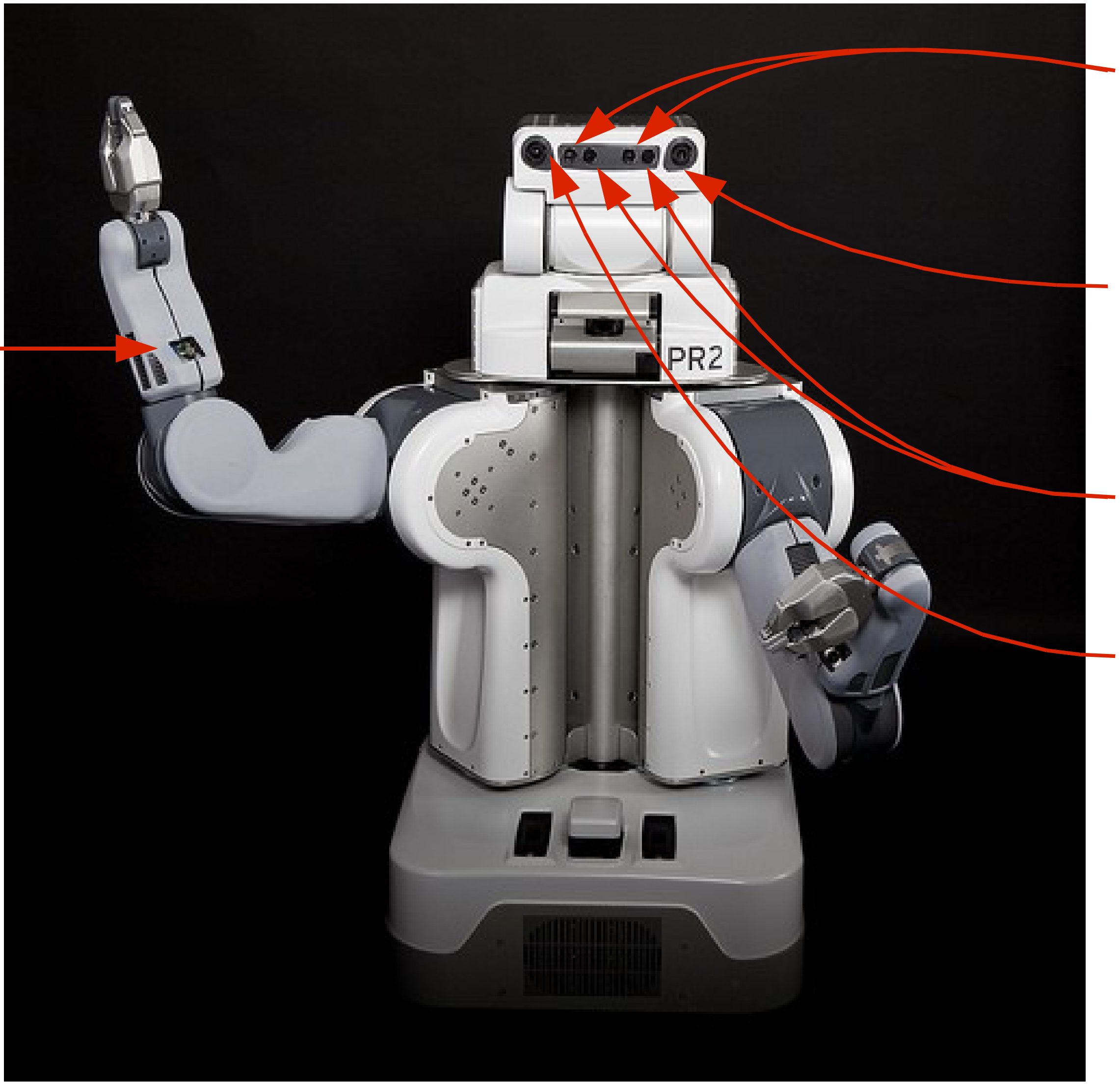
Outline

- Cameras on the PR2
- The monocular image pipeline
- The stereo image pipeline
- Logging sensor data
- Writing a vision node

Outline

- Cameras on the PR2
  - The camera suite
  - Viewing images
  - Adjusting camera parameters
  - Using the texture projector
  - Saving bandwidth
- The monocular image pipeline
- The stereo image pipeline
- Logging sensor data
- Writing a vision node

Cameras on the PR2



Viewing Images

- rviz node
  - Displays panel → Add → Camera

- image_view node
  - `$ rosrun image_view image_view image:=<camera>/<image>`

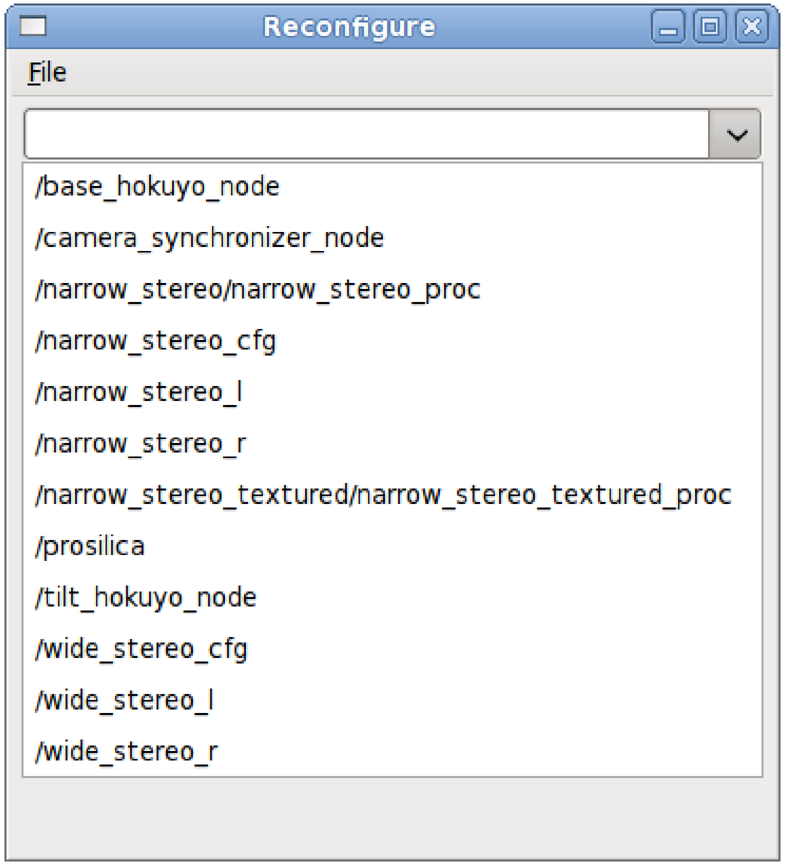
www.ros.org/wiki/rviz/DisplayTypes/Camera
www.ros.org/wiki/image_view

Configuring Your Cameras

```
$ rosrun dynamic_reconfigure reconfigure_gui
```

For cameras:

- Adjust camera parameters (exposure, gain, ...)
- Turn texture projector on/off
- Adjust stereo processing parameters



/base_hokuyo_node
/camera_synchronizer_node
/narrow_stereo/narrow_stereo_proc
/narrow_stereo_cfg
/narrow_stereo_l
/narrow_stereo_r
/narrow_stereo_textured/narrow_stereo_textured_proc
/prosilica
/tilt_hokuyo_node
/wide_stereo_cfg
/wide_stereo_l
/wide_stereo_r

