Manual de uso

Título TFG	Detección de heridas en imágenes RGB-D integrado en un sistema ciberfísico para la asistencia a la sutura laparoscópica.
Autora	Duna de Luis Moura
Tutora	Dra. Isabel García Morales
CoTutor	Álvaro Galán Cuenca
Fecha	2022-2023

El presente documento se ha desarrollado con el fin de elaborar un manual de instalación y utilización de la aplicación llevada a cabo en el trabajo de fin de grado comentado, partiendo desde un sistema limpio, es decir, sin nada pre-instalado.

Configuración hardware

Aunque resulte obvio, lo primero es comprobar que la cámara está correctamente conectada al ordenador mediante el cable USB correspondiente. Además, se debe cerciorar de que no hay ninguna máquina virtual en ejecución en el ordenador ya que en este caso los datos de la cámara irán a dicha máquina en caso de tenerlo así configurado.

Los ángulos articulares del brazo robótico que sujeta la cámara se han configurado a los siguientes valores en grados: [61.81, -166.95, -57.27, -55.81, 66.27, 18.10]. Además, la cámara se ha colocado con respecto al efector final del robot que la sujeta cumpliendo la siguiente matriz de transformación homogénea:

$$^{E}T_{Cam} = desp([4.1,\ 0,\ 5.63])\ *\ rotZ([-\ \pi/2])$$

Estos valores se deben a las medidas del soporte de la cámara y a su orientación en el mismo.

Configuración software

En primer lugar, este programa está pensado para ser utilizado en **Ubuntu 20.04**, con **Python 3** y **ROS Noetic** y se ha trabajado desde el IDE Visual Studio Code. Sobre esta base, se clonará el repositorio git (aquí) donde se encuentra almacenado el proyecto completo y se configurará el entorno de trabajo para que cuente con todas las

librerías Python requeridas. Para evitar conflictos entre distintas versiones de paquetes en la misma máquina se creará un entorno virtual con los pasos que se muestran a continuación:

1. Instalar pip con: `sudo apt install python3-pip`
2. Instalamos con pip el entorno virtual ejecutando: `sudo pip3 install virtualenv`
3. Crear el entorno virtual: `python3 -m venv venv`

De esta forma, para activar el entorno virtual se ejecuta el comando `source venv/bin/activate` y para desactivar `deactivate`. En el fichero requirements.txt se encuentran almacenadas la lista de librerías necesarias para la ejecución de la aplicación. Para instalarlas ejecutamos `pip install -r requirements.txt`. Con esto ya estaría el entorno virtual completo con todas las librerías necesarias (como las librerías se pueden ir actualizando y cambiando de nombre si falta alguna se instalará ejecutado `pip install`).

A continuación, se debe comentar que hay dos programas diferentes, uno para investigación y otro para la implantación con el sistema robótico. Ambas cumplen el objetivo de detectar la herida a partir de los datos de la cámara RGB-D, su clasificación entre plana y tubular y el cálculo de las coordenadas de los puntos de sutura y bias para generar la trayectoria. Sin embargo, en el de investigación se generan gráficas para visualizar distintos puntos del algoritmo y se muestra por terminal una serie de información relevante. Mientras que el de implantación se ejecuta directamente en un nodo ROS y envía las coordenadas calculadas a un servidor ROS.

El primer paso en ambos modos es modificar la dirección donde se encuentra almacenado el proyecto. Esto se guarda en la variable `dir` justo después de importar todas las librerías.

Programa para investigación

Este modo permite visualizar las distintas fases de la ejecución mediante gráficas 2D y 3D relevantes. Para utilizarlo se ejecuta el fichero main.py. Para ir saltando cada una de las figuras se pulsará la tecla ‘q’ para evitar errores. Todas las gráficas que se muestran aparecerán con unidades en ambos ejes y un título descriptivo, y son:

1. Instantánea capturada por la cámara de la escena quirúrgica, con la imagen a color y el mapa de calor que representa la profundidad media con la leyenda asociada.
2. Gráfica 3D con los datos de profundidad sin procesar.
3. Gráfica 3D con los píxeles inválidos recortados.

4. Gráfica 3D con los píxeles nulos interpolados, valores en Z invertidos y suavizado gaussiano aplicado (se puede visualizar cada una de estas fases por separado descomentando las líneas que utilizan la función `utils.grid()`).
5. Gráfica con el gradiente 3D.
6. Gráfica con el gradiente 2D binarizado.
7. Imagen binaria con la región considerada como herida tras el procesamiento del gradiente binario.
8. Gráfica 3D con el ROI calculado a partir de la herida detectada.
9. Gráfica de dispersión del clasificador (está suele tardar un poco más dado que ejecuta el clasificador). Si da error habrá que cambiar la dirección indicada en la variable `dir` (debe apuntar al módulo M5_classifier_module dentro de la carpeta CameraFeature del proyecto).
10. Imagen 2D con los puntos de sutura representados en la diagonal principal de la herida.
11. Imagen 2D con los puntos de sutura representados en la línea de sutura (en caso de ser una herida gruesa).
12. Imagen 2D con la herida con sus coordenadas de puntos de sutura en la orientación original.
13. Imagen RGB con los puntos de sutura calculados.
14. Trayectoria 3D con los puntos de sutura y puntos bias en metros en el sistema de referencia de la base del robot encargado de la sutura.
15. De nuevo, la imagen RGB con los puntos de sutura calculados.

Además, durante la ejecución se mostrarán una serie de datos por la consola como el ancho y alto de la imagen capturada en píxeles, la distancia entre la cámara y la superficie en centímetros, la escala de píxel calculada para esa imagen, el porcentaje de píxeles nulos de la imagen que serán interpolados, el número de contornos detectados en el gradiente binario, si se trata de una herida plana o tubular, la orientación original de la herida, las coordenadas en píxeles de los extremos de la herida, las coordenadas de los puntos de sutura en la diagonal principal en píxeles y si la herida es gruesa se muestran también los calculados en la línea de sutura, los píxeles en la orientación original, los píxeles en centímetros con las 3 coordenadas en el sistema de referencia de la cámara, los puntos de sutura respecto de la base del robot en metros y las coordenadas de la trayectoria final completa.

A lo largo del código hay muchas más gráficas y mensajes comentados que no se mostrarán, pero se han mantenido estos al considerarse los más relevantes.

Programa para implantación

En primer lugar, es necesario tener correctamente instalado ROS Noetic previamente. Para utilizarlo se deben abrir 4 terminales dentro de la carpeta `M6_ros_service_module/catkin_ws`:

1. En la primera ejecutamos `roscore` para inicial el ros master.
2. En la segunda ejecutamos `rosrun camera_pkg camera_server.py` para iniciar el servidor de la cámara que almacenará las coordenadas de la trayectoria.
3. En la tercera ejecutamos `rosrun camera_pkg camera_client.py` para ejecutar la aplicación completa y enviar las coordenadas de los puntos al servidor.
4. Por último, ejecutamos `rosrun camera_pkg user_client.py` para recibir en una estructura tipo diccionario las coordenadas. La variable tiene la forma siguiente: {“num”: int64, “stitches”: float64[]}. Los puntos de sutura se envían en un solo array, para reconstruirlos es necesario reagruparlos de 3 en 3 para tener las coordenadas (x, y, z) de cada uno de los puntos (el total de puntos se almacena en la variable num para comprobar que no han habido errores). Las coordenadas están calculadas en metros en el sistema de referencia del robot encargado de realizar la sutura.