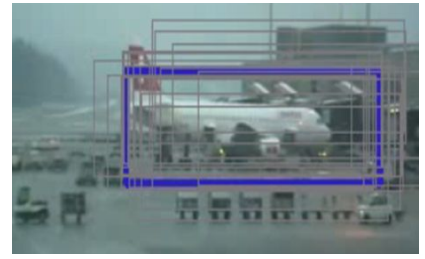


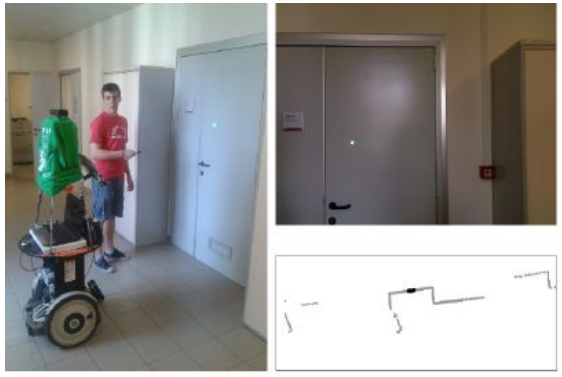
**UNIVERSITÀ DEGLI STUDI DELLA BASILICATA**

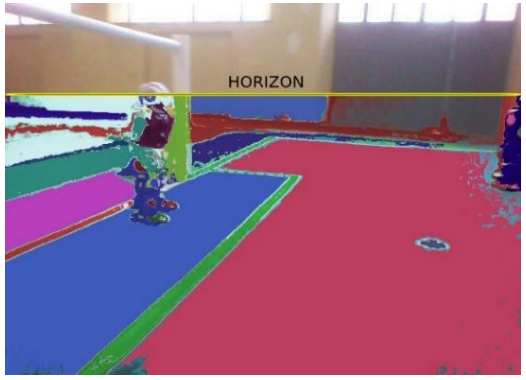
*Corso di Visione e Percezione*

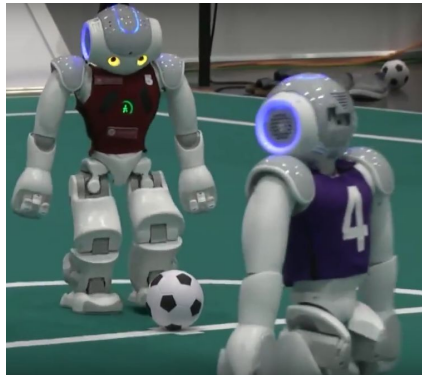
Docente
Domenico D. Bloisi

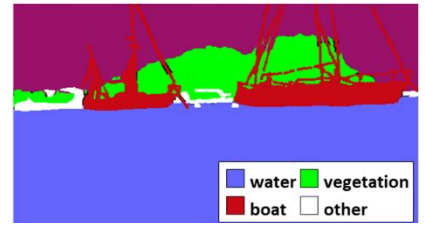



# ROS+OpenCV



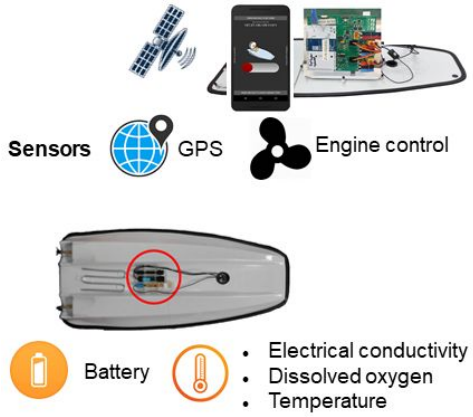


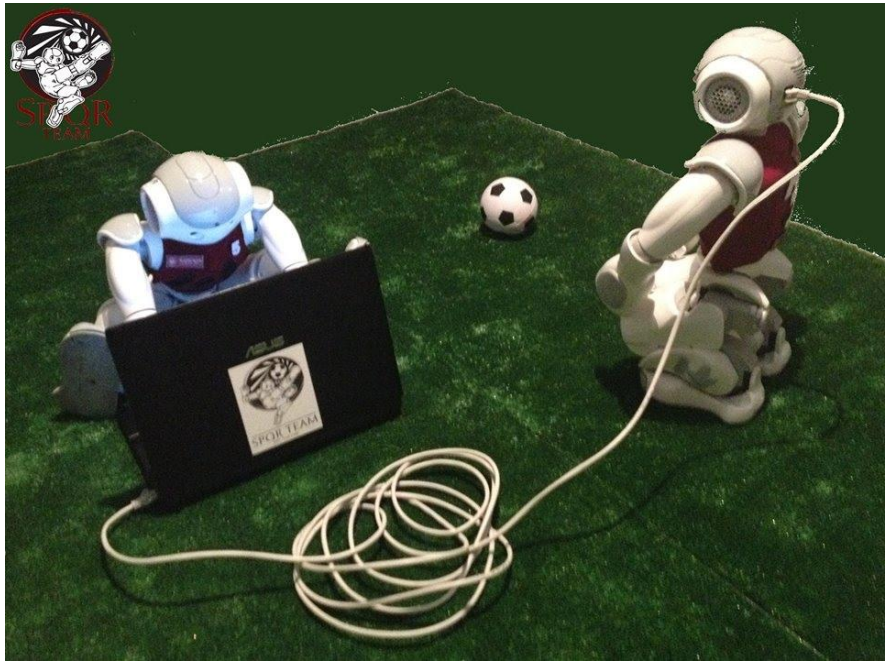



# Domenico Daniele Bloisi

- Ricercatore RTD B
  Dipartimento di Matematica, Informatica ed Economia
  Università degli studi della Basilicata
  http://web.unibas.it/bloisi

- SPQR Robot Soccer Team
  Dipartimento di Informatica, Automatica e Gestionale Università degli studi di Roma “La Sapienza”
  http://spqr.diag.uniroma1.it



# Informazioni sul corso

- Home page del corso
  http://web.unibas.it/bloisi/corsi/visione-e-percezione.html
- Docente: Domenico Daniele Bloisi
- Periodo: II semestre marzo 2021 – giugno 2021

  Martedì 17:00-19:00 (Aula COPERNICO)
  Mercoledì 8:30-10:30 (Aula COPERNICO)

Codice corso Google Classroom:
https://classroom.google.com/c/NjI2MjA4MzgzNDFa?cjc=xgolays

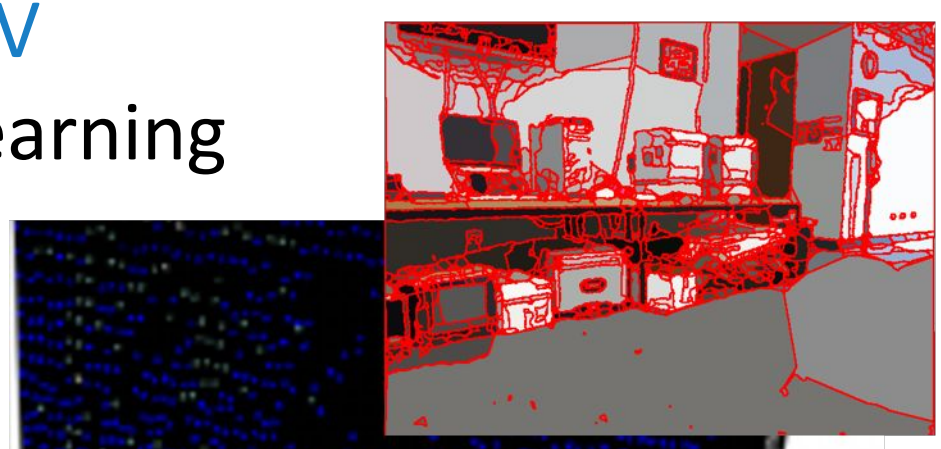
# Ricevimento

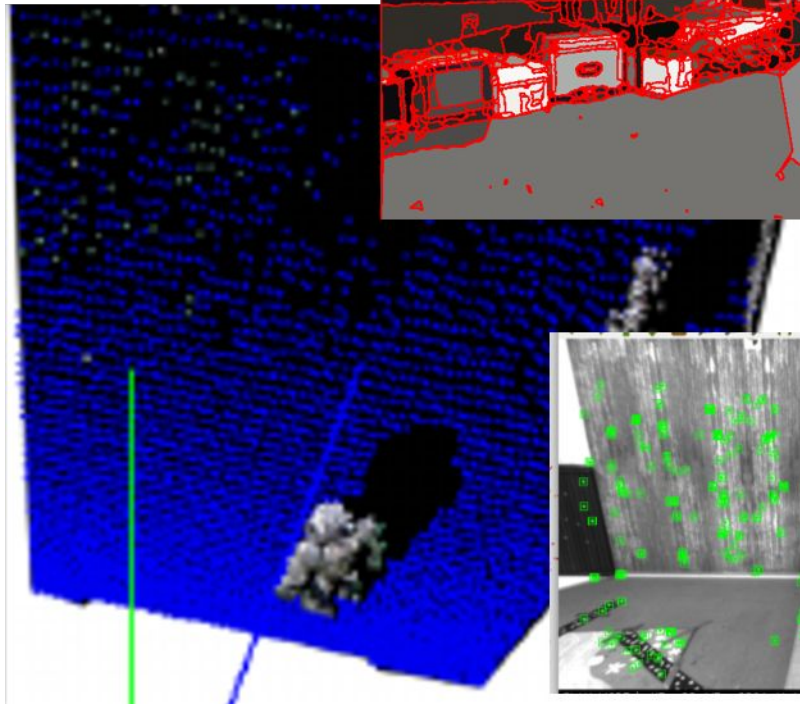
- Su appuntamento tramite Google Meet

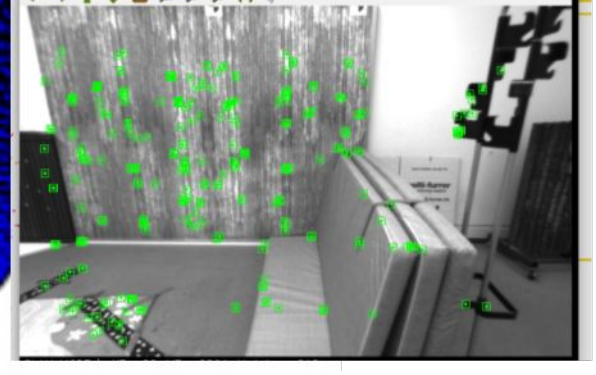
Per prenotare un appuntamento inviare una email a domenico.bloisi@unibas.it

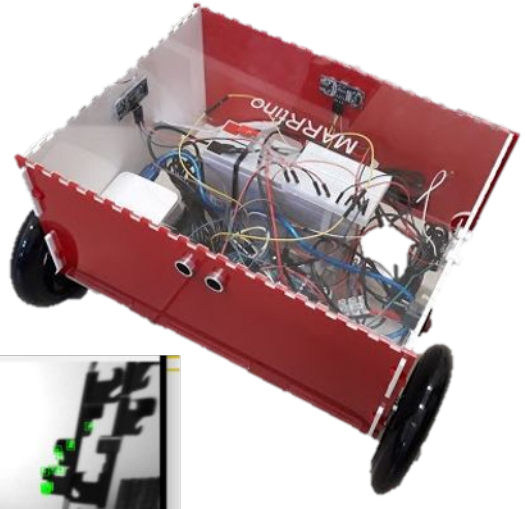
# Programma – Visione e Percezione

- Introduzione al linguaggio Python
- Elaborazione delle immagini con Python
- Percezione 2D – OpenCV
- Introduzione al Deep Learning
- ROS
- Il paradigma publisher and subscriber
- Simulatori
- Percezione 3D - PCL









# References and credits

Alcune di queste slide si basano sul materiale contenuto nel libro

YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim,
*“ROS Robot Programming - A Handbook Written by TurtleBot3 Developers”*
http://www.robotis.com/service/download.php?no=719

# rosbag

- I dati contenuti nei messaggi ROS possono essere registrati in appositi file
- Il file che contiene i messaggi prende il nome di bag e ha l'estensione ".bag"
- Il vantaggio offerto dai file di bag è quello di avere una registrazione che può essere utilizzata più volte, riproducendo ogni volta l'esatto scenario operativo in cui la bag è stata registrata

# rosbag per i dati dei sensori

- Un esempio dell'utilità dei file di bag è dato dalla registrazione dei messaggi contenenti i dati prodotti dai sensori del robot
- Durante gli esperimenti con il robot reale, i dati dei sensori possono essere registrati in una bag
- I messaggi registrati possono essere poi caricati senza la necessità di ripetere l'esperimento, permettendo così di sviluppare con maggiore facilità algoritmi che richiedano modifiche frequenti dei parametri

# Usare rosbag

`rosbag` è un package ROS per creare, riprodurre e comprimere bag di messaggi. Una bag è un file contenente i dati relativi a messaggi serializzati