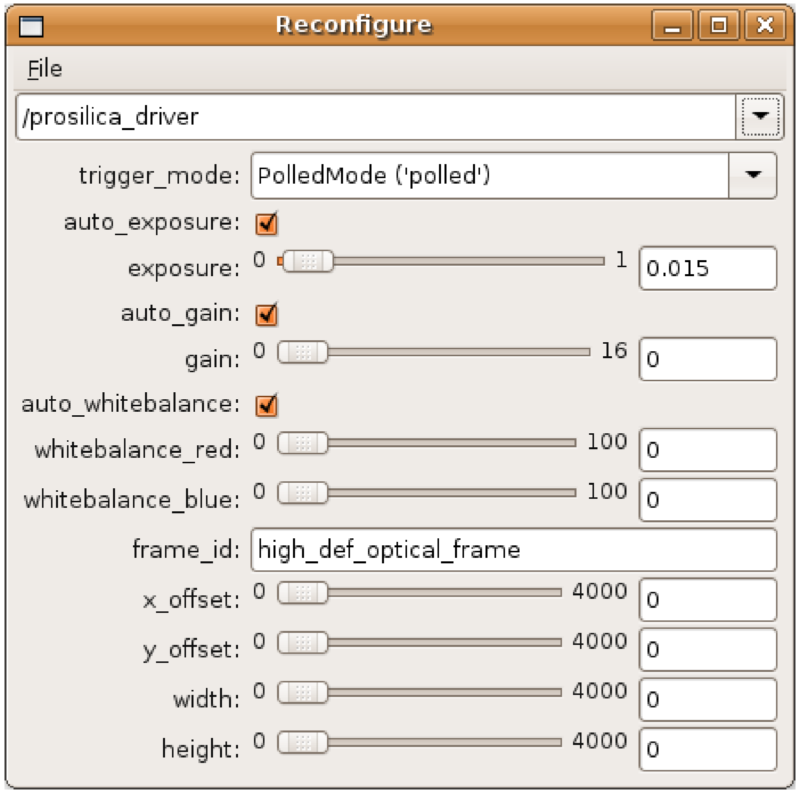
www.ros.org/wiki/dynamic_reconfigure

Adjust Camera Parameters

Cameras:

- /l_forearm_cam
- /r_forearm_cam
- /wide_stereo_both
- /narrow_stereo_both
- /prosilica_driver

For stereo cameras, "both" propagates settings to left & right

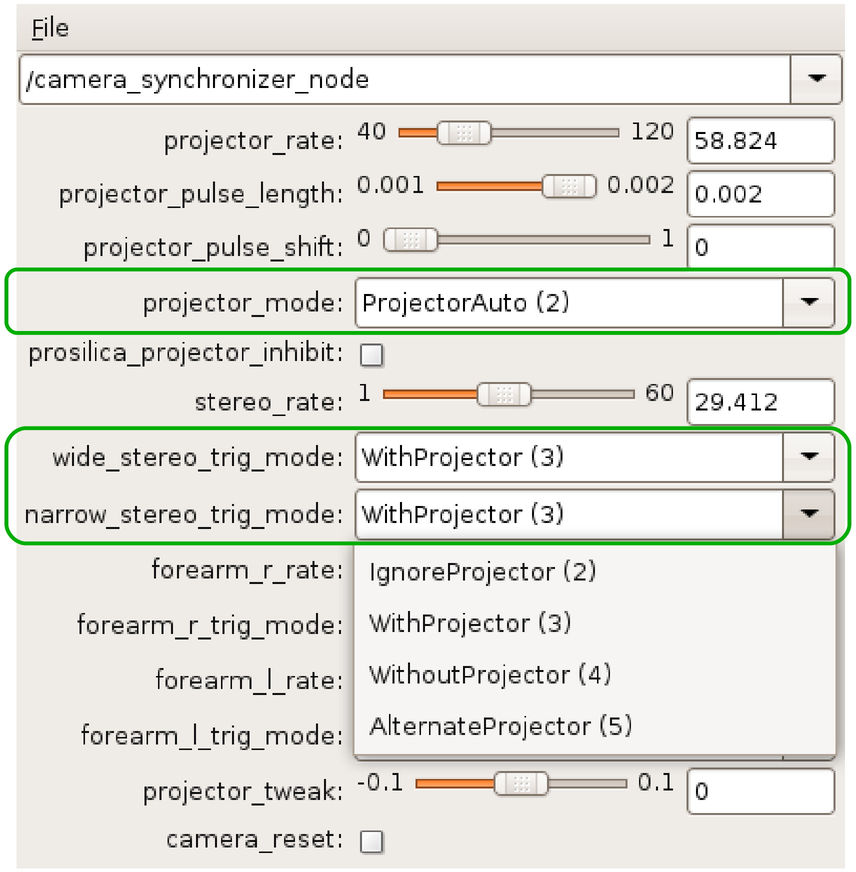


/prosilica_driver

trigger_mode:	PolledMode ('polled')		
auto_exposure:	✓		
exposure:	0	1	0.015
auto_gain:	✓		
gain:	0	16	0
auto_whitebalance:	✓		
whitebalance_red:	0	100	0
whitebalance_blue:	0	100	0
frame_id:	high_def_optical_frame		
x_offset:	0	4000	0
y_offset:	0	4000	0
width:	0	4000	0
height:	0	4000	0

Using the Texture Projector

- projector_mode - whether projector is turned on
- *_trig_mode - whether the camera synchs with the projector on all, no, or some frames
- Camera namespaces change when using the texture projector:
  - /narrow_stereo
  - /narrow_stereo_textured


File
/camera_synchronizer_node

projector_rate:	40		120	58.824
projector_pulse_length:	0.001		0.002	0.002
projector_pulse_shift:	0		1	0
projector_mode:	ProjectorAuto (2)			
prosilica_projector_inhibit:	☐			
stereo_rate:	1		60	29.412
wide_stereo_trig_mode:	WithProjector (3)			
narrow_stereo_trig_mode:	WithProjector (3)			
forearm_r_rate:				
forearm_r_trig_mode:				
forearm_l_rate:				
forearm_l_trig_mode:				
projector_tweak:	-0.1		0.1	0
camera_reset:	☐			

Saving Bandwidth

- Each image topic has alternate compressed versions
- Transports available out of the box
  - “raw” – default, uncompressed
  - “compressed” – JPEG or PNG
  - “theora” – Theora video codec
- With image_view:

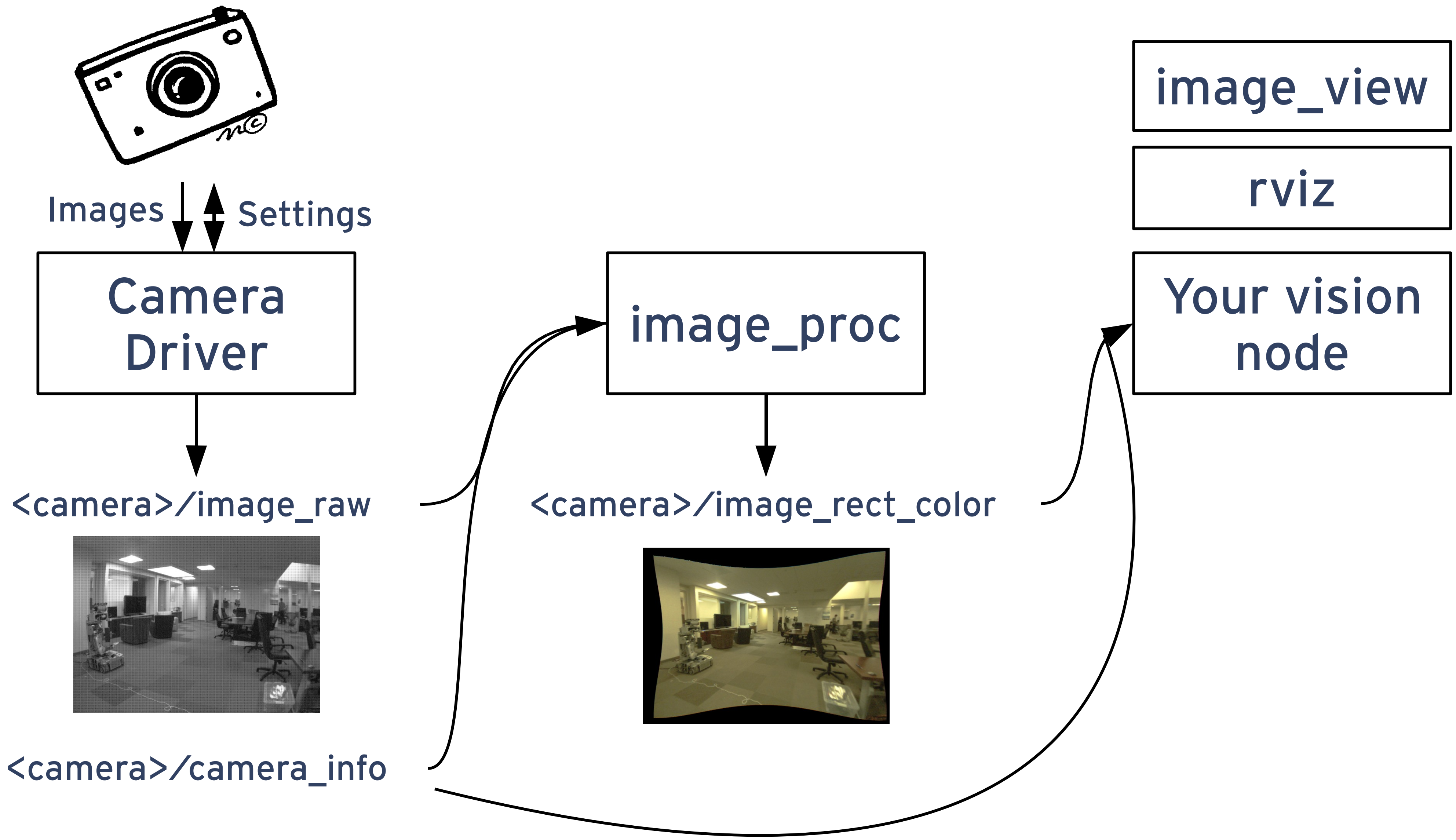
```
$ rosrun image_view image_view
image:=<camera>/<image> compressed
```

www.ros.org/wiki/image_transport

Outline

- ✓ Cameras on the PR2
- The monocular image pipeline
  - Camera calibration
  - Basic processing
    - De–Bayering
    - Rectification
- The stereo image pipeline
- Logging sensor data
- Writing a vision node

The Monocular Image Pipeline



www.ros.org/wiki/camera_drivers, www.ros.org/wiki/image_pipeline

Mono Camera Calibration

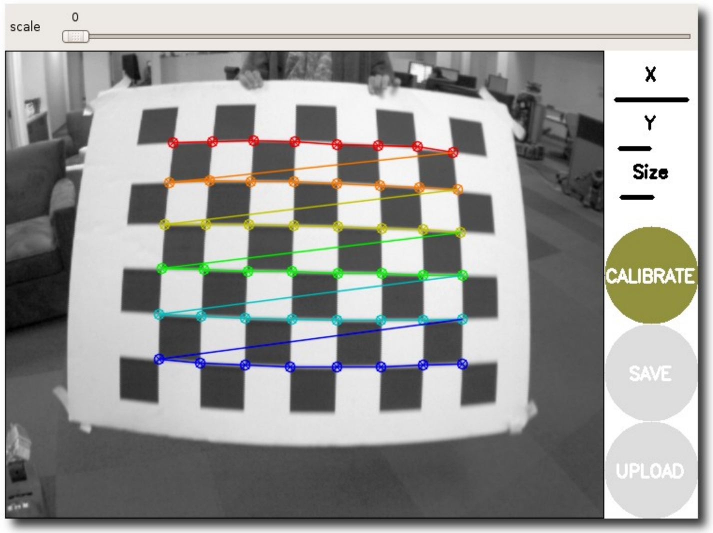
Distortion

Before
After

- Parameters → camera driver
- CameraInfo message published with each Image

www.ros.org/wiki/camera_calibration

Mono Camera Calibration



- Get a large checkerboard

- `$ rosrun camera_calibration cameracalibrator.py --size 8x6 --square 0.108 image:=/<camera>/image_raw camera:=/<camera>`

www.ros.org/wiki/camera_calibration/Tutorials/MonocularCalibration

Mono Processing

image_proc publishes image topics that are

- De-bayered (grayscale or color)
- Rectified

On topics:

<camera>/image_mono

<camera>/image_color

<camera>/image_rect

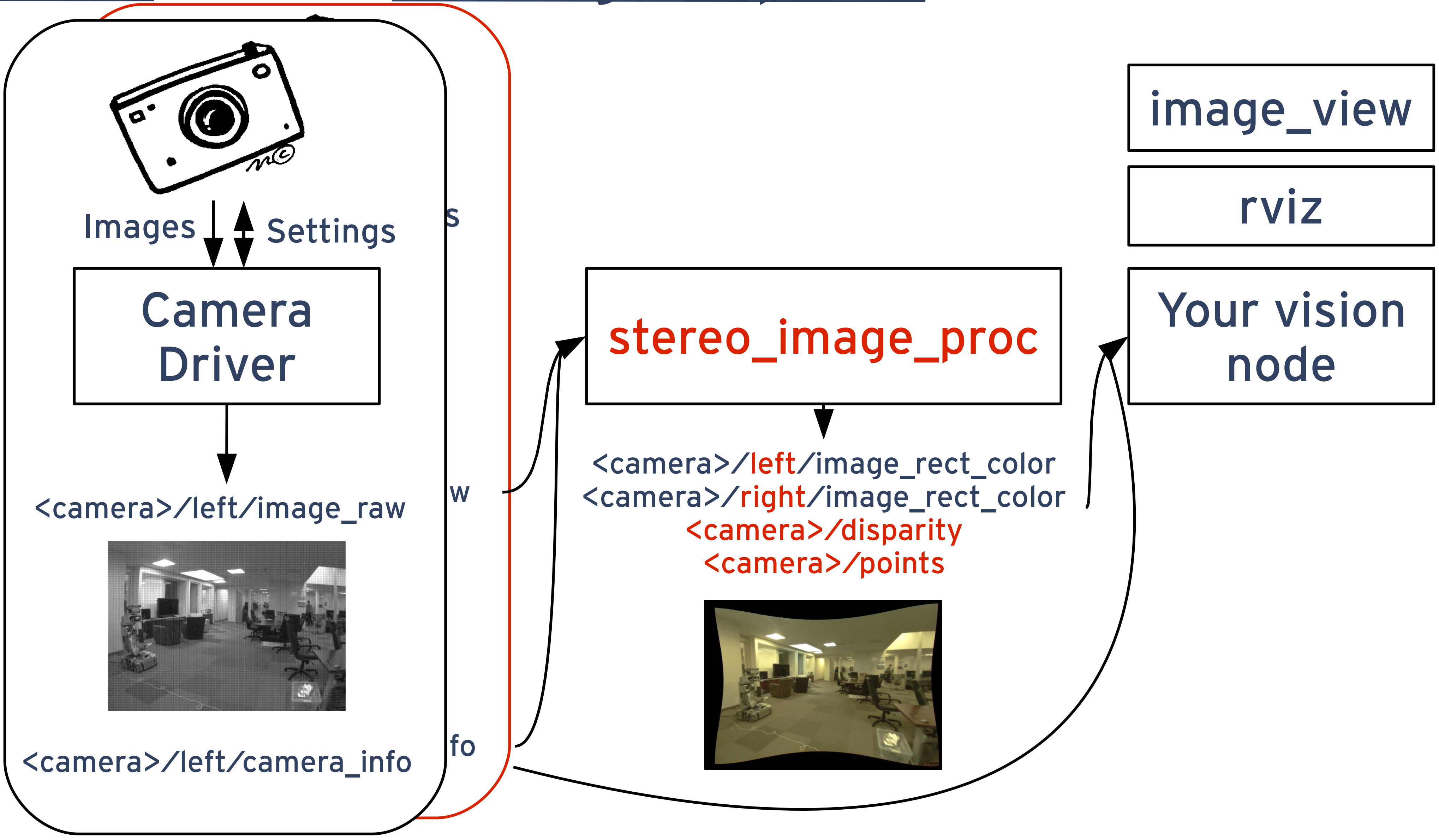
<camera>/image_rect_color

www.ros.org/wiki/image_proc

Outline

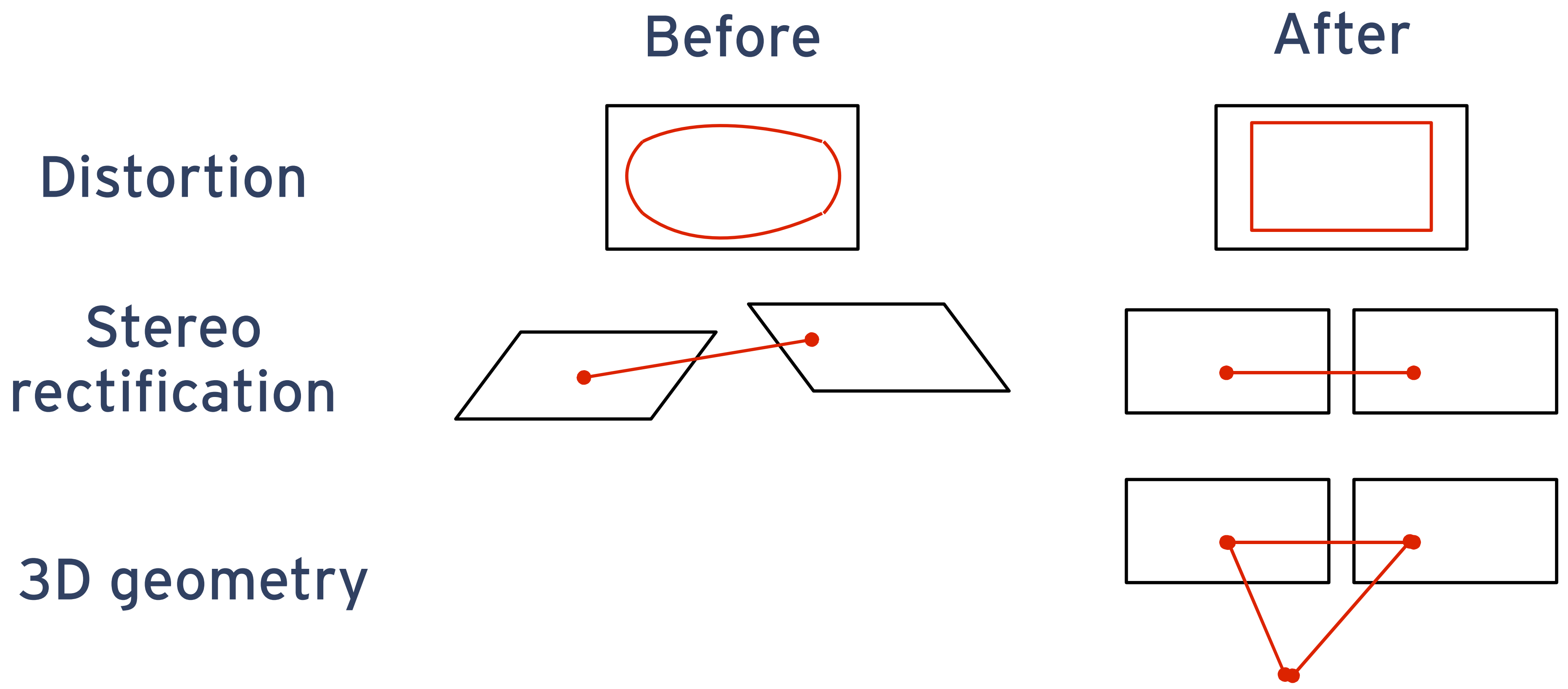
- ✓ Cameras on the PR2
- ✓ The monocular image pipeline
- The stereo image pipeline
  - Stereo calibration
  - Stereo processing (3D)
  - Viewing disparity images and point clouds
  - Adjusting stereo parameters
- Logging sensor data
- Writing a vision node

The Stereo Image Pipeline



www.ros.org/wiki/camera_drivers, www.ros.org/wiki/image_pipeline

Stereo Camera Calibration



- Parameters → camera drivers
- CameraInfo message published with each Image

www.ros.org/wiki/camera_calibration

Stereo Processing

stereo_image_proc publishes

- Disparity images
- Point clouds

On topics:

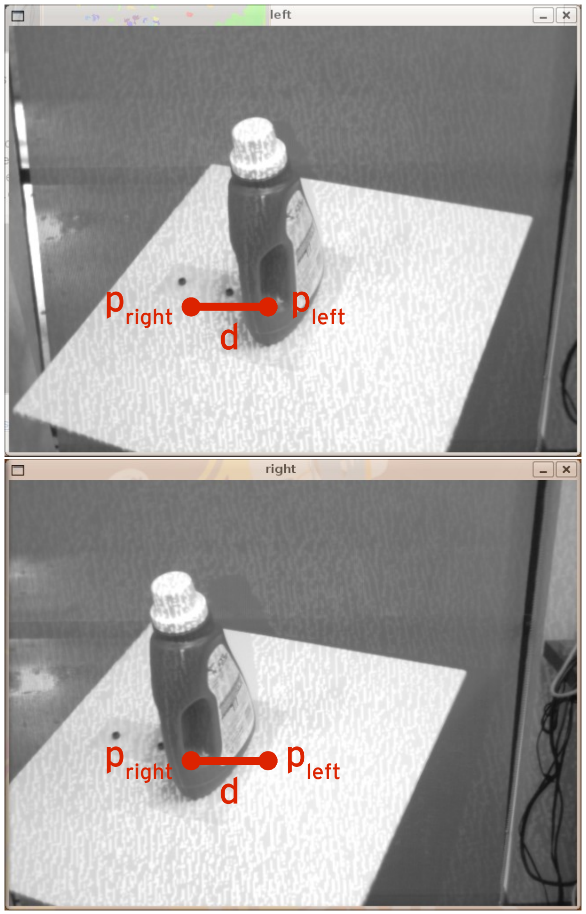
<stereo_camera>/disparity

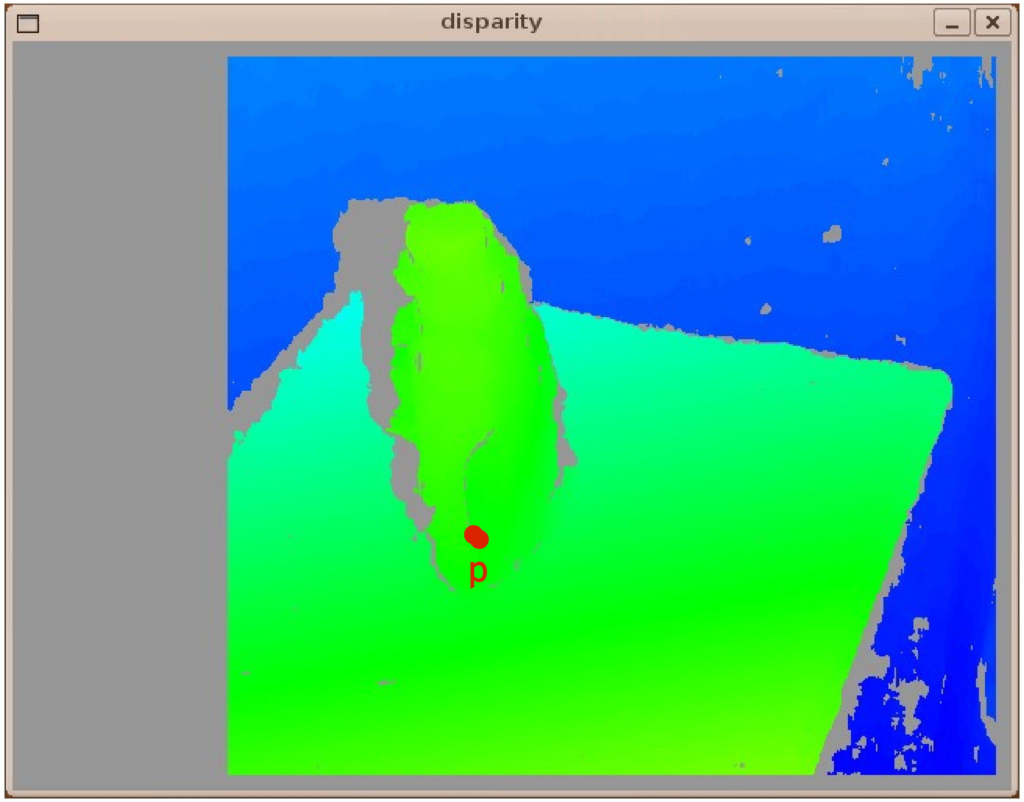
<stereo_camera>/points

www.ros.org/wiki/stereo_image_proc

Viewing Disparity Images

```
$ rosrun image_view stereo_view
    stereo:=narrow_stereo image:=image_rect
```

