TFG: CON BRAZO ROBOTICO UR3e:

He comenzado con el guion de practicas del lab de robótica. Para entender y adaptare al robot que vamos a utilizar para la realización del tfg.

En el inicio para comunicarnos con el robot estamos utilizando una Tablet con el software de ur ya conectada con el robot, desde la cual se pueden editar los parámetros de instalación del robot, moverlo y crear programas; Es decir, se puede hacer de todo pero dentro de las limitaciones del software ur.

Comenzamos con la instalazacion de la pinza, el calculo del centro de masas y centro de la herramientas, a partir de ahora llamada TCP (Tool center point).

Después de ello probamos a realizar algunos movimientos desde el apartado mover robot.

A continuación creamos nuestro primer programa. Despde un punto inicial programado se ejecuta el programa (programas movimientos) y se van realizando de forma secuencial.

Poco a poco fuimos aprendiendo funciones como el control de entradas y salidas, abrir y cerrar la pinza, el modo fuerza, la creación de variables y extraer datos de funciones dadas por el robot como la posición actual del TCP respecto de la base.

Una vez adquiridos estos conceptos y su simple manejo en programas específicos decidimos que era hora de realizar la ultima tarea de las practicas, un programa pick and place.

Los mayores problemas que tuvimos en la implementación fue la rotación únicamente de la pinza, ya que era en movimiento relativo. Todavía no he aprendido a como mover un angulo determinado cuando se quiera, y el limite del suelo para saber si quedan piezas o no. Al principio no sabia como extraer la coordenada z del TCP y asignarla a una variable especifica. Ahora mismo se hacerlo con 2 variables, una para adquirir el vector de datos del TCP y la otra para coger únicamente el [2] del vector, es decir, la pos de la cooodenada z.

Todavía tenemos que coger más nivel con el uso de funciones, manejo de scripts, condiciones, subrutinas y subprogramas, pero creo que ha llegado el momento de pasar al control remoto e ir manejando el robot desde un ordenador externo.

18/12/2023

Primer día de intento de conexión con el robot. Gracias a la ayuda del tutor estamos apuntado en la dirección correcta desde el principio.

Para comunicarnos con el robot vamos a utilizar la interfaz de comunicación RTDE (Real-time-data-exchange), propia de universal robots.

La añadimos al proyecto con pip install url.

El repositorio tiene un archivo setup.py en el directorio principal que permite la instalación de la librería usando erl pip (lo definido anteriormente).

Obviamente hemos creado un proyecto en Python y generado el entorno virtual

De momento hemos conseguido conectarnos al robot habiendo programado en la Tablet del robot la red -> la ip estática, 102.168.10.222 ; utilizamos los dns de la uni y mascara /24

Estamos utilizando y entendiendo el programa de ejemplo record.py para leer los datos del robot a tiempo real desde el programa Python. Ya hemos hechos algunos ajustes.

De momento se escriben todods los datos de los registros seleccionados en el XML en un csv cada poco tiempo

El objetivo del próximo día es conseguir extraer datos específicos como variables dentro de python

29/01/2024

Llevo 2 dias intentando mover el robot con el código de ejemplo.

No he intentado conectarlo en control remoto pero no tiene sentido para lo que quiero hacer yo ya que perdemos todo el software del robot y dependería únicamente del PC.

Por ello nosotros vamos a mover el robot utilizando registros. Debemos escribir en los registros desde el programa en Python y luego hacer un programa en la Tablet que se encarge de traducir esos registros en movimientos, velocidades, posiciones o lo que queramos.

Por tanto debe estar corriendo el programa en la Tablet y después pondermos a funcionar el script de Python para que ambos funcionen en conjunto.

Lo hemos conseguido.

Después de eso ha venido el tutor y hemos pensado una aplicación para el robot. Hacer seguimiento facial se queda corto por lo que de momento me ha dado una cámara (Luxonix OAK-D lite) que tiene 3 camaras. Se le pueden programar models AI, tiene acelerómetro, puede ver en 3d, etc para que vaya aprendiendo a manejarla durante esta semana con scripts de Python.

14/2/2024

He estado varias semanas aprendiendo a manejar la cámara desde un programa en Python.