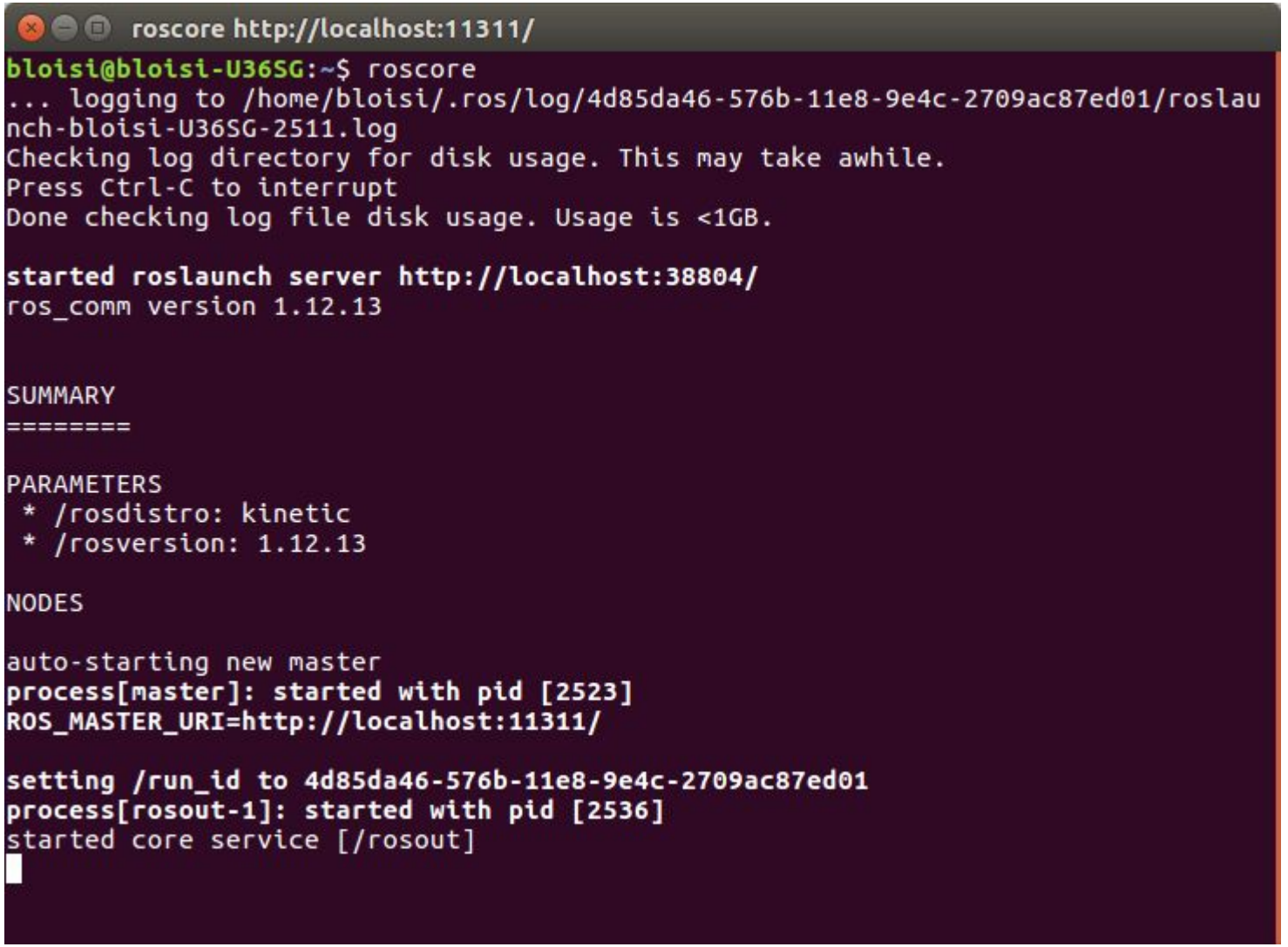
- `rosbag record` → record all the topics
- `rosbag info bag-name` → info on the recorded bag
- `rosbag play --pause bag-name` → play the recorded bag, starting paused
- `rosbag play -r #number bag-name` → play the recorded bag at rate #number

# Comandi rosbag

| Command | Description |
| --- | --- |
| rosbag record [OPTION] [TOPIC_NAME] | Record the message of a specific topic on the bsg file |
| rosbag info [FILE_NAME] | Check information of a bag file |
| rosbag play [FILE_NAME] | Play a specific bag file |
| rosbag compress [FILE_NAME] | Compress a specific bag file |
| rosbag decompress [FILE_NAME] | Decompresses a specific bag file |
| rosbag filter [INPUT_FILE] [OUTPUT_FILE] [OPTION] | Create a new bag file with the specific content removed |
| rosbag reindex bag [FILE_NAME] | Reindex |
| rosbag check bag [FILE_NAME] | Check if the specific bag file can be played in the current system |
| rosbag fix [INPUT_FILE] [OUTPUT_FILE] [OPTION] | Fix the bag file version that was saved as an incompatible version |

# Esempio rosbag

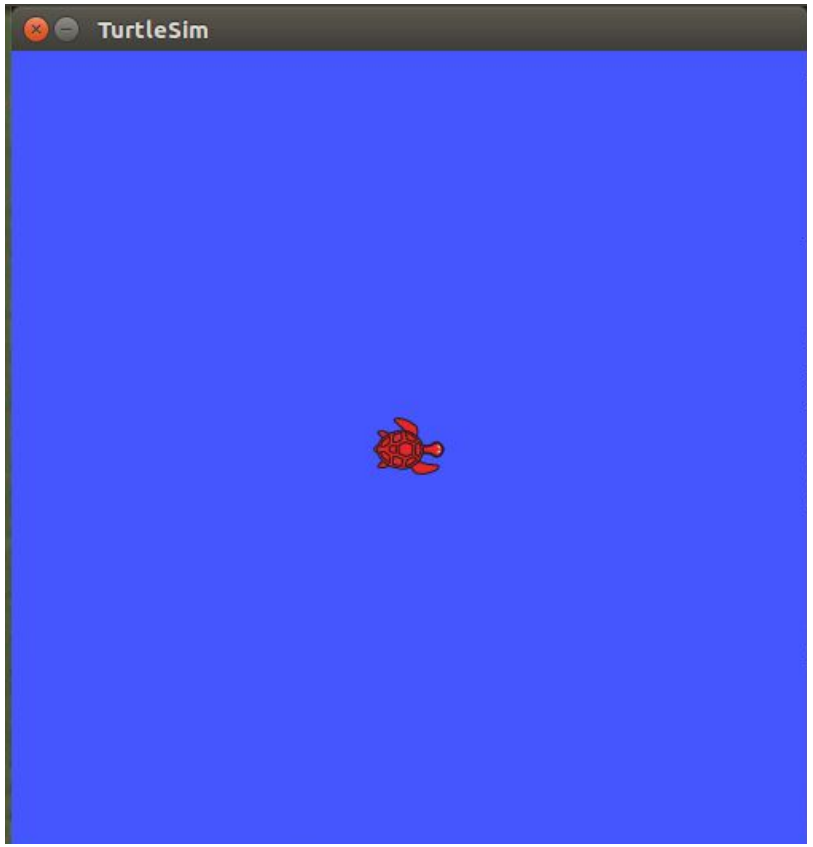
Apriamo un primo terminal e digitiamo il comando `roscore`

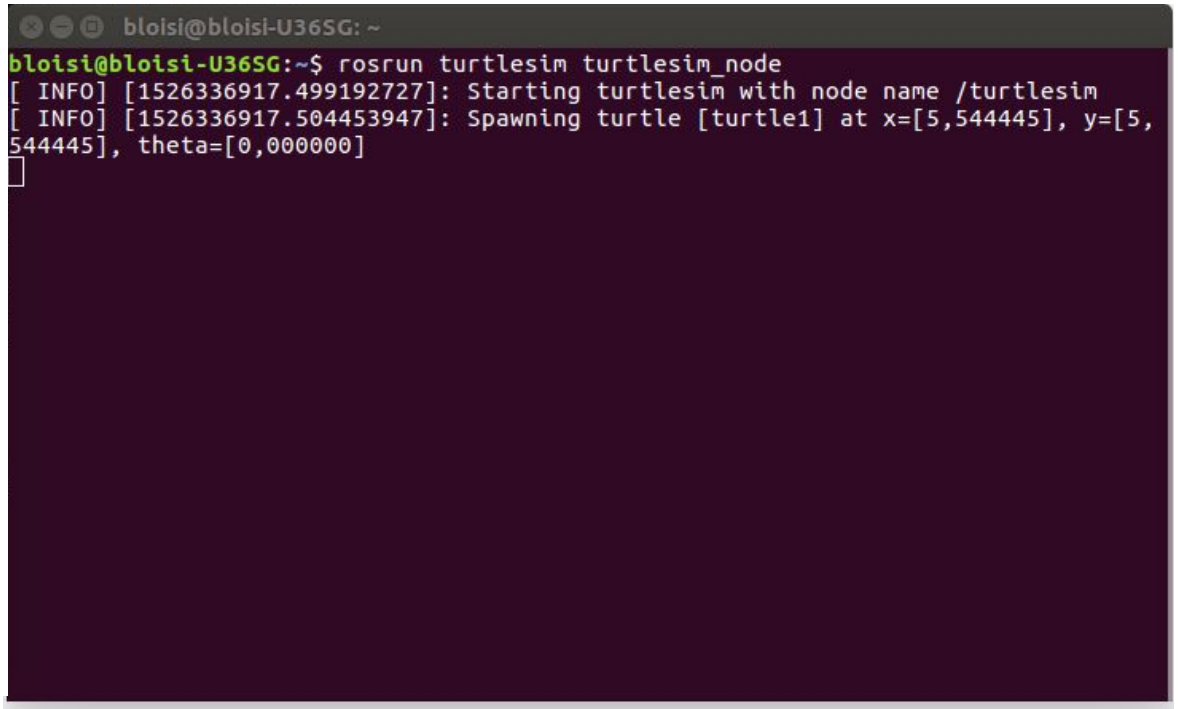


# Esempio rosbag – turtlesim

Apriamo un secondo terminal e digitiamo

```
rosrun turtlesim turtlesim_node
```

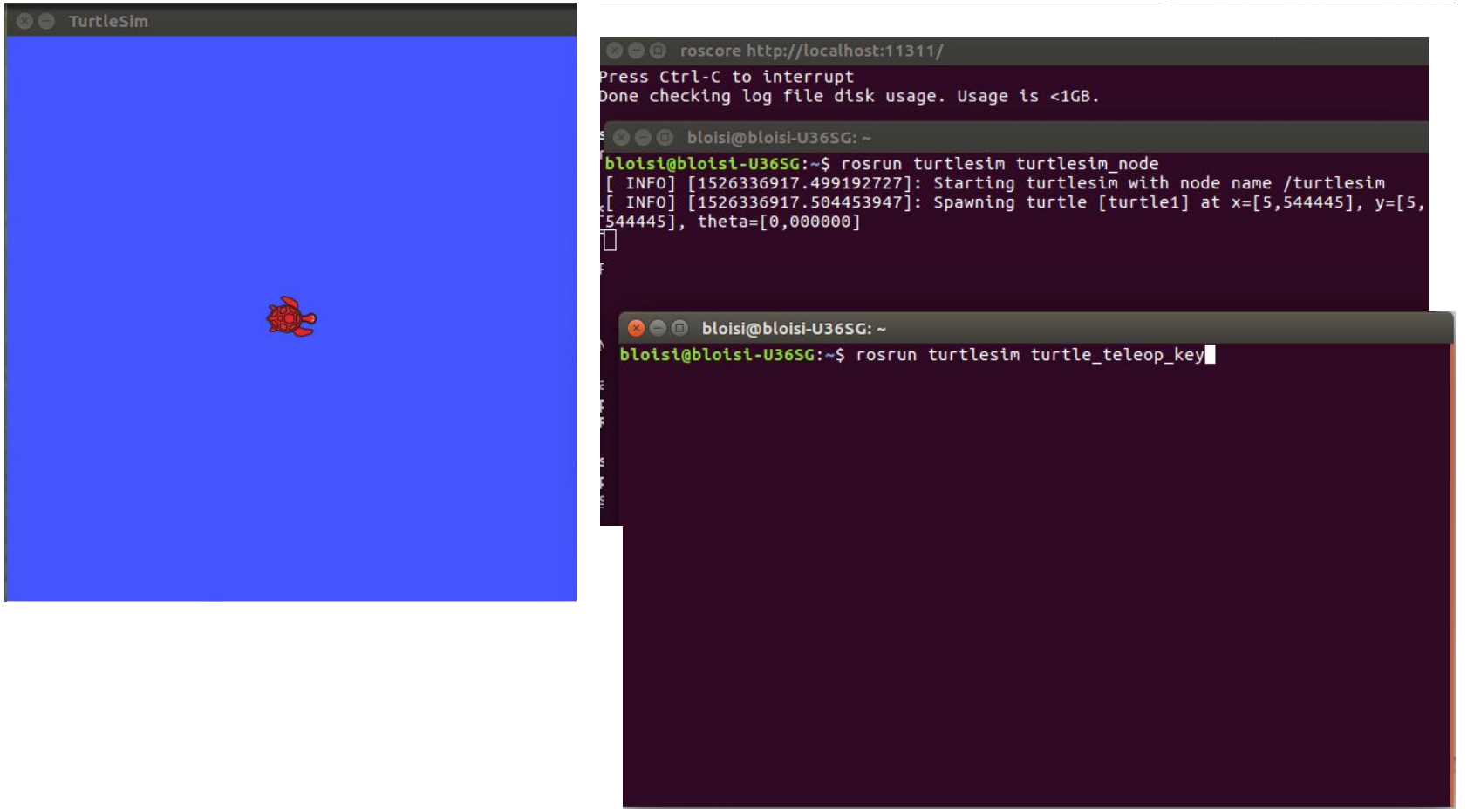


# Esempio rosbag - teleop

Apriamo un terzo terminal e digitiamo

```
rosrun turtlesim turtle_teleop_key
```