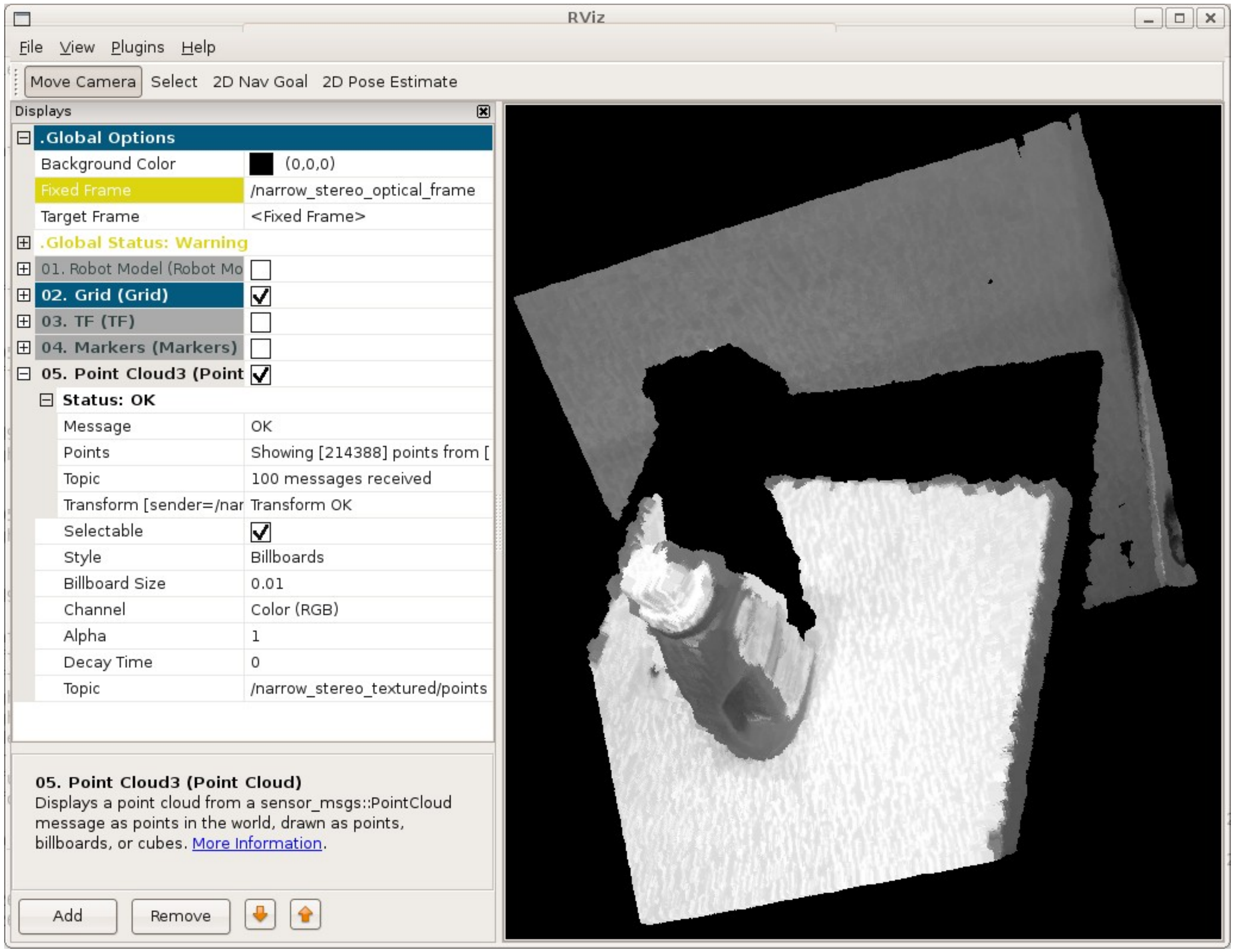




Viewing Point Clouds

Displays panel → Add → Point Cloud

Enter the topic in the red box

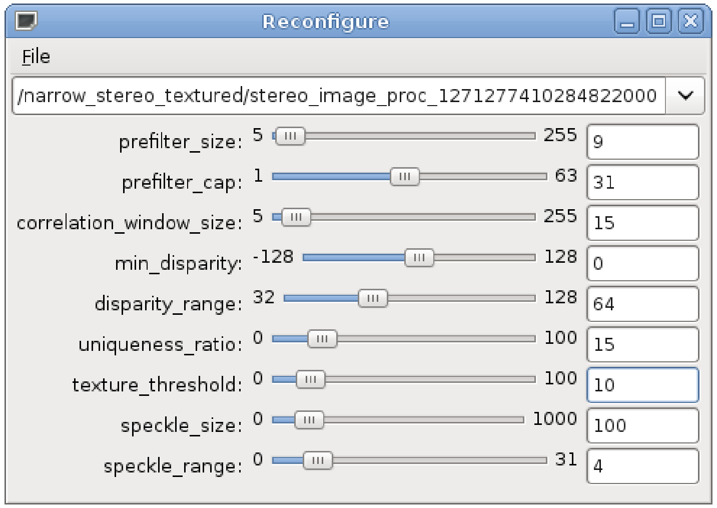


.Global Options	
Background Color	(0,0,0)
Fixed Frame	/narrow_stereo_optical_frame
Target Frame	<Fixed Frame>
.Global Status: Warning	
01. Robot Model (Robot Mo	☐
02. Grid (Grid)	✓
03. TF (TF)	☐
04. Markers (Markers)	☐
05. Point Cloud3 (Point	✓
Status: OK	
Message	OK
Points	Showing [214388] points from [
Topic	100 messages received
Transform [sender=/nar	Transform OK
Selectable	✓
Style	Billboards
Billboard Size	0.01
Channel	Color (RGB)
Alpha	1
Decay Time	0
Topic	/narrow_stereo_textured/points

05. Point Cloud3 (Point Cloud)
Displays a point cloud from a sensor_msgs::PointCloud message as points in the world, drawn as points, billboards, or cubes. More Information.

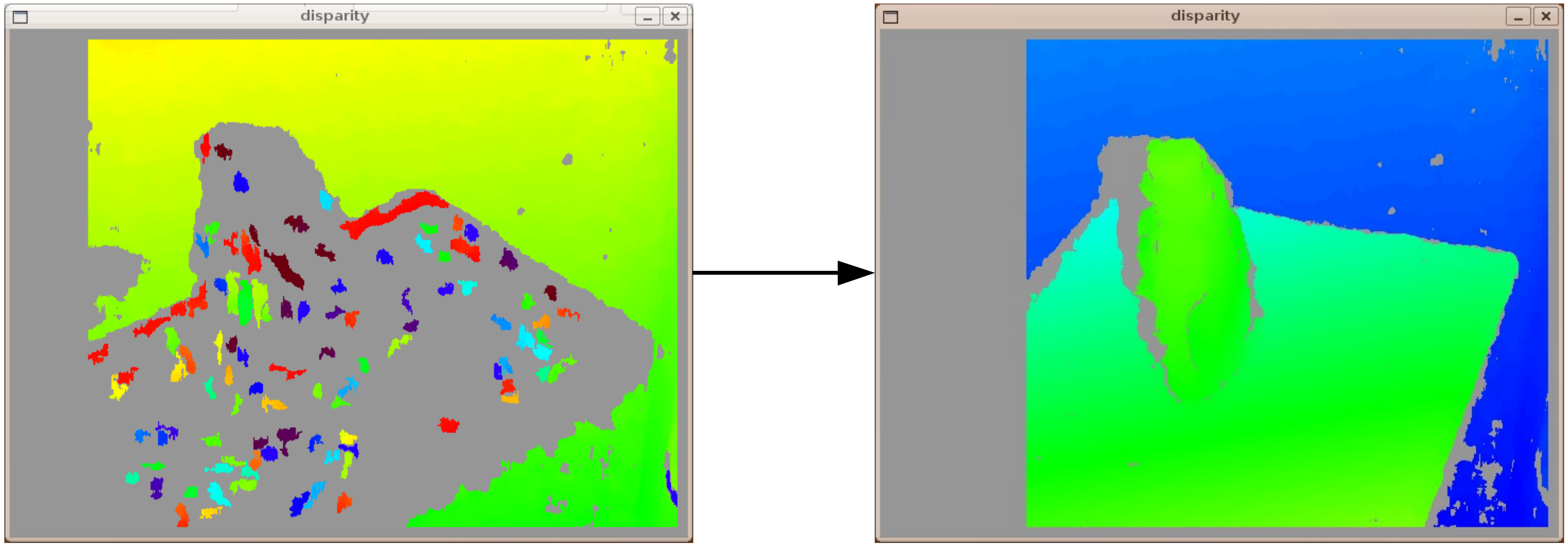
Adjust Stereo Parameters

In reconfigure_gui, look for <stereo>/stereo_image_proc



/narrow_stereo_textured/stereo_image_proc_1271277410284822000

prefilter_size:	5	255	9
prefilter_cap:	1	63	31
correlation_window_size:	5	255	15
min_disparity:	-128	128	0
disparity_range:	32	128	64
uniqueness_ratio:	0	100	15
texture_threshold:	0	100	10
speckle_size:	0	1000	100
speckle_range:	0	31	4



www.ros.org/wiki/stereo_image_proc/Tutorials/ChoosingGoodStereoParameters

Exercise #1

Calibrate your narrow stereo cameras.

www.ros.org/wiki/camera_calibration/Tutorials/StereoCalibration

But do not "Commit" to the camera, or you will have to run full-body calibration again!

