IIC2685 - Robótica Móvil
I – 2022

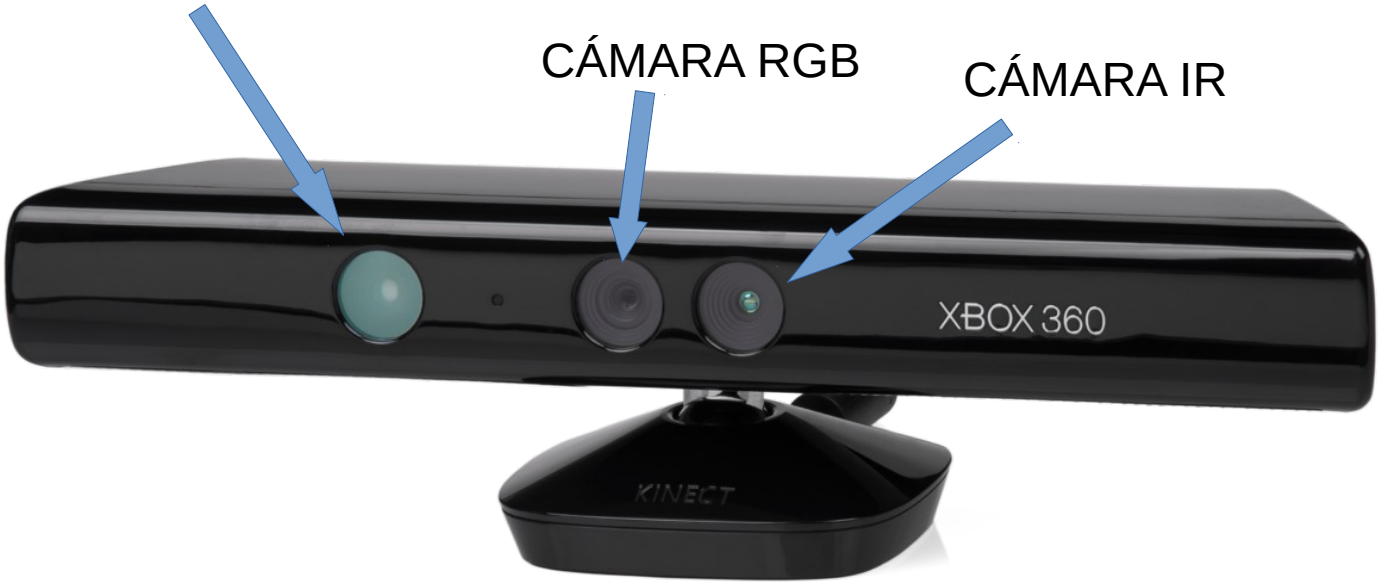
Clase práctica 2

Kinect

Profesor: Gabriel Sepúlveda V.
grsepulveda@uc.cl

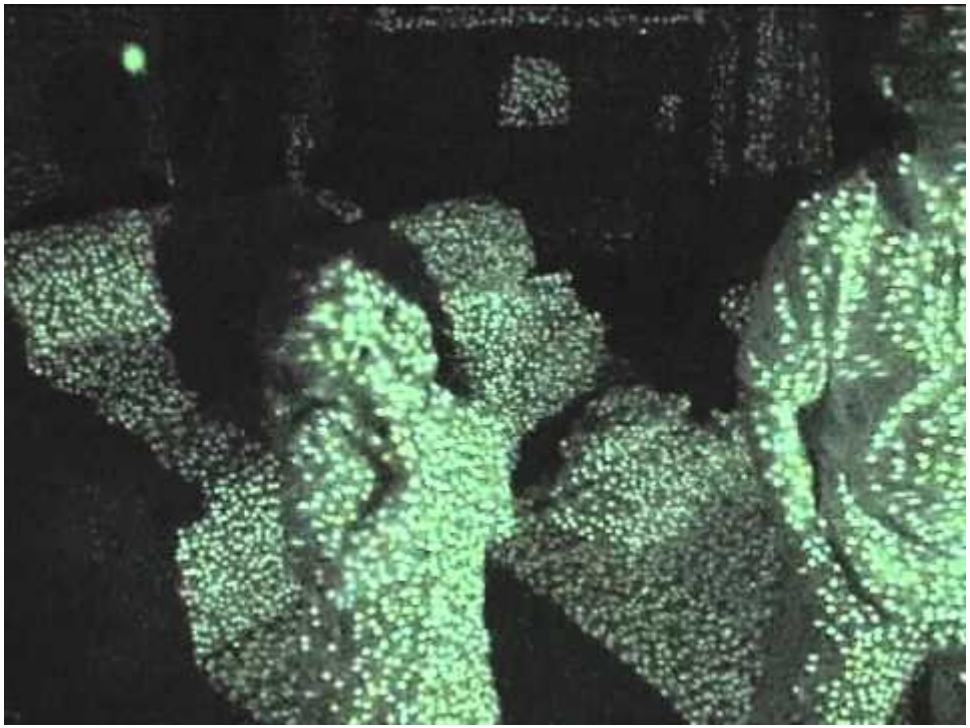
Kinect

EMISOR LUZ ESTRUCTURADA IR



Kinect

- Depth: proyección de luz estructurada IR es capturada por cámara IR y procesada
- Procesamiento: análisis de intensidad, densidad, etc.
- Salida: profundidad estimada (eje z en Kinect)



Kinect

- Ojo con las sombras

Kinect

- Ojo con las sombras

  - Emisor y receptor en distinta posición = sombra a luz estructurada = sectores sin estimación de profundidad (con valor 0 o NaN)

Kinect

- En términos prácticos
  - Imágenes VGA: 640 pixeles (ancho) x 480 pixeles (alto)
  - Imagen de profundidad “cercana” a imagen RGB en términos de ajuste
  - 30 frames por segundo
  - En ROS
    - Imagen RGB: 24 bits (8 bits por color R,G,B), cada canal entre 0 y 255
    - Imagen Depth: 16 bits, un canal, profundidad en milimetros
  - Medición de profundidad funciona en >= 45 [cm] (unos 15 [cm] del robot)
  - Ángulo de visión de cámaras (FOV): ~57° horizontal, ~43° vertical
    - Sirve para estimar (x, y) a partir de z, el punto (x_{pix}, y_{pix}) y la resolución de la imagen

OpenCV

- Imágenes son arreglos multidimensionales
  - Imagen RGB: (480, 640, 3)
  - Imagen Depth: (480, 640)
  - Pixel (0, 0): esquina superior izquierda de la imagen

What We See