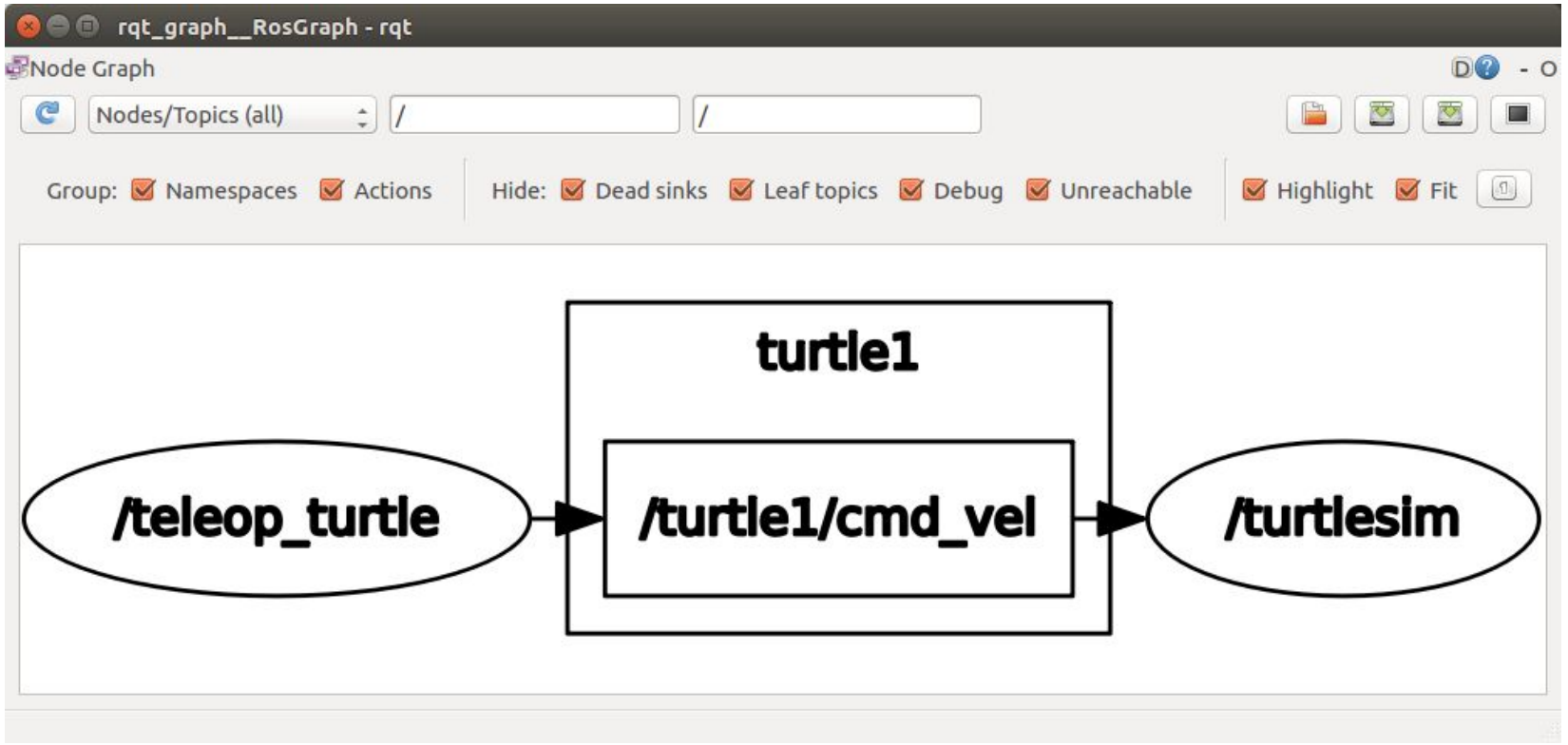


# rqt_graph

Apriamo un quarto terminal e digitiamo

`rqt_graph`

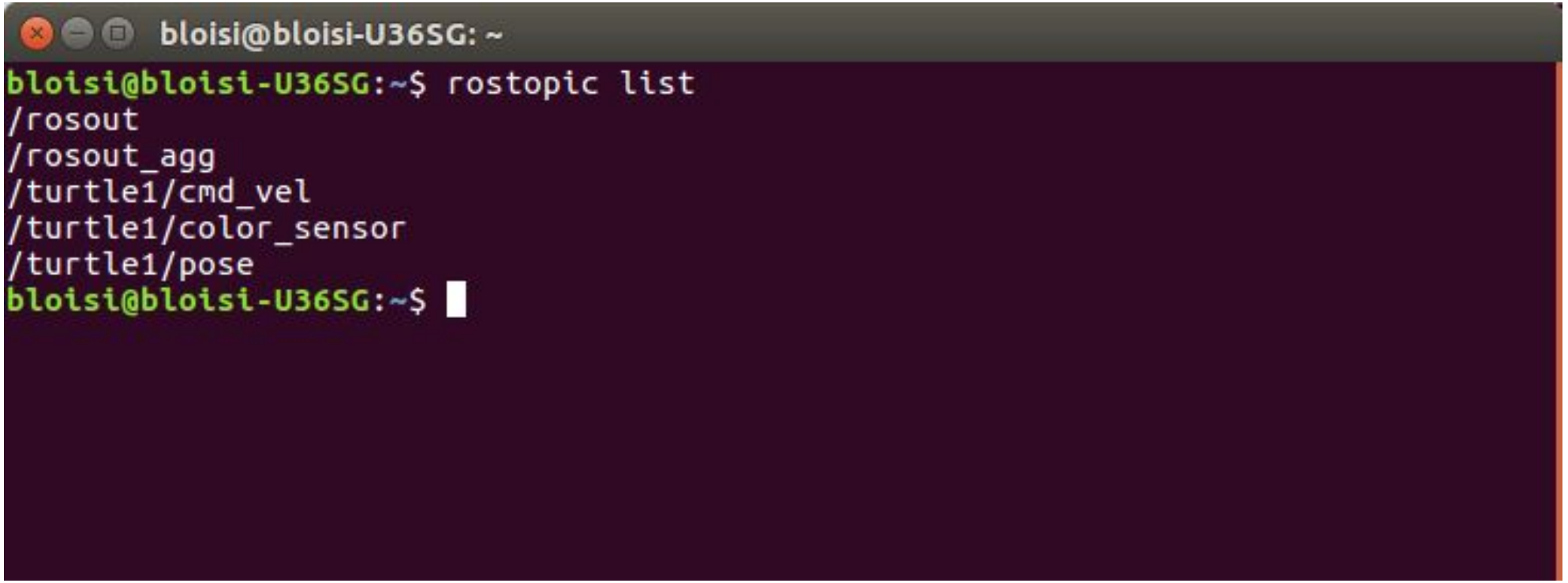
per verificare che i due nodi siano in collegamento tra loro



# rostopic

Controlliamo anche a lista dei topic attivi

`rostopic list`



# Registrare un topic

Tra tutti i topic attivi, possiamo scegliere quali registrare usando i comandi e le opzioni di rosbag

```
rosbag record <topic name>
```

Per esempio, per registrare i comandi inviati tramite `cmd_vel` useremo

```
rosbag record /turtle1/cmd_vel
```

# Esempio - Registrare un topic

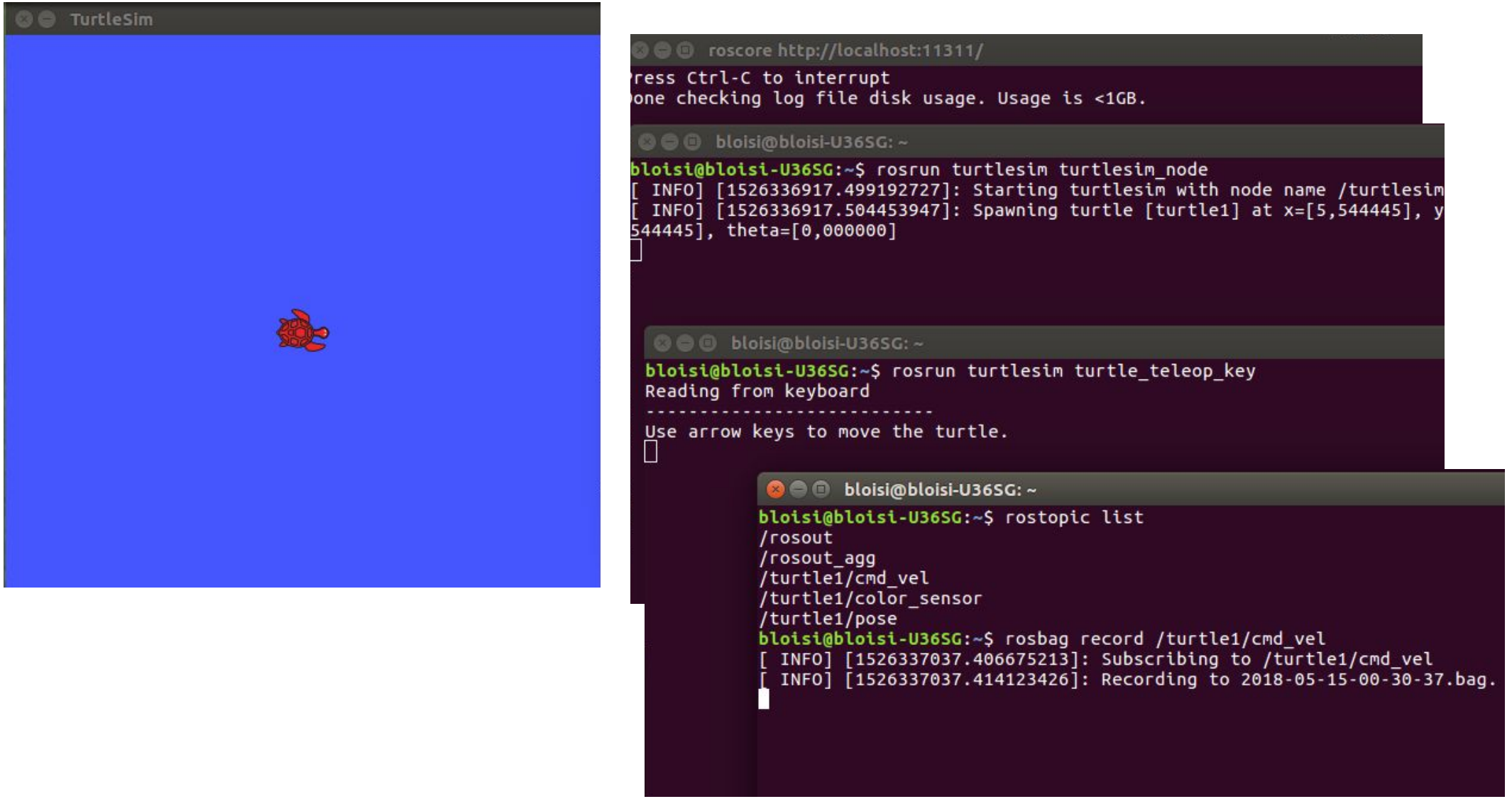

```
roscore http://localhost:11311/
ress Ctrl-C to interrupt
one checking log file disk usage. Usage is <1GB.
```

```
bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtlesim_node
[ INFO] [1526336917.499192727]: Starting turtlesim with node name /turtlesim
[ INFO] [1526336917.504453947]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
```

```
bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtle_teleop_key
Reading from keyboard
---------------------------
Use arrow keys to move the turtle.
```

```
bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ rostopic list
/rosout
/rosout_agg
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
bloisi@bloisi-U36SG:~$ rosbag record /turtle1/cmd_vel
```

# Esempio – Registrazione in corso…



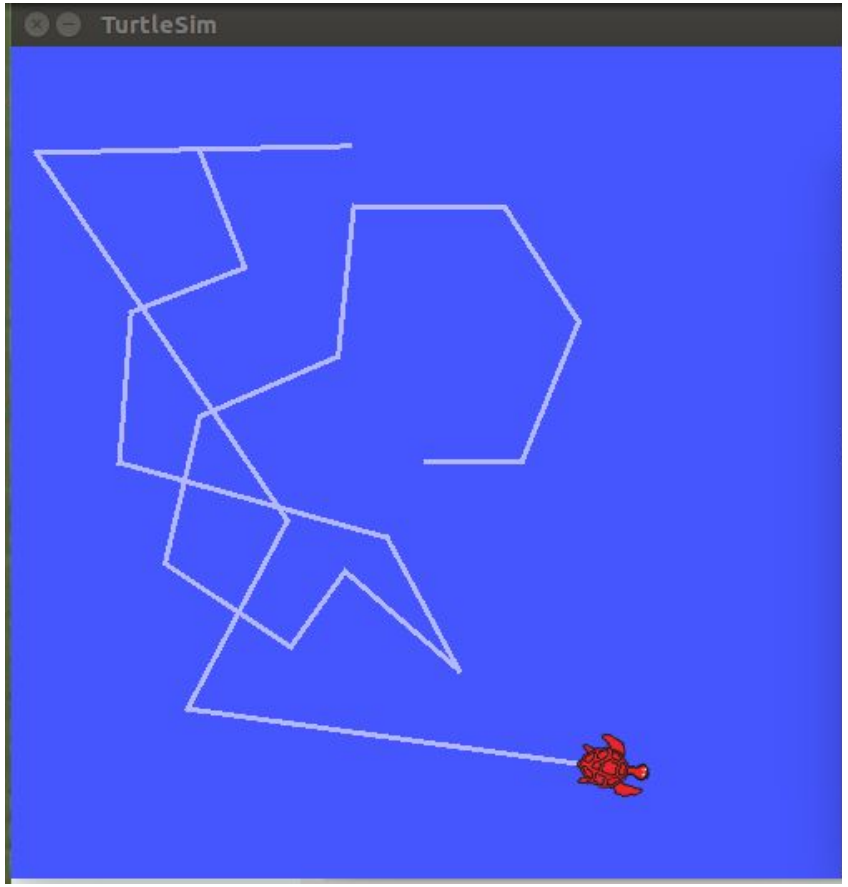
# Registrare tutti i topic

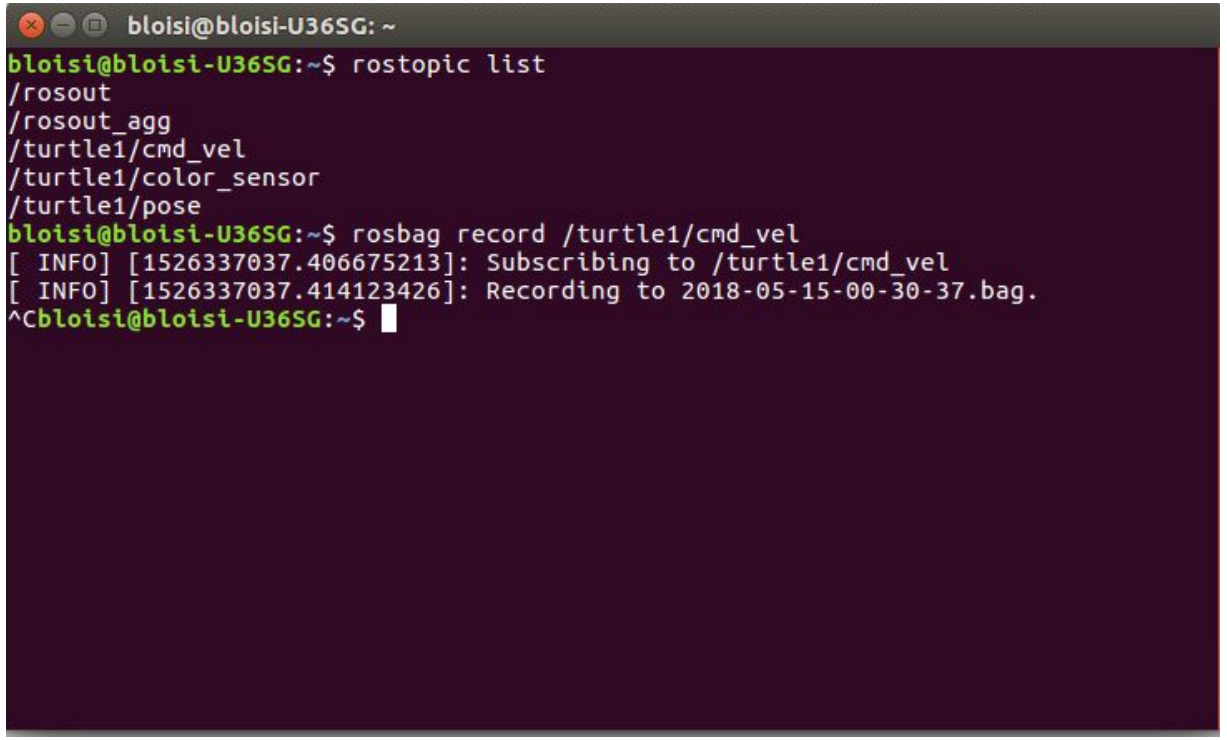
Se si vogliono registrare tutti i topic attivi, si può usare l'opzione `-a`

```
rosbag record -a
```