He utilizado la librería Depth ai. También la sdk\_camera pero era tan normativa que solo se podía hacer lo que ellos tenían pro lo que la descarte. Estuve varios días aprendiendo openVINO, software para la gestión, generación, etc de redes neuronales artificiales aunque no me ha servido para nada del proyecto. Al final he conseguido cambiar el modelo de redes a uno para la detección de la cara y con el control del stereo y objectTraker he conseguido saber la posición de la cara. El objetivo es aprender a cargar cualquier red neuronal y saber la posición del objeto detectado. También quiero probar a crear una red neuronal propia. De momento son preentrenadas. También se utiliza el tipo de archivo blob pero no se como generarlo si tengo el xml, el yml o el bin.

19/2/2924

Esta semana voy a trabajar únicamente con el concepto de red neuronal. Tengo que aprender que tipos de redes hay y como funcionan. He comenzado hace¡iendo un ejemplo sencillo de una nn secuencial. Estoy utilizando la librería keras como framework para construir la red, entrenarla y predecir. Keras utiliza tensorflow en el backend. Hay otra librería que es pytorch que no he utilizado tyodavia. Voy a utilizar matplotlib para la ¡s graficas y numpy para el tratamiento de arrays, generar randoms, etc (direccionado a los datos). Tengo como objetivo aprender las librerías, enender los modelos y saber exportarlos e importarlos (de momento se el .json (configuración de la red) .keras(tada la info) .h5(info de los pesos)). Tengo como objetivo coger una red entrenada con objetos y que aprenda los mios. A este concepto se llama Transfer learning. También quiero a prender a exportar la red como tipo de archivo .blob y .bin

22/2/2024

De momento he decidido utilizar el framewor y red neuronal YOLOv8 DE ultralytics para preentrenar el modelo. Encuanto a la generación de imágenes etiquetadas utilizare roboflow. Me estoy dando cuenta que para entrenar modelos se suele hacer en un notebool de Google colab ya que Google de ‘presta’ una maquina virtual para wue hagas uso de sus recursos. Es un mix con jupiter. Voy a tener que aprender Pytorch porque yolo la utiliza. El objetivo es entrenar la red de yolo para que detecte mis figuras.

28/2/2024

Ultimo día antes de las vacaciones. No me acuerdo que hice pero ya reconocia mis objetos desde la cámara utilizando la YOLO entrenada

14/3/2024

Vuelta de vacaciones. Empezamos haciendo un pull del repositorio en mi pc porque no hay internet por cable en el laboratorio. Después empezamos a instalar las dependencias necesarias para poder correr los programas en el PC. Aparte de esto hay que instalar el software especifico para reconocer la cámara -> luxonix Depthai para Windows.

Dependencias de Python:

Numpy: tratamiento de vectores y números. Utilizando por otras librerias

Urrtde -> para la conexión con el robot. Se descarga desde el repositorio de github

Opencv-python -> para tratamiento y visualización de imágenes

Depthai -> para el control de la cámara OAK-D Little de luxonix

Blobconverter -> para la conversión de las redes neuronales en otros formatos (Caffe, openVIno, Tensor, etc)

Ultralytics -> para entrenar mi propia red neuroal YOLO v8

Panteamos el primer programa global:

Tenemos piezas de diferentes formas (circulo, cuadrado, hexágono, estrella, irregular)

Debemos cogerlas con el robot y colocarlas en el hoyo que corresponde a cada pieza

V1:

Conocemos las posiciones de recogida y las de los hoyos.

No conocemos que pieza esta en cada sitio en la posición inicial por lo que utilizamos la cámara para saber el orden de recogida.

V2:

Piezas en posiciones iniciales desconocidas pero siempre colocadas de la misma manera (apuntando hacia arriba)

V3:

Piezas en posiciones iniciales desconocidas y colocadas de distinta manera e.j. tumbadas, giradas, etc.

V4:

Utilizamos hoyos mas pequeñoas -> menor porcentaje de error

Hoy hemos comenzando haciendo el reconocimiento de las figuras con la red neuronal, permitnendo que detectev mas de una y te diga que pieza es.