What Computers See

Kinect y ROS

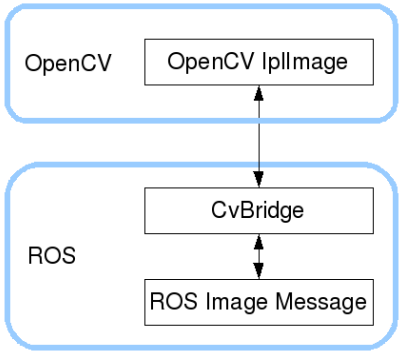
- En ROS: paquete openni_launch

```
$ roslaunch openni_launch openni.launch
```

Para visualizar rápidamente

```
$ rosrun rqt_image_view rqt_image_view
```

- Tópicos importantes
  - `/camera/rgb/image_color`
  - `/camera/depth/image_raw`
- Los mensajes de imágenes son del tipo sensor_msgs/Image
  - Para convertirlas a OpenCV: cv_bridge



Kinect y ROS: ejemplo de captura

```
import rospy
import roslib
import numpy
from sensor_msgs.msg import Image
import cv2
from cv_bridge import CvBridge

class Turtlebot_Kinect( object ):
  def __init__( self ):
    self.depth_sub = rospy.Subscriber( '/camera/depth/image', Image, self.depth_cb )
    self.rgb_sub = rospy.Subscriber( '/camera/rgb/image_color', Image, self.rgb_cb )
    self.bridge = CvBridge()
    self.current_cv_depth_image = None
    self.current_cv_rgb_image = None

  def depth_cb( self, data ):
        self.current_cv_depth_image = self.bridge.imgmsg_to_cv2( data )

  def rgb_cb( self, data ):
        self.current_cv_rgb_image = self.bridge.imgmsg_to_cv2( data )

if __name__ == '__main__':
    rospy.init_node( 'test_kinect' )
    handler = Turtlebot_Kinect()
    rospy.spin()
```

Bibliografía

- *Digital Image Processing*, Gonzalez, R. and Woods, R.
- Learning OpenCV, Bradski, G. and Kaehler, A.
- https://opencv.org/
- https://numpy.org/