# Terminare la registrazione

[Ctrl-C] ci permette di terminare la registrazione della bag

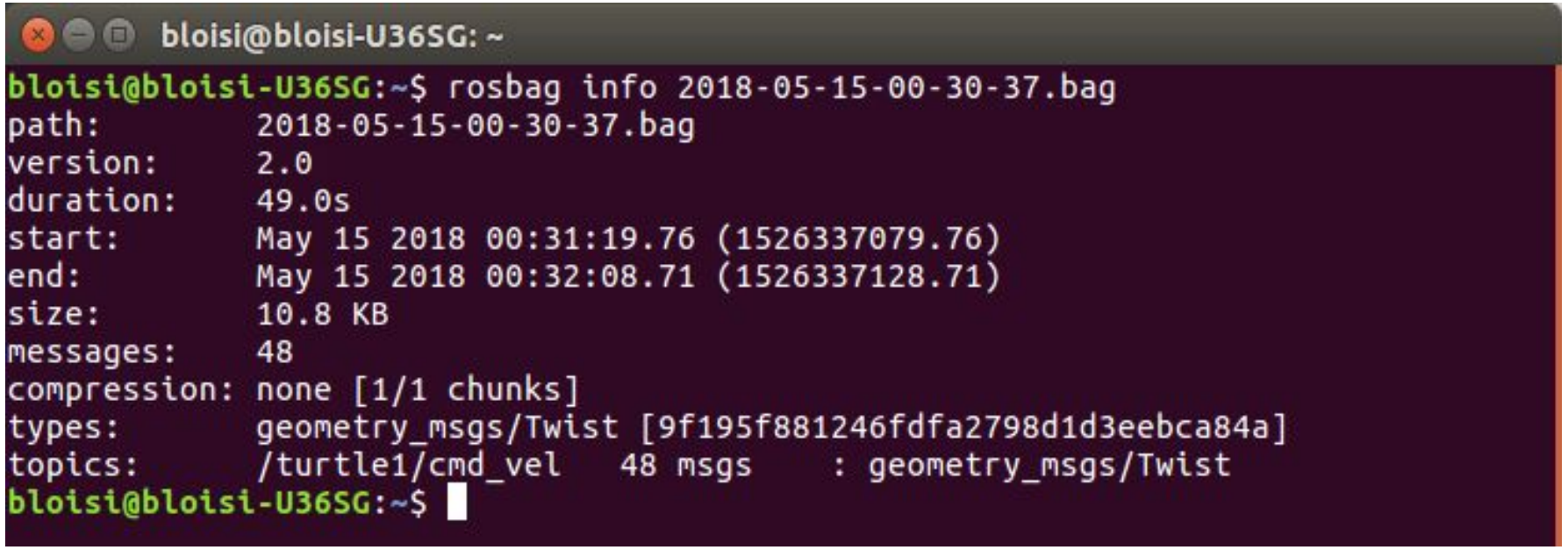


# rosbag info

- Il comando `info` stampa a video informazioni sulla bag fornita come parametro
- Ad esempio, avremo informazioni sul file 2018-05-15-00-30-37.bag digitando

  `rosbag info 2018-05-15-00-30-37.bag`



# rosbag play

Avendo a disposizione una bag registrata, si può riprodurla tramite

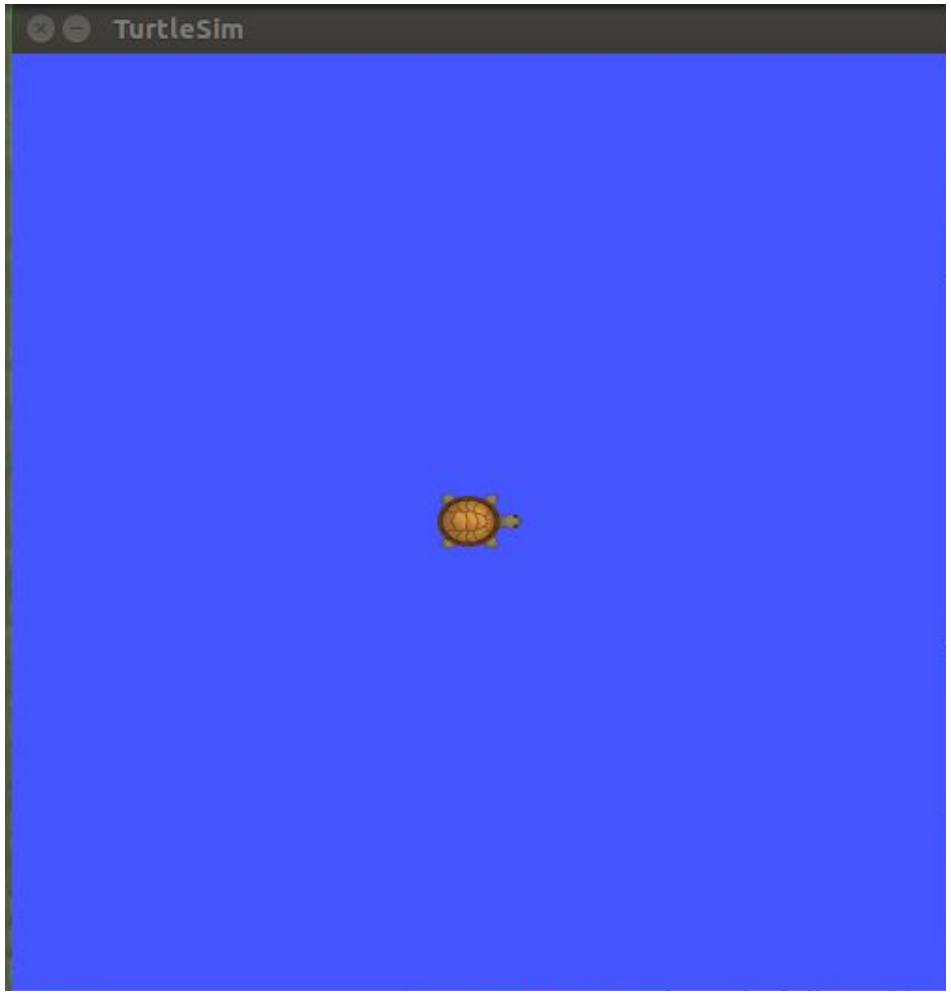
```
rosbag play <bagfile name>
```

Per riprodurre la bag 2018-05-15-00-30-37.bag:

1. Terminiamo tutti i nodi attivi
2. Lanciamo il nodo `turtlesim_node`
3. Digitiamo il comando

```
rosbag play 2018-05-15-00-30-37.bag
```

# rosbag play – esecuzione




```
bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtlesim_node
[ INFO] [1526336917.499192727]: Starting turtlesim with node name /turtlesim
[ INFO] [1526336917.504453947]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
^C
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtlesim_node
[ INFO] [1526337414.146208633]: Starting turtlesim with node name /turtlesim
[ INFO] [1526337414.151953802]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
```

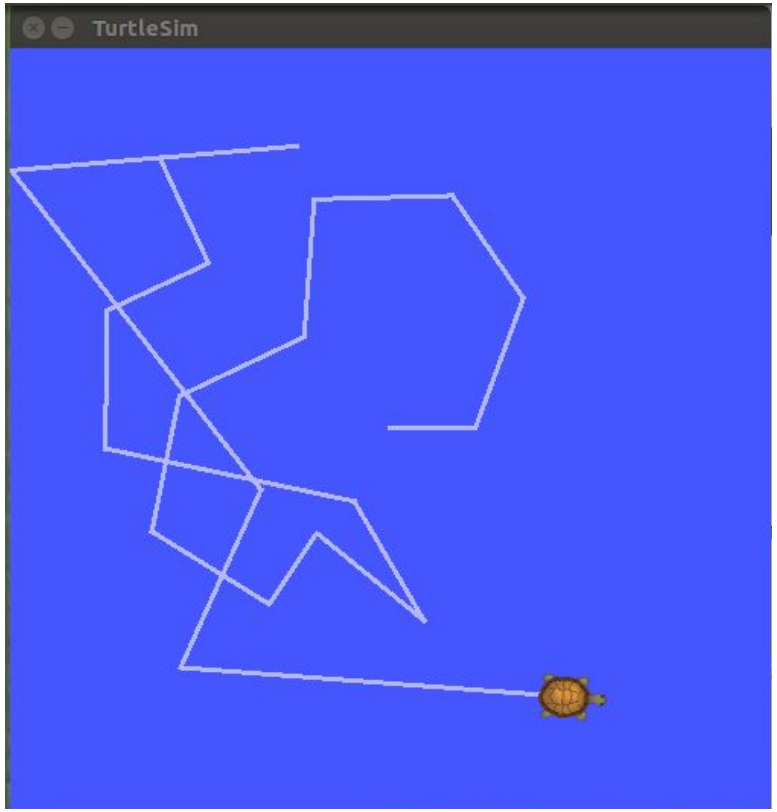
```
bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ rosbag play 2018-05-15-00-30-37.bag
```

# rosbag play – esecuzione in corso


```
bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtlesim_node
[ INFO] [1526336917.499192727]: Starting turtlesim with node name /turtlesim
[ INFO] [1526336917.504453947]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
^C
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtlesim_node
[ INFO] [1526337414.146208633]: Starting turtlesim with node name /turtlesim
[ INFO] [1526337414.151953802]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
```

```
bloisi@bloisi-U36SG: ~
[RUNNING]  Bag Time: 1526337090.591895   Duration: 10.836131 / 48.957648
[RUNNING]  Bag Time: 1526337090.692036   Duration: 10.936272 / 48.957648
[RUNNING]  Bag Time: 1526337090.791511   Duration: 11.035747 / 48.957648
[RUNNING]  Bag Time: 1526337090.891664   Duration: 11.135900 / 48.957648
[RUNNING]  Bag Time: 1526337090.991832   Duration: 11.236068 / 48.957648
[RUNNING]  Bag Time: 1526337091.091962   Duration: 11.336197 / 48.957648
[RUNNING]  Bag Time: 1526337091.192129   Duration: 11.436365 / 48.957648
[RUNNING]  Bag Time: 1526337091.248360   Duration: 11.492596 / 48.957648
[RUNNING]  Bag Time: 1526337091.348569   Duration: 11.592805 / 48.957648
[RUNNING]  Bag Time: 1526337091.448749   Duration: 11.692984 / 48.957648
[RUNNING]  Bag Time: 1526337091.548924   Duration: 11.793160 / 48.957648
[RUNNING]  Bag Time: 1526337091.649109   Duration: 11.893345 / 48.957648
[RUNNING]  Bag Time: 1526337091.749240   Duration: 11.993475 / 48.957648
[RUNNING]  Bag Time: 1526337091.849477   Duration: 12.093713 / 48.957648
[RUNNING]  Bag Time: 1526337091.949632   Duration: 12.193868 / 48.957648
[RUNNING]  Bag Time: 1526337092.049816   Duration: 12.294051 / 48.957648
[RUNNING]  Bag Time: 1526337092.150008   Duration: 12.394243 / 48.957648
[RUNNING]  Bag Time: 1526337092.250157   Duration: 12.494392 / 48.957648
[RUNNING]  Bag Time: 1526337092.312010   Duration: 12.556246 / 48.957648
[RUNNING]  Bag Time: 1526337092.412182   Duration: 12.656418 / 48.957648
[RUNNING]  Bag Time: 1526337092.512428   Duration: 12.756664 / 48.957648
[RUNNING]  Bag Time: 1526337092.612576   Duration: 12.856811 / 48.957648
[RUNNING]  Bag Time: 1526337092.712744   Duration: 12.956980 / 48.957648
```