Outline

- ✓ Cameras on the PR2
- ✓ The monocular image pipeline
- ✓ The stereo image pipeline
- Logging sensor data
  - Recording and playback
  - Visual inspection with rxbag
  - Bags and ROS time
- Writing a vision node

Logging Sensor Data

Recording data:

```
$ rosbag record r_forearm_cam/image_raw
r_forearm_cam/camera_info tf
```

Play back data:

```
$ rosbag play XXX.bag
```

What's in a bag file:

```
$ rosbag info mystery_data.bag
```

www.ros.org/wiki/rosbag/Tutorials/Recording and playing back data

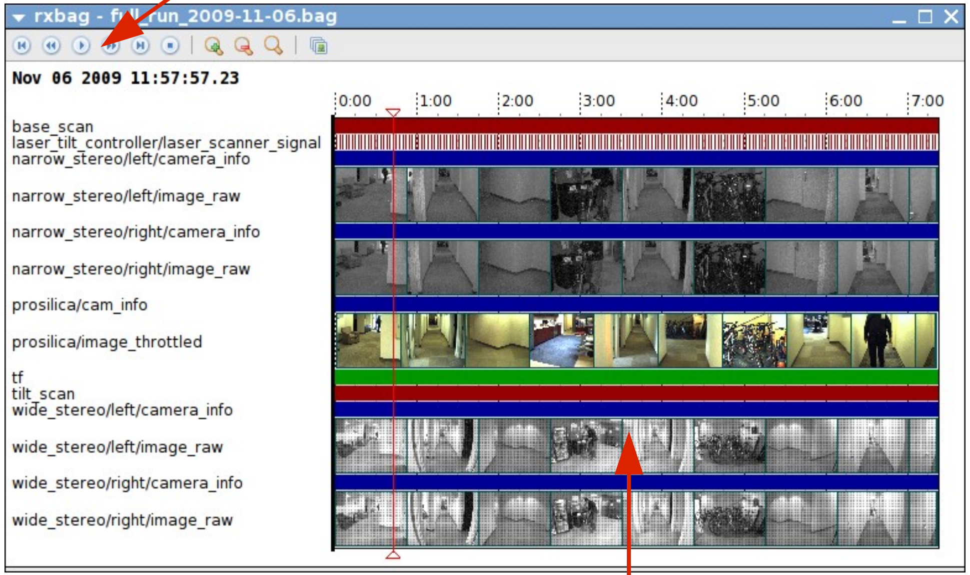
Visual Inspection with rxbag

```
$ rxbag XXX.bag
```

View thumbnails:
Right–click → Thumbnails...
  → select topic(s)

Image viewer for a topic:
Right–click
  → View (by datatype)...
  → sensor_msgs/Image
  → <topic> → Image

Playback


rxbag - full_run_2009-11-06.bag
Nov 06 2009 11:57:57.23
0:00
1:00
2:00
3:00
4:00
5:00
6:00
7:00
base_scan
laser_tilt_controller/laser_scanner_signal
narrow_stereo/left/camera_info
narrow_stereo/left/image_raw
narrow_stereo/right/camera_info
narrow_stereo/right/image_raw
prosilica/cam_info
prosilica/image_throttled
tf
tilt_scan
wide_stereo/left/camera_info
wide_stereo/left/image_raw
wide_stereo/right/camera_info
wide_stereo/right/image_raw

Click to jump

www.ros.org/wiki/rxbag

Bags and ROS Time

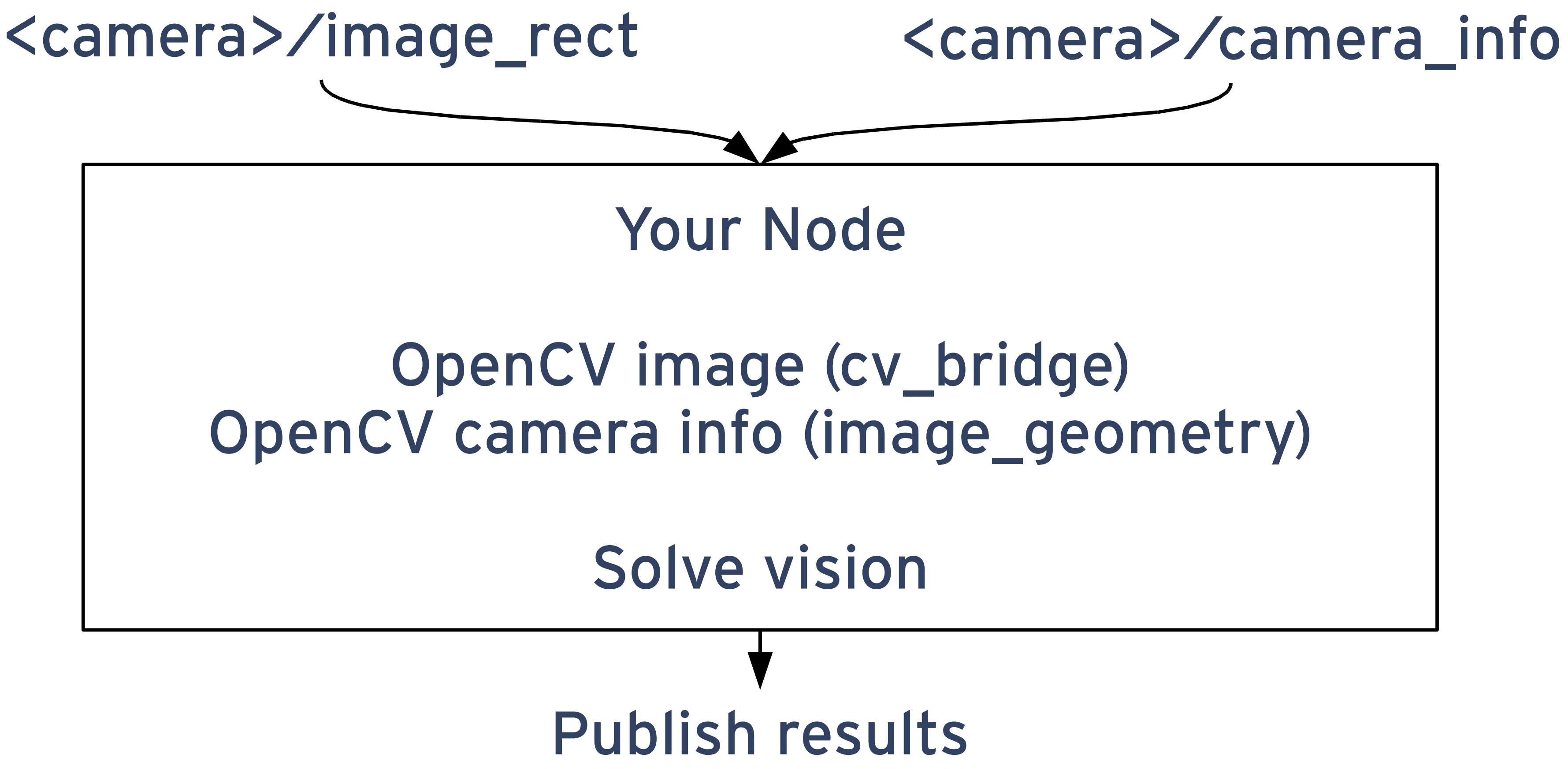
- Do not play back a bag against a live robot!
- Recorded message timestamps will be far in the past relative to “wall-clock” time
- When using time-aware nodes with bagged data:
  - `$ rosbag play --clock XXX.bag`
  - Set parameter /use_sim_time = True before starting nodes
- Do not set /use_sim_time on a live robot!

www.ros.org/wiki/Clock

Outline

- ✓ Cameras on the PR2
- ✓ The monocular image pipeline
- ✓ The stereo image pipeline
- ✓ Logging sensor data
- Writing a vision node
  - Subscribing to camera topics
  - Interfacing with OpenCV
  - Publishing images