Mañana veremos como decir en que orden van (de izquierda a derecha)

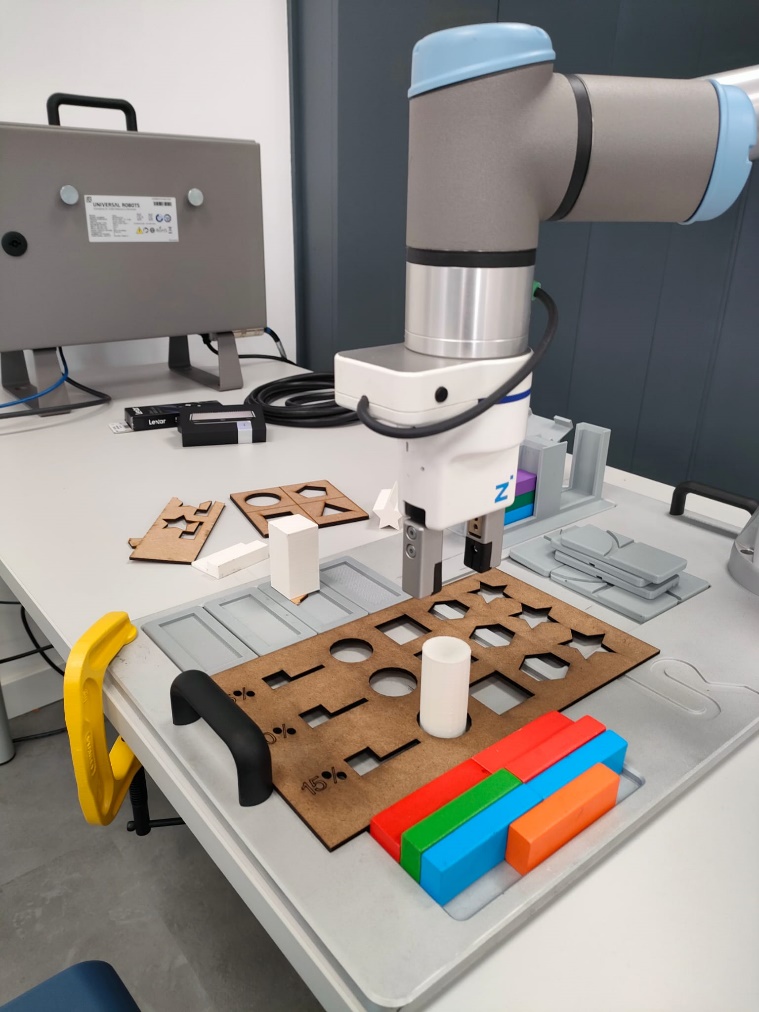
Posteriormente daremos las instrucciones al robot parea que las coja y las deje en el hoyo que toque. Siempre cogiendo las piezas de izquierda a derecha

Es necesario tener los puntos exactos de donde se cogen las piezas y los puntos de cada hoyo sabiendo a que pieza pertenece.

21/03/2024

Hoy se ha acabado la primera versión funcional del tfg.

Como seguramente explicamos anteriormente las piezas se encuentran en posiciones peestablecidas, pero no sabemos que pieza esta en cada posición por lo que utilizamos la cámara y la red neuronal peentranada para que diferencie que piezas hay en las posiciones. Una vez tenemos el orden de las piezas (ej: en la posicion1 se encuentra el circulo y en la posición 2 se encuentra el cuadrado) nos conectamos al robot y se realizan todos los movimientos de manera secuencial. Empezando por la primera pieza y terminando por la ultima. Nosotros tenemos también fijos los hoyos donde hay que introducir las piezas. Por lo que sabiendo que pieza es podemos cogerla y llevara al hoyo correspondiente sin ningún problema.



PLANTEAMIENTO DE LA VERSION 2:

Las piezas se encuentran en un espacio conocido pero no sabemos su posición exacta, Por lo que es necesario que la cámara mida profundidades y se encuentre fija (No como en la versión 1 (No era necesario porque solo necesitábamos el orden)) para así poder calcular la posición relativa de la pieza respecto del robot con matemática.

Las piezas se encontrarán siempre orientadas de la misma manera (boca arriba) y apuntando en la misma dirección.