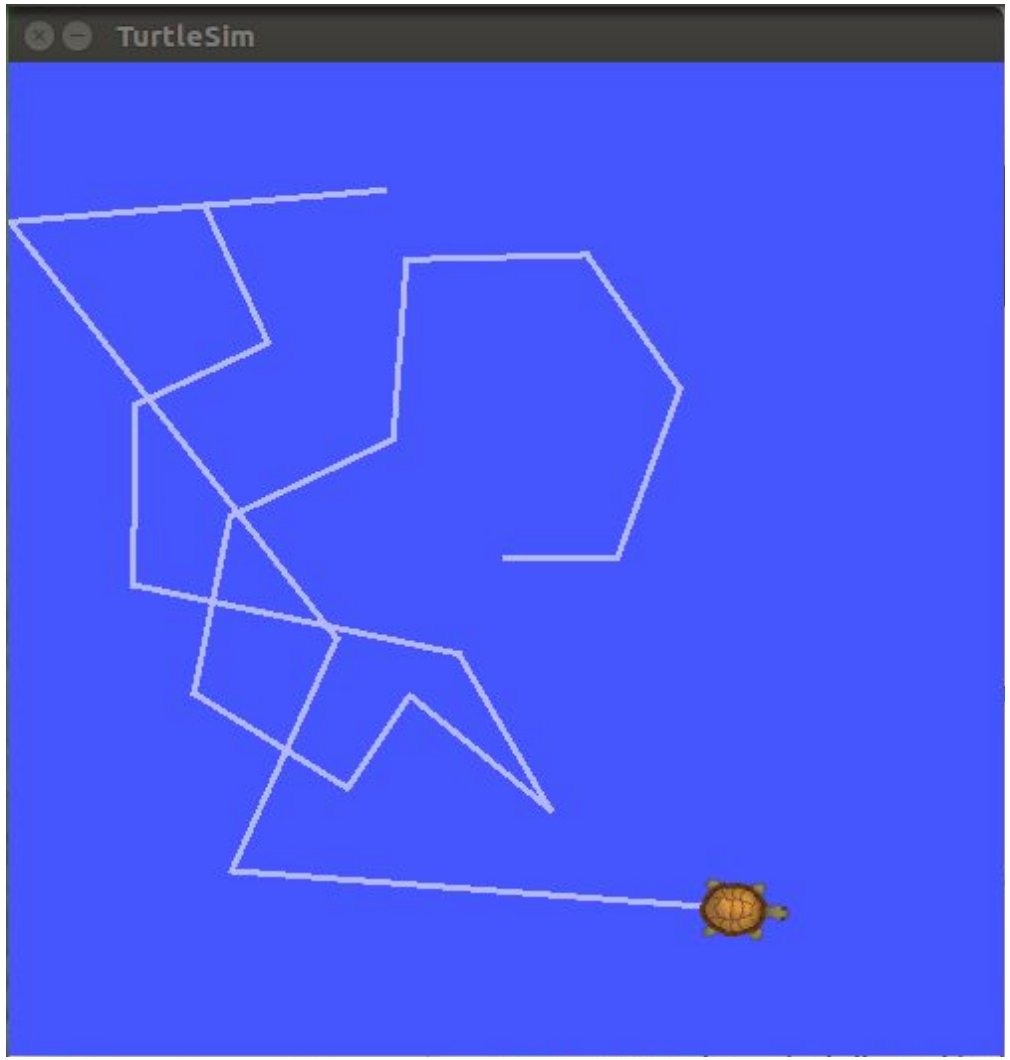
# rosbag play – risultato finale

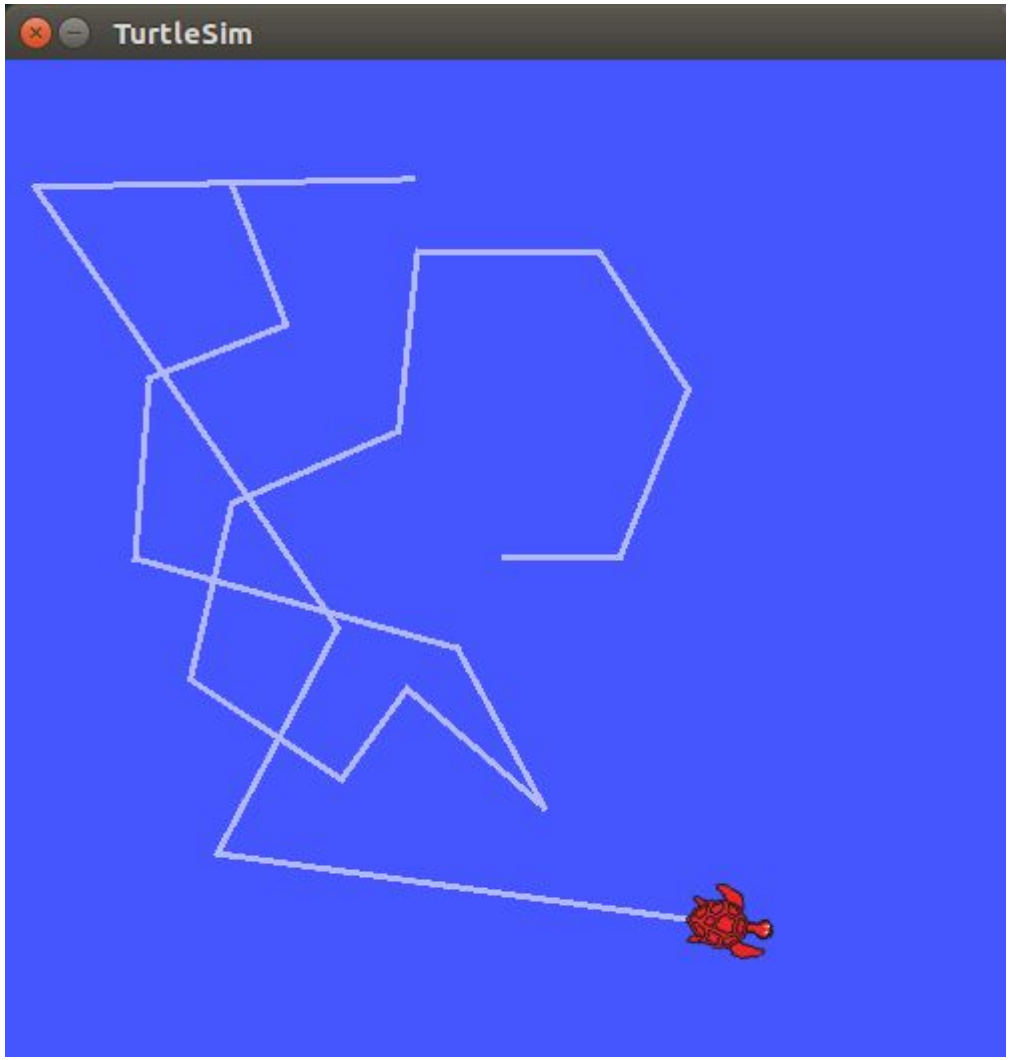



```
bloisi@bloisi-U36SG: ~
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtlesim_node
[ INFO] [1526336917.499192727]: Starting turtlesim with node name /turtlesim
[ INFO] [1526336917.504453947]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
^C
bloisi@bloisi-U36SG:~$ rosrun turtlesim turtlesim_node
[ INFO] [1526337414.146208633]: Starting turtlesim with node name /turtlesim
[ INFO] [1526337414.151953802]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]
```

```
bloisi@bloisi-U36SG: ~
[RUNNING]  Bag Time: 1526337126.557608   Duration: 46.801843 / 48.957648
[RUNNING]  Bag Time: 1526337126.657763   Duration: 46.901998 / 48.957648
[RUNNING]  Bag Time: 1526337126.757943   Duration: 47.002178 / 48.957648
[RUNNING]  Bag Time: 1526337126.858106   Duration: 47.102342 / 48.957648
[RUNNING]  Bag Time: 1526337126.958296   Duration: 47.202532 / 48.957648
[RUNNING]  Bag Time: 1526337127.058506   Duration: 47.302742 / 48.957648
[RUNNING]  Bag Time: 1526337127.158664   Duration: 47.402900 / 48.957648
[RUNNING]  Bag Time: 1526337127.258812   Duration: 47.503048 / 48.957648
[RUNNING]  Bag Time: 1526337127.358996   Duration: 47.603232 / 48.957648
[RUNNING]  Bag Time: 1526337127.459208   Duration: 47.703444 / 48.957648
[RUNNING]  Bag Time: 1526337127.559368   Duration: 47.803604 / 48.957648
[RUNNING]  Bag Time: 1526337127.659556   Duration: 47.903791 / 48.957648
[RUNNING]  Bag Time: 1526337127.759781   Duration: 48.004016 / 48.957648
[RUNNING]  Bag Time: 1526337127.857417   Duration: 48.101653 / 48.957648
[RUNNING]  Bag Time: 1526337127.957637   Duration: 48.201872 / 48.957648
[RUNNING]  Bag Time: 1526337128.057879   Duration: 48.302115 / 48.957648
[RUNNING]  Bag Time: 1526337128.158112   Duration: 48.402348 / 48.957648
[RUNNING]  Bag Time: 1526337128.258319   Duration: 48.502555 / 48.957648
[RUNNING]  Bag Time: 1526337128.358572   Duration: 48.602807 / 48.957648
[RUNNING]  Bag Time: 1526337128.458740   Duration: 48.702976 / 48.957648
[RUNNING]  Bag Time: 1526337128.558883   Duration: 48.803119 / 48.957648

Done.
bloisi@bloisi-U36SG:~$
```

# rosbag play – confronto



# I bag file possono essere molto grandi

Un bag file registrato per un breve periodo di tempo comporta la creazione di file aventi dimensioni contenute

Se, invece, si ha bisogno di registrare messaggi per un lungo periodo di tempo, allora la dimensione del bag file può crescere fino ad occupare molta memoria

Si provi per esempio a scaricare la ROS bag a questo indirizzo

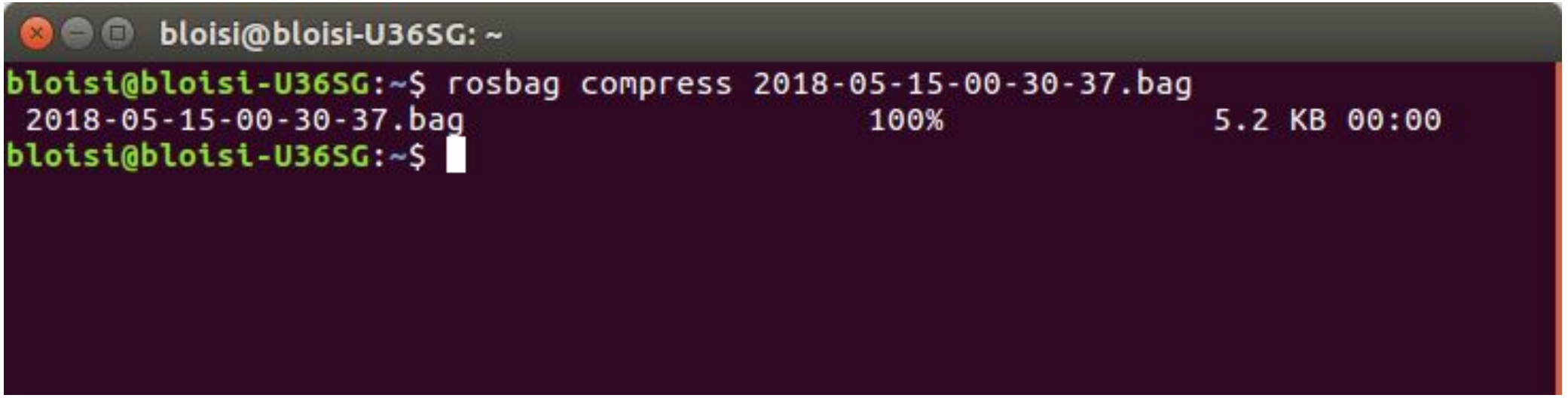
https://drive.google.com/file/d/1F8pd_Cc5n67cMkWdvTZphpi7zecRJDEJ/view?usp=sharing

# rosbag compress

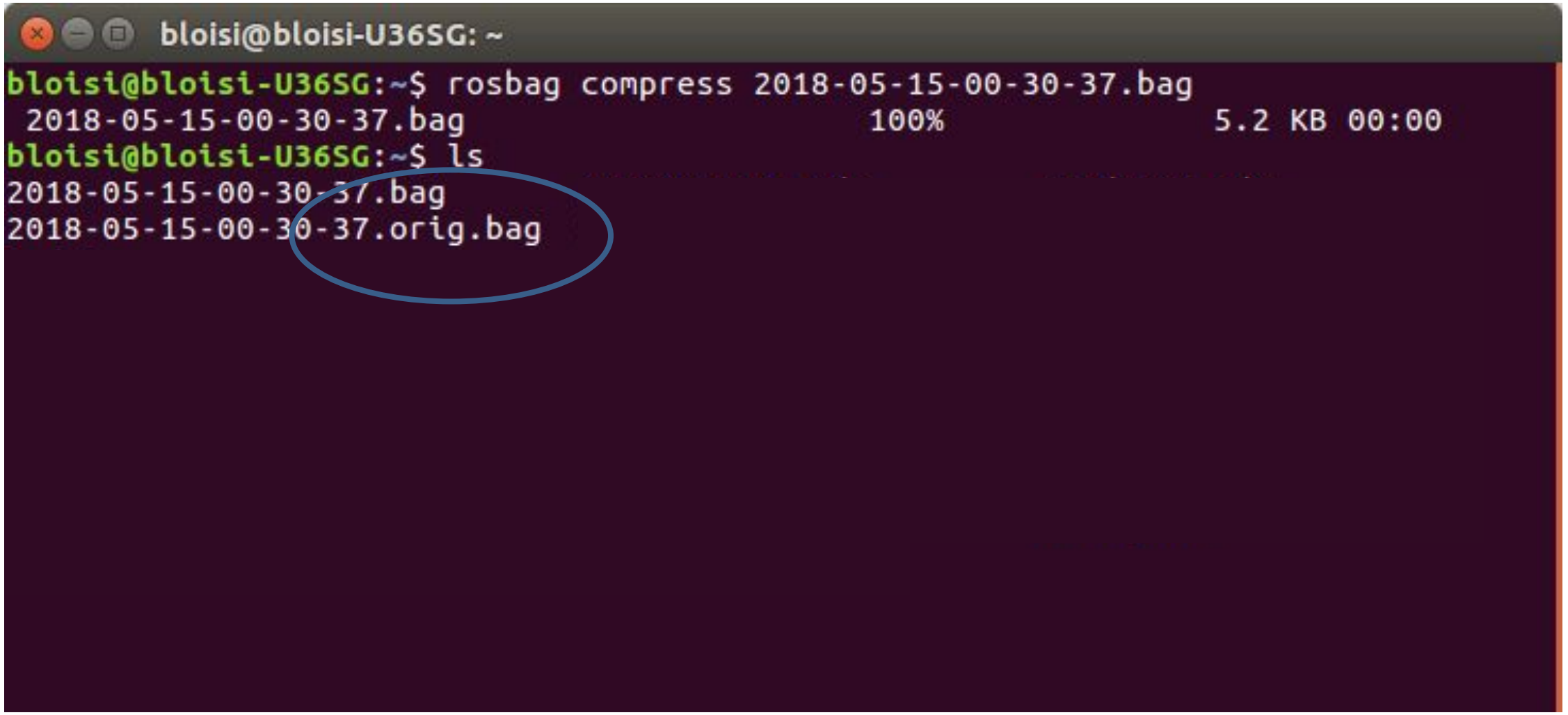
ROS fornisce la possibilità di comprimere i bag file grazie all'opzione `compress`

Esempio

```
rosbag compress 2018-05-15-00-30-37.bag
```



# rosbag compress – esecuzione



# rosbag decompress

Per riportare il bag file al suo formato originale è possibile utilizzare `decompress`

```
rosbag decompress 2018-05-15-00-30-37.bag
```

# rosbag con immagini

- Una bag può contenere qualunque tipo di dato sia possibile inviare tramite i messaggi ROS
- Le bag possono essere molto utili per la registrazione di dati provenienti da telecamere montate su robot
- In particolare, essendo presente un timestamp per ogni immagine, è possibile riprodurre fedelmente lo stream dati del sensore usato per effettuare le riprese