Processing Images in ROS



www.ros.org/wiki/cv_bridge
www.ros.org/wiki/image_geometry

Subscribing to Camera Topics

Use image_transport instead of ros::Subscriber

- Compression (JPEG/PNG, Theora)
- Add others as plugins

Subscribe to image AND camera_info topics

- Synchronized time stamps
- CameraSubscriber handles the synchro

www.ros.org/wiki/image_transport

Subscribing to Camera Topics

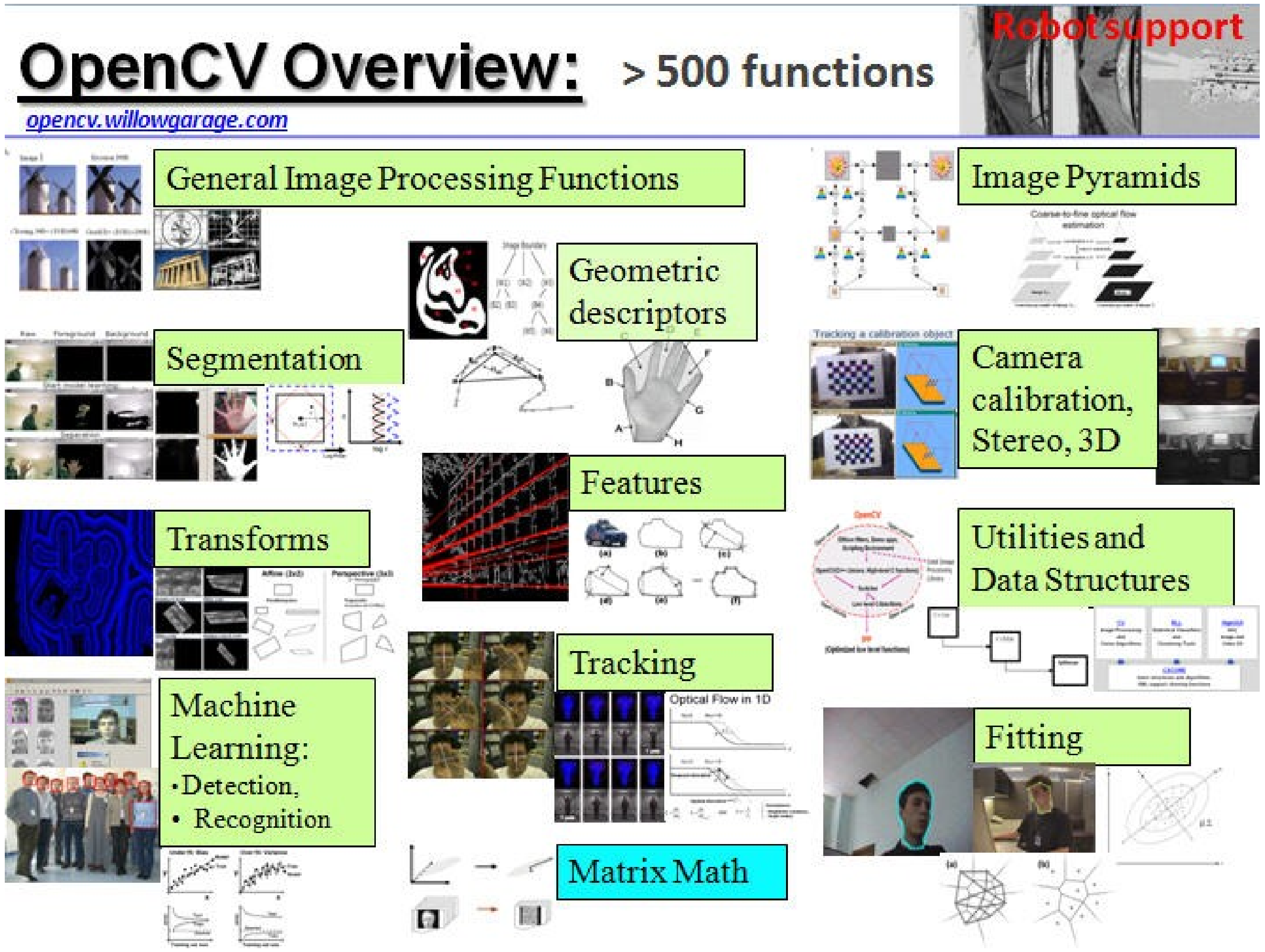
```cpp
#include <ros/ros.h>
#include <image_transport/image_transport.h>

class MyVisionNode
{
  ros::NodeHandle nh_;
  image_transport::ImageTransport it_;
  image_transport::CameraSubscriber sub_;

public:
  MyVisionNode()
    : it_(nh_)
  {
    sub_ = it_.subscribeCamera("image_topic", 1,
&MyVisionNode::imageCb, this);
  }

  void imageCb(const sensor_msgs::ImageConstPtr& image_msg,
               const sensor_msgs::CameraInfoConstPtr& info_msg)
  {
    // ...
  }
};
```

Open Computer Vision Library


OpenCV Overview: > 500 functions

opencv.willowgarage.com

opencv.willowgarage.com

Using ROS messages with OpenCV

cv_bridge

- ROS sensor_msgs/Image → OpenCV IplImage

image_geometry

- ROS sensor_msgs/CameraInfo → OpenCV calibration matrices

Many useful functions in the camera model classes

www.ros.org/wiki/cv_bridge/Tutorials
www.ros.org/wiki/image_geometry

ROS Image -> OpenCV

```cpp
#include <cv_bridge/CvBridge.h>

class MyVisionNode
{
  sensor_msgs::CvBridge bridge_;

public:
  void imageCb(const sensor_msgs::ImageConstPtr& image_msg,
               const sensor_msgs::CameraInfoConstPtr& info_msg)
  {
    IplImage *cv_image = NULL;
    try {
      cv_image = bridge_.imgMsgToCv(image_msg, "bgr8");
    }
    catch (sensor_msgs::CvBridgeException& error) {
      ROS_ERROR("Couldn't convert image with encoding %s",
                image_msg->encoding.c_str());
      return;
    }
  }
};
```

ROS CameraInfo -> OpenCV

```cpp
#include <image_geometry/pinhole_camera_model.h>

class MyVisionNode
{
  image_geometry::PinholeCameraModel cam_model_;

public:
  void imageCb(const sensor_msgs::ImageConstPtr& image_msg,
               const sensor_msgs::CameraInfoConstPtr& info_msg)
  {
    cam_model_.fromCameraInfo(info_msg);
  }
};
```

Publishing Image Topics

```cpp
class MyVisionNode
{
  ros::NodeHandle nh_;
  image_transport::ImageTransport it_;
  image_transport::Publisher pub_;

public:
  MyVisionNode()
    : it_(nh_)
  {
    pub_ = it_.advertise("image_out", 1);
  }

  void imageCb(const sensor_msgs::ImageConstPtr& image_msg,
               const sensor_msgs::CameraInfoConstPtr& info_msg)
  {
    // ...
    pub_.publish(bridge_.cvToImgMsg(image, "bgr8"));
  }
};
```

Outline

- ✓ Cameras on the PR2
- ✓ The monocular image pipeline
- ✓ The stereo image pipeline
- ✓ Logging sensor data
- ✓ Writing a vision node

Exercise #2

Draw the location of the robot gripper (according to tf) on an image stream.

http://www.ros.org/wiki/image_geometry/Tutorials/ProjectTfFrameToImage

The tutorial uses a bag as the data source, so remember:

- Do not play back a bag against a live robot!
- Do not set /use_sim_time on a live robot!

Questions?

http://www.ros.org/

http://opencv.willowgarage.com/

ros-users@code.ros.org