Manualidades:

1. Fijación de base de hoyos que sea no cutre
2. Estructura de soporte de la cámara enlazada al chasis del robot y a la base de los hoyos

Necesito:

1. Cable usb3 -usbc largo para conectar el ordenador con la cámara. El de ahora es muy corto y no permite maniobrabilidad
2. Vamos a utilizar de momento solo 2 piezas (círculo y cuadrado)
3. No es necesario, pero deberíamos entrenar la red con más fotos

9/04/2024

Ha habido un cambio de planes. Empezamos la semana con la intención de usar el stereo de la cámara para sacar la matriz de disparidad y con ello una imagen de profundidad (escala de grises según la distancia a la cámara). Con esto sacaríamos los puntos en 3d respecto a un eje predefinido.

Hoy ha habido reunión con el tutor y me ha explicado las nubes de puntos, por lo que la intención ahora es:

1. Con Depthai y open3d adquirimos una nube de puntos. En esa nube hay que encontrar la pieza. Si encontramos la pieza ya conoceremos sus coordenadas 3d respecto al sistema de referencia de la cámara. Una vez tenemos la pieza detectada hay que ver que orientación tiene. Eso se logra gracias a Object pose estimation, y el registro del objeto (que esta en otra nube de puntos predefinida). Estas ‘encajan’ para así ver cuanto se ha movido la cámara.
2. Una vez conocida la posición del objeto respecto de la cámara faltaría la posición de la cámara respecto de la base del robot.
3. Utilizaremos una librería llamada apriltag. Lo que hace es reconocer un QR en la imagen. Ese QR debe estar situado en un lugar donde conozcamos su posición respecto de la base del robot. La librería lo que hace es sacar la posición de la cámara respecto del apriltag (QR ).
4. Una vez conocidas todas las posiciones procedemos a generar la matriz homogénea para hacer las transformaciones de ejes.
5. Con la matriz podríamos pasar cualquier punto de la nube de puntos al sistema de referencia del robot (base) con una simple operación.

Con todo eso ya tendríamos la posición del objeto para poder cogerlo y llevarlo a su agujero.

18/04/2024

Durante esta semana hemos aprendido a utilizar la librería open3d para la visualizacion y el trabajo con nubes de puntos.

Con la libreria depthai adquirimos la imagenRGB para la deteccion de los objetos con la red neuronal (adquirimos los pixeles del recuadro del objeto) y tambien adquirimos la nube de puntos correspondiente.

Con el objeto encuadrado en la imagen 2d solo nos falta adquirir los puntos correspondientes de la nube de puntos y eliminar todos los demas (recorte de la nube de puntos). Sabemos recortar pero no nos coinciden los pixeles de la imagen con los de la nube de puntos poruqe no sabemos la relacion que hay.

Por otro lado ya hemos adquirido la matriz de transformacion para saber la posicion de la camara respecto del apriltag. Hemos utilizado la libreria apriltag para reducir calculos. Ha sido necesario saber las dimensiones del recuadro y los parametros especificos de la camara (distancia focal y resolucion) para poder adquirir la matriz.

Cuando resolvamos el corte de pixeles de 2d en 3d nos pondremos con el pose estimation para saber la rotacion del objeto.

Ya sabemos cortar la nube según los pixeles de la imagen (matriz de la resolución coincidiente con la matriz de la nube de puntos). Nos falta ver como cortar en el eje z.

OJETIVOS DE LA PRÓXIMA SEMANA:

1. Nube de puntos cortada perfectamente por sus 3 ejes (de momento solo x e y)
2. Matris de transformación del apriltag respecto de la base del robot (april tag pegado en un sitio conocido para que la matriz sea lo mas sencilla posible -> ejes alienados -> solo traslación de x e y -> apriltag pegado en la base -> z = 0)
3. Probar la matriz de traslación llevando el robot al centro del apriltag habiendo puesto de coordenadas las (0,0,0). Obviamente tiene que haber una transformación.
4. Pegar a cada objeto un apriltag en su tejado para poder conocer la pose del objeto y sus coordenadas respecto al apriltag BASE.
5. Con esto tendríamos la segunda versión del tfg funcional

OBJETIVOS FINALES:

1. Entrenar la red neuronal para que detecte la pose del objeto
2. Sustituir el apriltag de los objetos por la pose stimation de la red neuronal
3. Pose estimation con la nube de puntos y un modelo 3d conocido