# Image message

## sensor_msgs/Image Message

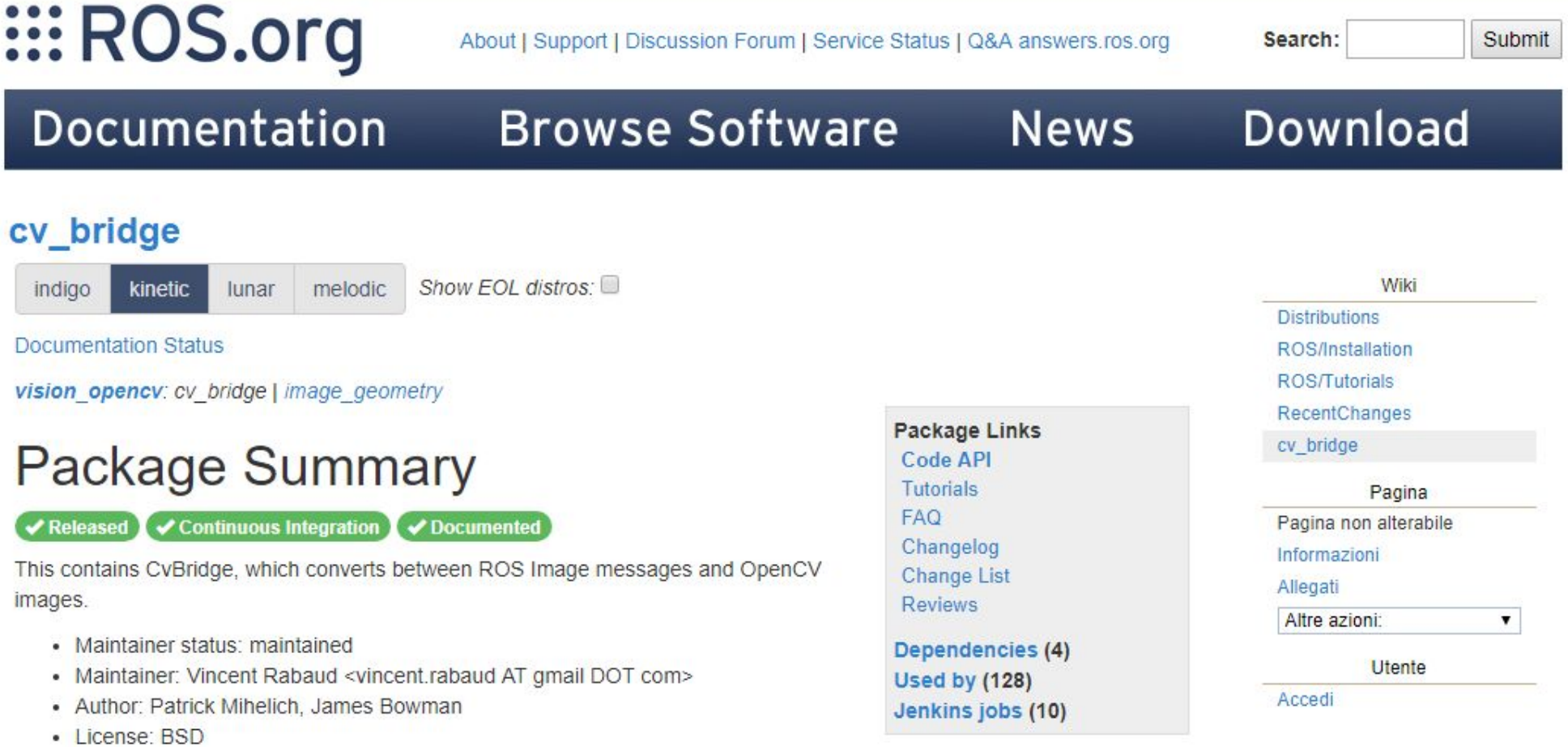
**File:** `sensor_msgs/Image.msg`

### Raw Message Definition

```
# This message contains an uncompressed image
# (0, 0) is at top-left corner of 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
```

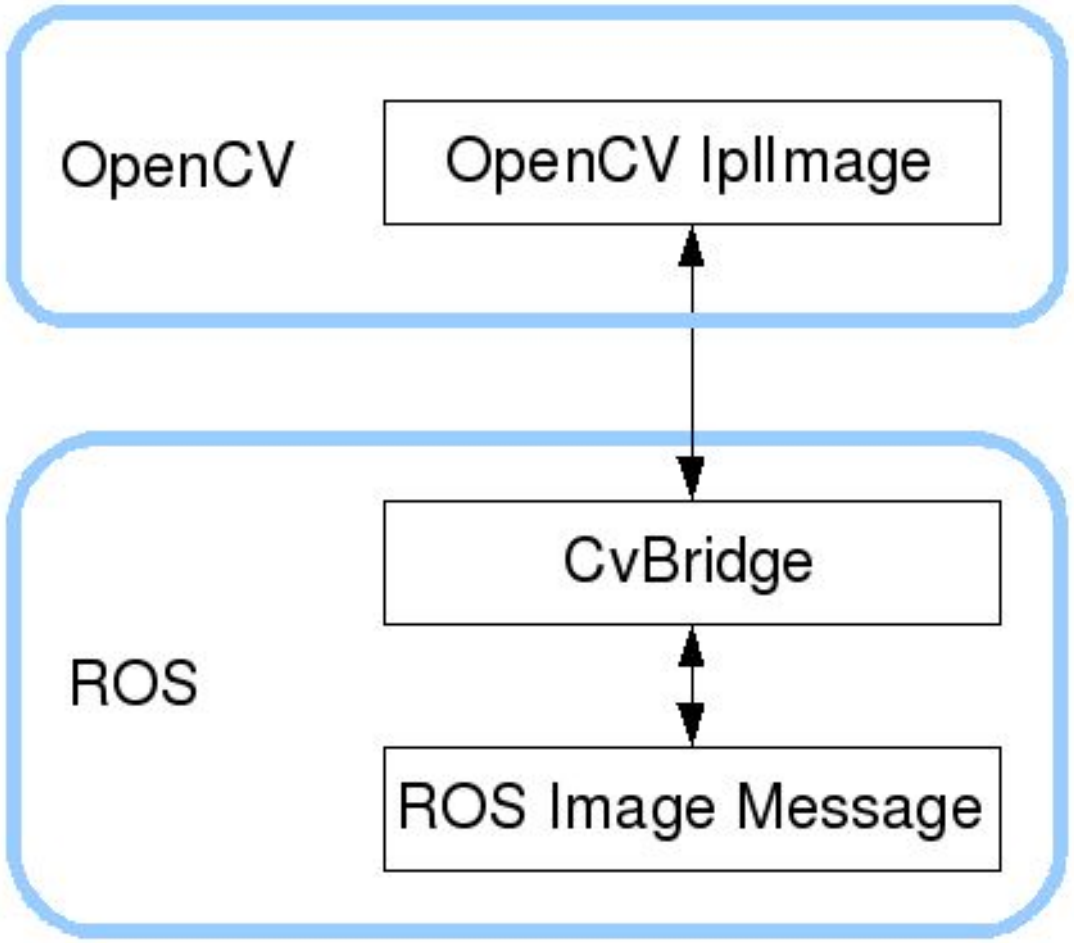
http://docs.ros.org/kinetic/api/sensor_msgs/html/msg/Image.html

# cv_bridge




http://wiki.ros.org/cv_bridge

# cv_bridge




http://wiki.ros.org/cv_bridge

# Esempio: unibas_viewer

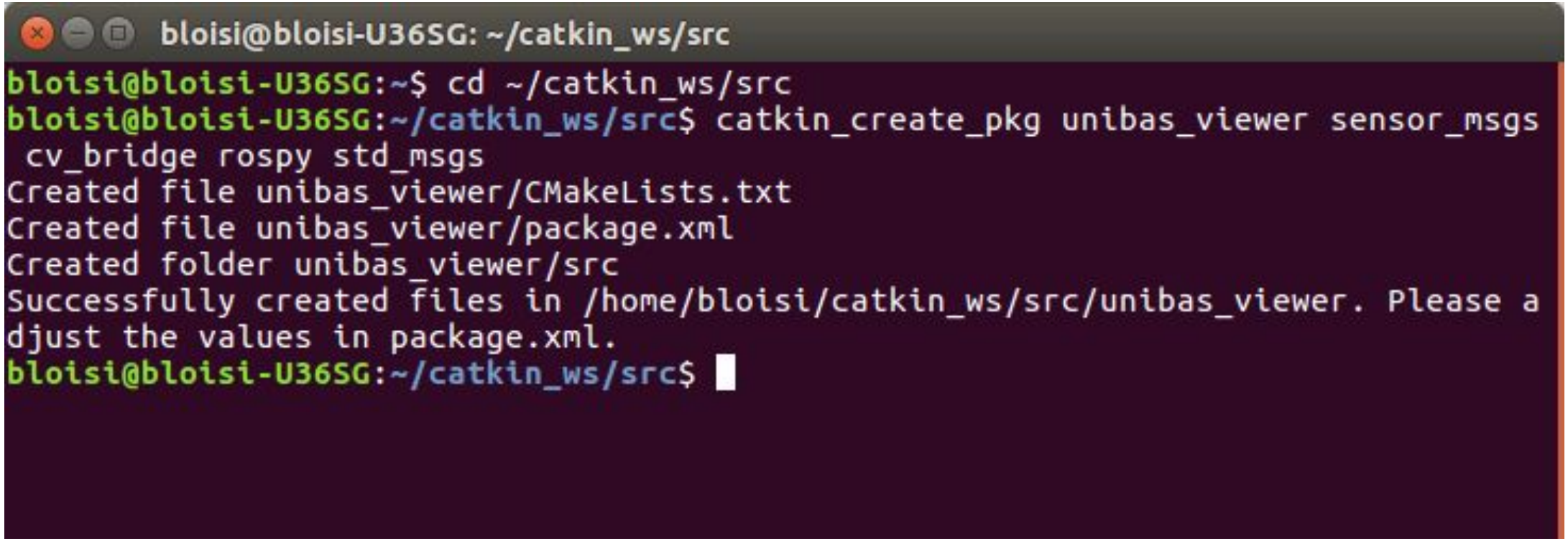
rosbag acquisita con
un sensore RGBD



unibas_viewer

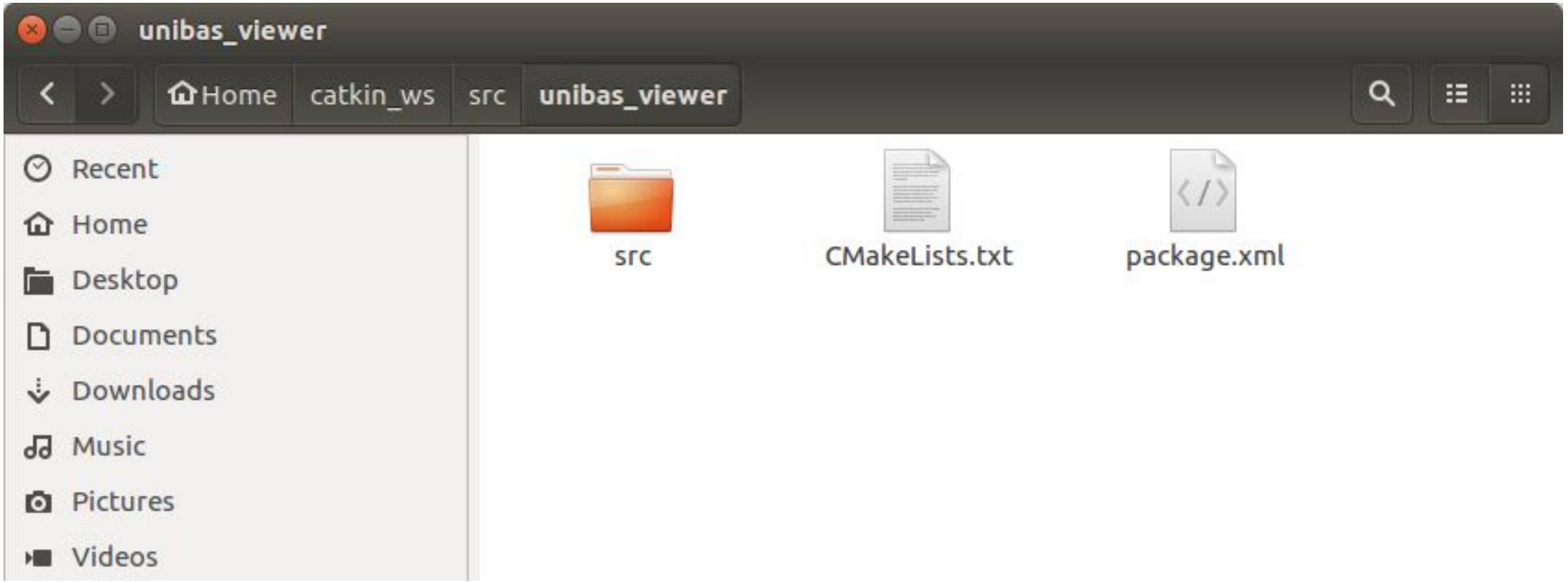
visualizzazione dati
color + depth
con OpenCV

# creazione del nodo unibas_viewer



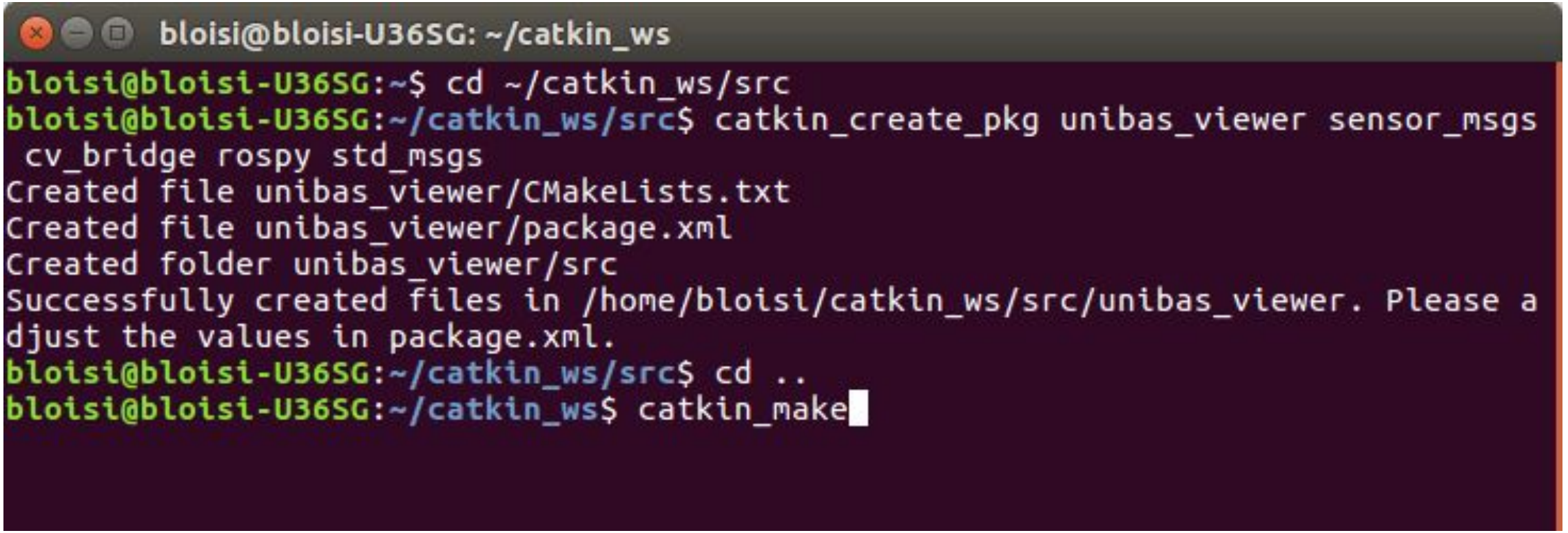

http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython

# cartella unibas_viewer




http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython

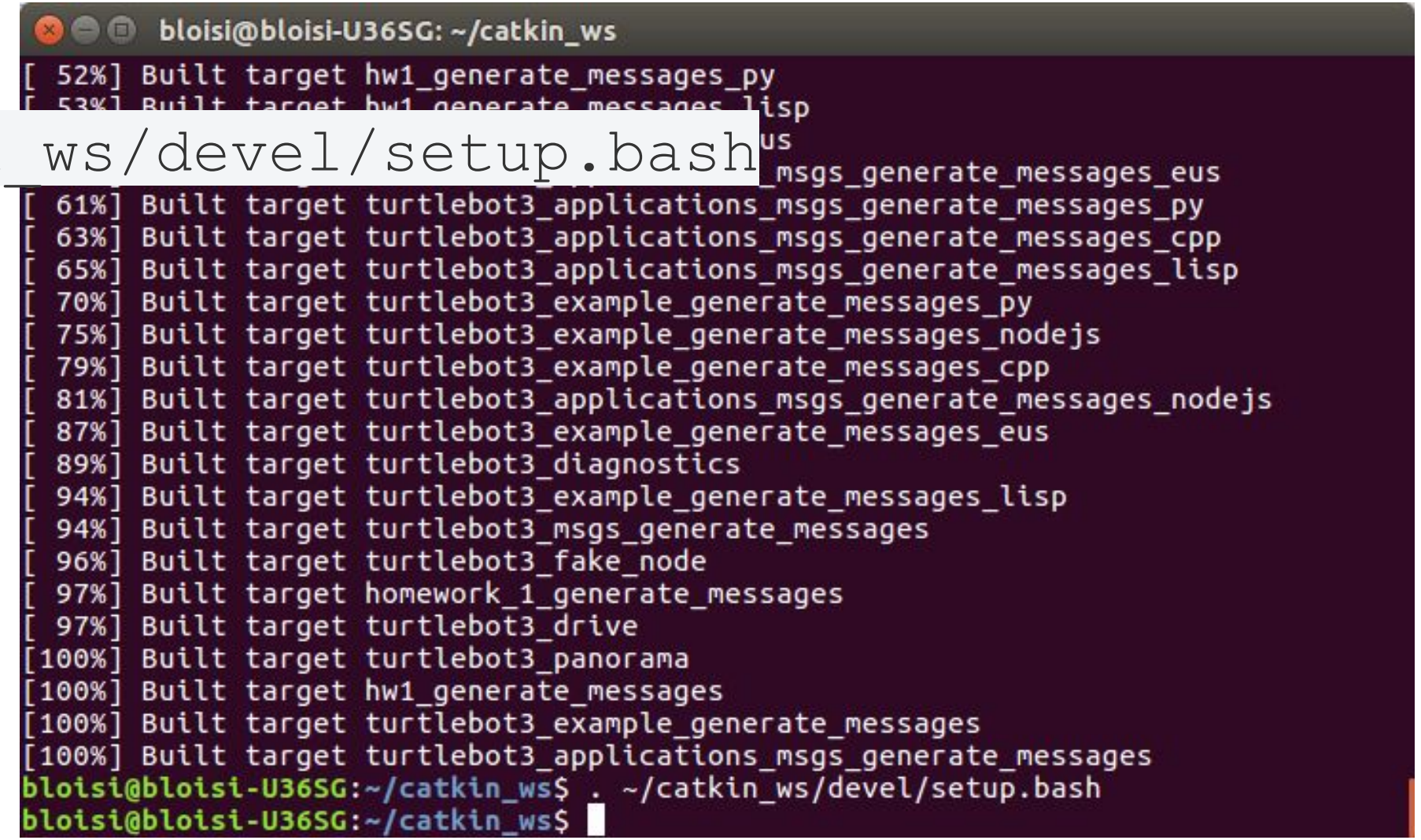
# catkin_make




http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython

# Settiamo l’ambiente ROS

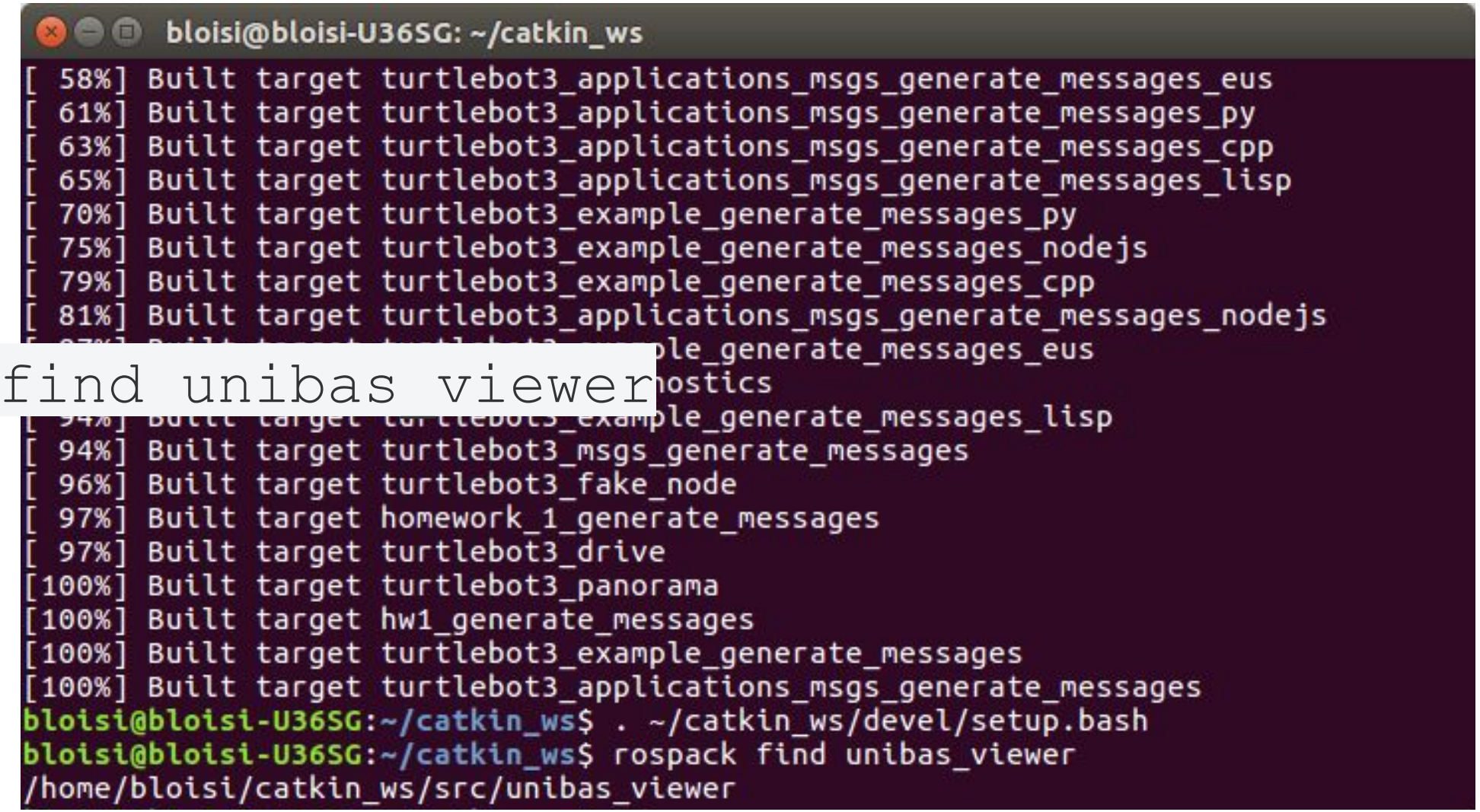
```
. ~/catkin_ws/devel/setup.bash
```




https://answers.ros.org/question/229365/do-i-really-need-to-source-catkin_wsdevelsetupbash/

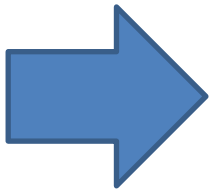
# rospack find

```
rospack find unibas viewer
```



# Creiamo unibas_rgbd_reader.py

```
1  #!/usr/bin/env python
2  from __future__ import print_function
3
4  import roslib
5  roslib.load_manifest('unibas_viewer')
6  import sys
7  import rospy
8  import cv2
9  import numpy as np
10 import message_filters
11 from std_msgs.msg import String
12 from sensor_msgs.msg import Image
13 from cv_bridge import CvBridge, CvBridgeError
14
15 class unibas_rgbd_reader:
```

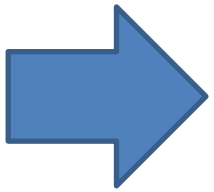


http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython

# Creiamo unibas_rgbd_reader.py

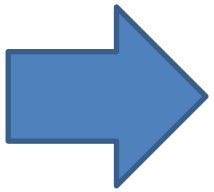
```
16
17      def __init__(self):
18          self.bridge = CvBridge()
19          self.image_sub = message_filters.Subscriber("/camera/rgb/image_raw",Image)
20          self.depth_sub = message_filters.Subscriber("/camera/depth_registered/image_raw",Image)
21          self.ts = message_filters.ApproximateTimeSynchronizer([self.image_sub, self.depth_sub], queue_size=10, slop=0.5)
22          self.ts.registerCallback(self.callback)
23
24          self.pub = rospy.Publisher('/unibas_viewer/rgbd', Image, queue_size=1)
25
```

the slop parameter defines the delay (in seconds) with which messages can be synchronized



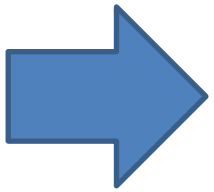
# Creiamo unibas_rgbd_reader.py

```
26    def callback(self, rgb_data, depth_data):
27
28      try:
29        cv_image = self.bridge.imgmsg_to_cv2(rgb_data, "bgr8")
30        depth_image = self.bridge.imgmsg_to_cv2(depth_data, "32FC1")
31        depth_array = np.array(depth_image, dtype=np.float32)
32        cv2.normalize(depth_array, depth_array, 0, 1, cv2.NORM_MINMAX)
33        depth_8 = (depth_array * 255).round().astype(np.uint8)
34        cv_depth = np.zeros_like(cv_image)
35        cv_depth[:,:,0] = depth_8
36        cv_depth[:,:,1] = depth_8
37        cv_depth[:,:,2] = depth_8
38
39      except CvBridgeError as e:
40        print(e)
41
42      rgbd = np.concatenate((cv_image, cv_depth), axis=1)
43
```



# Creiamo unibas_rgbd_reader.py

```
44        #cv2.imshow("res", res)
45        #cv2.waitKey(30)
46
47        #convert opencv format back to ros format and publish result
48        try:
49          rgbd_message = self.bridge.cv2_to_imgmsg(rgbd, "bgr8")
50          self.pub.publish(rgbd_message)
51        except CvBridgeError as e:
52          print(e)
53
54
```

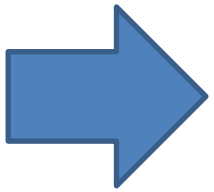


# Creiamo unibas_rgbd_reader.py

```
55
56  def main(args):
57    reader = unibas_rgbd_reader()
58    rospy.init_node('unibas_rgbd_reader', anonymous=True)
59    try:
60      rospy.spin()
61    except KeyboardInterrupt:
62      print("Shutting down")
63    cv2.destroyAllWindows()
64
65  if __name__ == '__main__':
66      main(sys.argv)
```

# Creiamo unibas_visualizer.py

```
1  #!/usr/bin/env python
2  from __future__ import print_function
3
4  import roslib
5  roslib.load_manifest('unibas_viewer')
6  import sys
7  import rospy
8  import cv2
9  from std_msgs.msg import String
10 from sensor_msgs.msg import Image
11 from cv_bridge import CvBridge, CvBridgeError
12
13 class unibas_visualizer:
14
15   def __init__(self):
16     self.bridge = CvBridge()
17     self.image_sub = rospy.Subscriber("/unibas_viewer/rgbd", Image, self.callback)
18
```



# Creiamo unibas_visualizer.py

```
19    def callback(self,data):
20      try:
21        cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
22      except CvBridgeError as e:
23        print(e)
24
25      cv2.imshow("rgbd", cv_image)
26      cv2.waitKey(30)
27
28
29  def main(args):
30    visualizer = unibas_visualizer()
31    rospy.init_node('unibas_viewer', anonymous=True)
32    try:
33      rospy.spin()
34    except KeyboardInterrupt:
35      print("Shutting down")
36    cv2.destroyAllWindows()
37
38  if __name__ == '__main__':
39      main(sys.argv)
```

# Permessi di esecuzione

Settiamo i permessi di esecuzione per i file

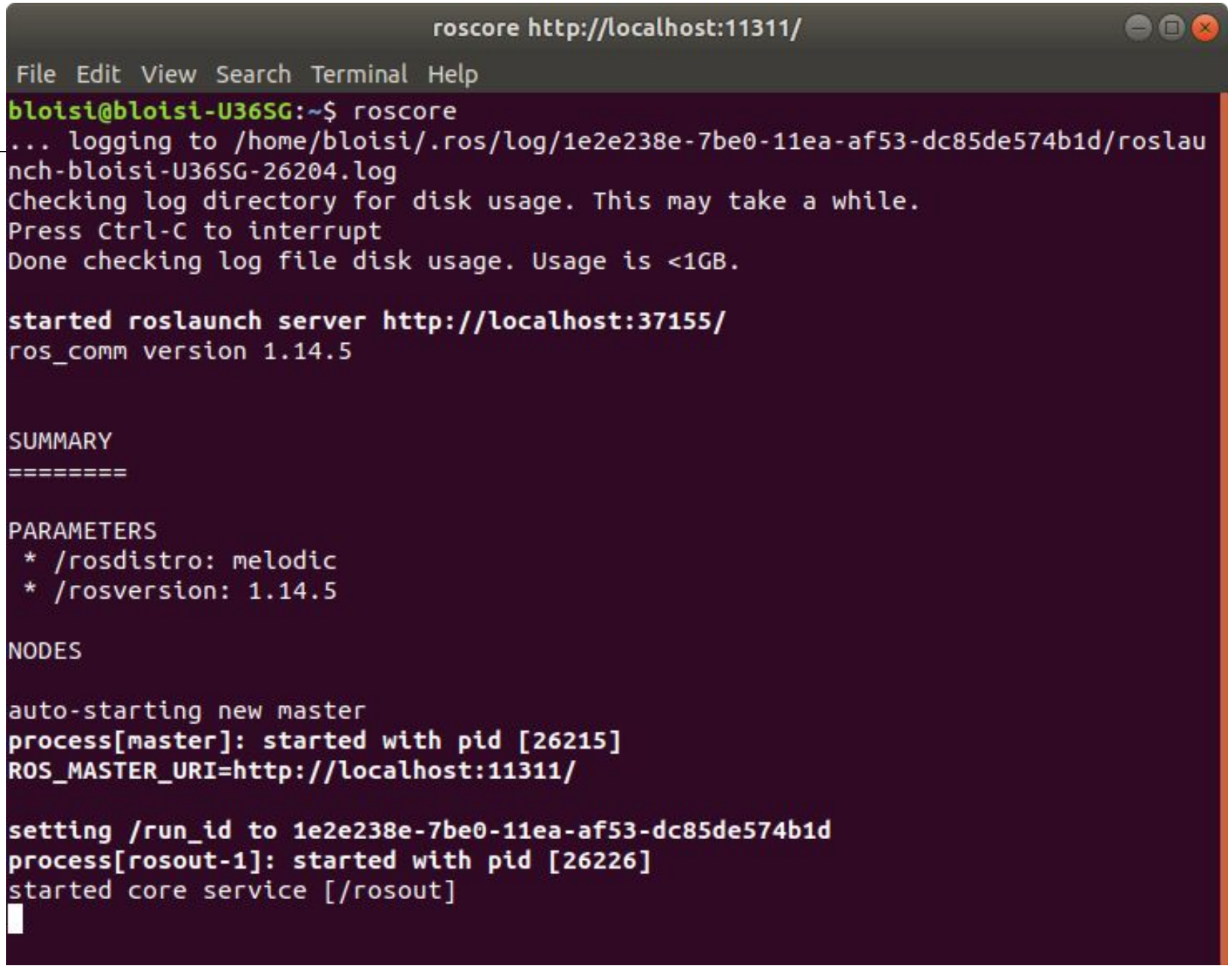
- unibas_rgbd_reader.py
- unibas_visualizer.py

con i comandi

```
chmod +x unibas_rgbd_reader.py
chmod +x unibas_visualizer.py
```

# roscore

Apriamo un primo terminal e lanciamo `roscore`



# rosrun

Apriamo un secondo terminal e lanciamo

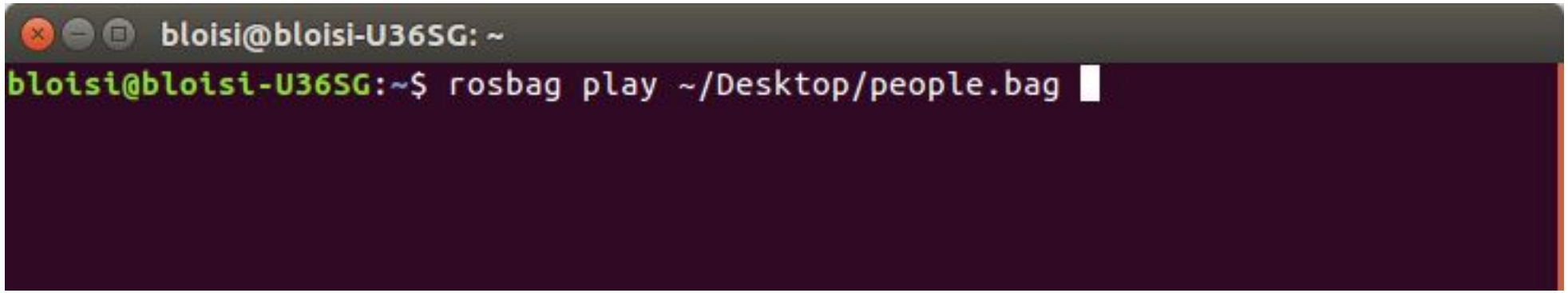
```
rosrun unibas_viewer unibas_rgbd_reader.py
```

# rosbag play

Apriamo un terzo terminal e lanciamo

```
rosbag play <path-to-bag>
```

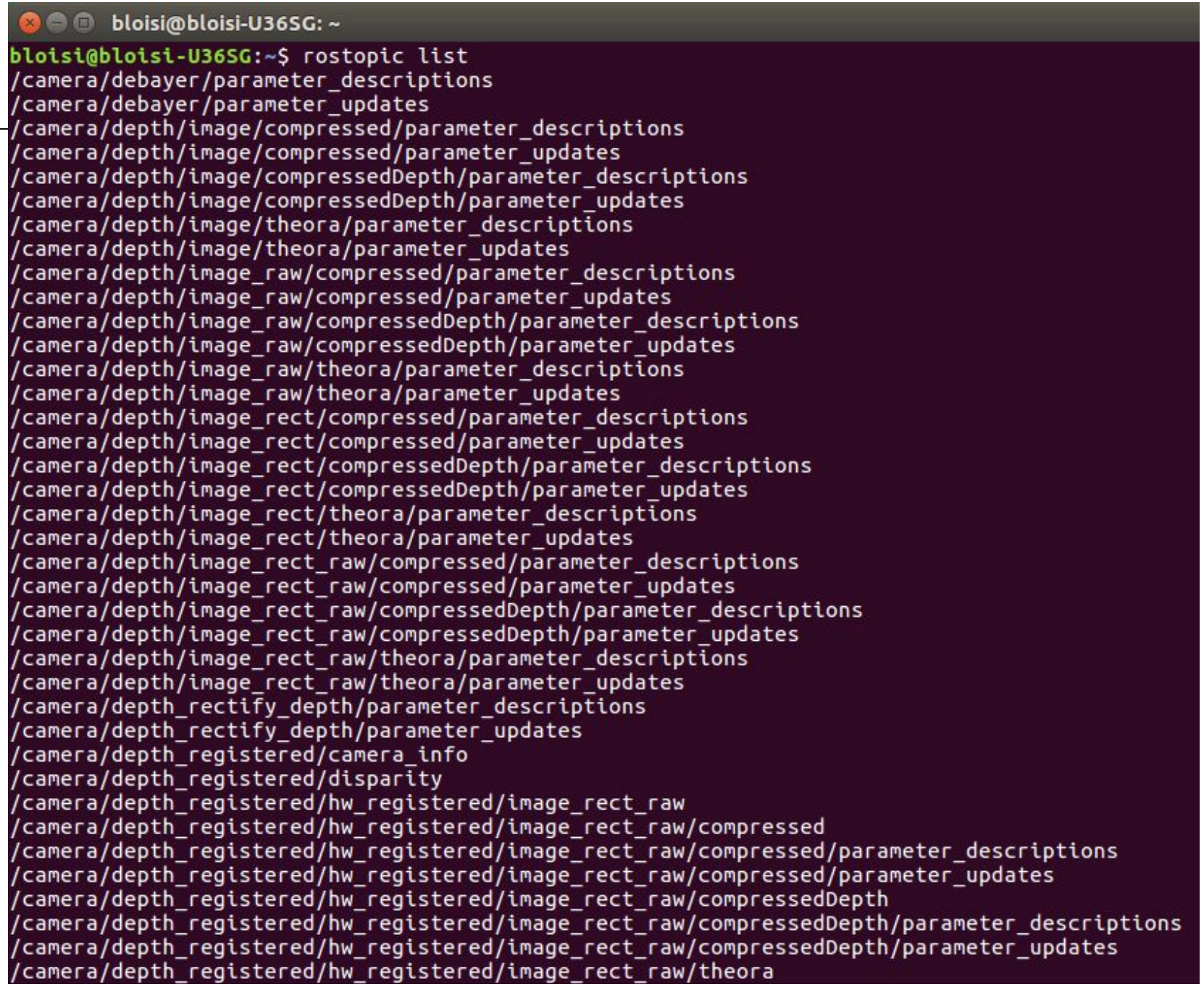
Per esempio:



# rostopic list

Apriamo un quarto terminal e lanciamo

```
rostopic list
```



# rostopic list

## continua...

```
bloisi@bloisi-U36SG: ~
/camera/depth_registered/hw_registered/image_rect_raw/theora/parameter_descriptions
/camera/depth_registered/hw_registered/image_rect_raw/theora/parameter_updates
/camera/depth_registered/image_raw
/camera/depth_registered/image_raw/compressed
/camera/depth_registered/image_raw/compressed/parameter_descriptions
/camera/depth_registered/image_raw/compressed/parameter_updates
/camera/depth_registered/image_raw/compressedDepth
/camera/depth_registered/image_raw/compressedDepth/parameter_descriptions
/camera/depth_registered/image_raw/compressedDepth/parameter_updates
/camera/depth_registered/image_raw/theora
/camera/depth_registered/image_raw/theora/parameter_descriptions
/camera/depth_registered/image_raw/theora/parameter_updates
/camera/depth_registered/points
/camera/depth_registered/sw_registered/image_rect_raw/compressed/parameter_descriptions
/camera/depth_registered/sw_registered/image_rect_raw/compressed/parameter_updates
/camera/depth_registered/sw_registered/image_rect_raw/compressedDepth/parameter_descriptions
/camera/depth_registered/sw_registered/image_rect_raw/compressedDepth/parameter_updates
/camera/depth_registered/sw_registered/image_rect_raw/theora/parameter_descriptions
/camera/depth_registered/sw_registered/image_rect_raw/theora/parameter_updates
/camera/depth_registered_rectify_depth/parameter_descriptions
/camera/depth_registered_rectify_depth/parameter_updates
/camera/driver/parameter_descriptions
/camera/driver/parameter_updates
/camera/ir/image_raw/compressed/parameter_descriptions
/camera/ir/image_raw/compressed/parameter_updates
/camera/ir/image_raw/compressedDepth/parameter_descriptions
/camera/ir/image_raw/compressedDepth/parameter_updates
/camera/ir/image_raw/theora/parameter_descriptions
/camera/ir/image_raw/theora/parameter_updates
/camera/ir/image_rect_ir/compressed/parameter_descriptions
/camera/ir/image_rect_ir/compressed/parameter_updates
/camera/ir/image_rect_ir/compressedDepth/parameter_descriptions
/camera/ir/image_rect_ir/compressedDepth/parameter_updates
/camera/ir/image_rect_ir/theora/parameter_descriptions
/camera/ir/image_rect_ir/theora/parameter_updates
/camera/projector/camera_info
/camera/rectify_color/parameter_descriptions
/camera/rectify_color/parameter_updates
/camera/rectify_ir/parameter_descriptions
```

# rostopic list

Abbiamo tantissimi topic!

```
bloisi@bloisi-U36SG: ~
/camera/rgb/image_color/theora/parameter_updates
/camera/rgb/image_mono
/camera/rgb/image_mono/compressed
/camera/rgb/image_mono/compressed/parameter_descriptions
/camera/rgb/image_mono/compressed/parameter_updates
/camera/rgb/image_mono/compressedDepth/parameter_descriptions
/camera/rgb/image_mono/compressedDepth/parameter_updates
/camera/rgb/image_mono/theora
/camera/rgb/image_mono/theora/parameter_descriptions
/camera/rgb/image_mono/theora/parameter_updates
/camera/rgb/image_raw
/camera/rgb/image_raw/compressed
/camera/rgb/image_raw/compressed/parameter_descriptions
/camera/rgb/image_raw/compressed/parameter_updates
/camera/rgb/image_raw/compressedDepth/parameter_descriptions
/camera/rgb/image_raw/compressedDepth/parameter_updates
/camera/rgb/image_raw/theora
/camera/rgb/image_raw/theora/parameter_descriptions
/camera/rgb/image_raw/theora/parameter_updates
/camera/rgb/image_rect_color
/camera/rgb/image_rect_color/compressed
/camera/rgb/image_rect_color/compressed/parameter_descriptions
/camera/rgb/image_rect_color/compressed/parameter_updates
/camera/rgb/image_rect_color/compressedDepth/parameter_descriptions
/camera/rgb/image_rect_color/compressedDepth/parameter_updates
/camera/rgb/image_rect_color/theora
/camera/rgb/image_rect_color/theora/parameter_descriptions
/camera/rgb/image_rect_color/theora/parameter_updates
/camera/rgb/image_rect_mono
/camera/rgb/image_rect_mono/compressed
/camera/rgb/image_rect_mono/compressed/parameter_descriptions
/camera/rgb/image_rect_mono/compressed/parameter_updates
/camera/rgb/image_rect_mono/compressedDepth/parameter_descriptions
/camera/rgb/image_rect_mono/compressedDepth/parameter_updates
/camera/rgb/image_rect_mono/theora
/camera/rgb/image_rect_mono/theora/parameter_descriptions
/camera/rgb/image_rect_mono/theora/parameter_updates
/clock
```

# Visualizzazione

Apriamo un quinto terminal e lanciamo

```
rosrun unibas_viewer unibas_visualizer.py
```

# Repository unibas_viewer

## unibas_viewer

ROS package for reading RGBD data

This code is part of the material of the course Computer Vision and Machine Perception - University of Basilicata (Italy)

**This code is provided without any warranty about its usability. It is for educational purposes and should be regarded as such.**

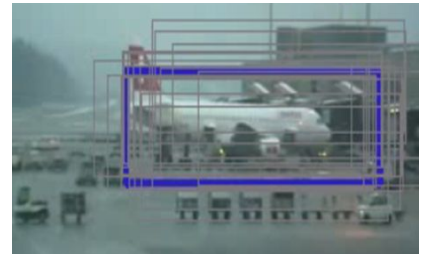

https://github.com/dbloisi/unibas_viewer



**UNIVERSITÀ DEGLI STUDI DELLA BASILICATA**

*Corso di Visione e Percezione*

Docente
Domenico D. Bloisi

# ROS+OpenCV