MEMORIA TÉCNICA 3

Nuevos Paradigmas de la Interacción

16 DE DICIEMBRE DE 2018

Óscar López Arcos
Óscar Jiménez Fernández
Ernesto Martínez del Pino

Esquemas de conexión

Para establecer la conexión con el dispositivo hemos seguido la documentación oficial del mismo que se puede encontrar en el siguiente enlace:

https://github.com/IntelRealSense/librealsense/blob/master/doc/distribution_linux.md

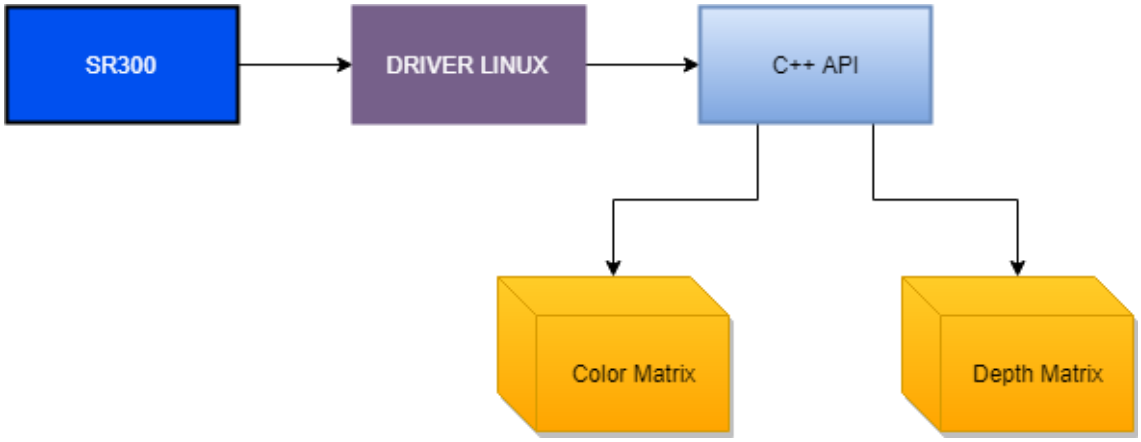
Esto nos ha permitido usar la API de programación implementada para $c++$. Con la que hemos obtenido la información necesaria aportada por la cámara para reconocer gestos. Los ejemplos en los que nos hemos basado para realizar esta práctica se encuentran en el siguiente enlace:

https://github.com/IntelRealSense/librealsense/tree/master/examples

Usando la API hemos extraído son conjuntos de datos:

- $Color\ Matrix$: Es la matriz que guarda la imagen color captada por la cámara.
- $Depth\ Matrix$: Es la matriz que guarda la profundidad de cada pixel.

Esquema para ilustrar:

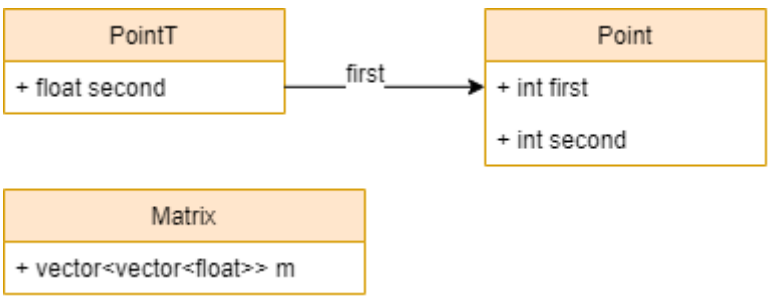


Esquemas de clases

Hemos implementado 3 clases para representar la información que extraemos de la cámara de Intel $RealSense$:

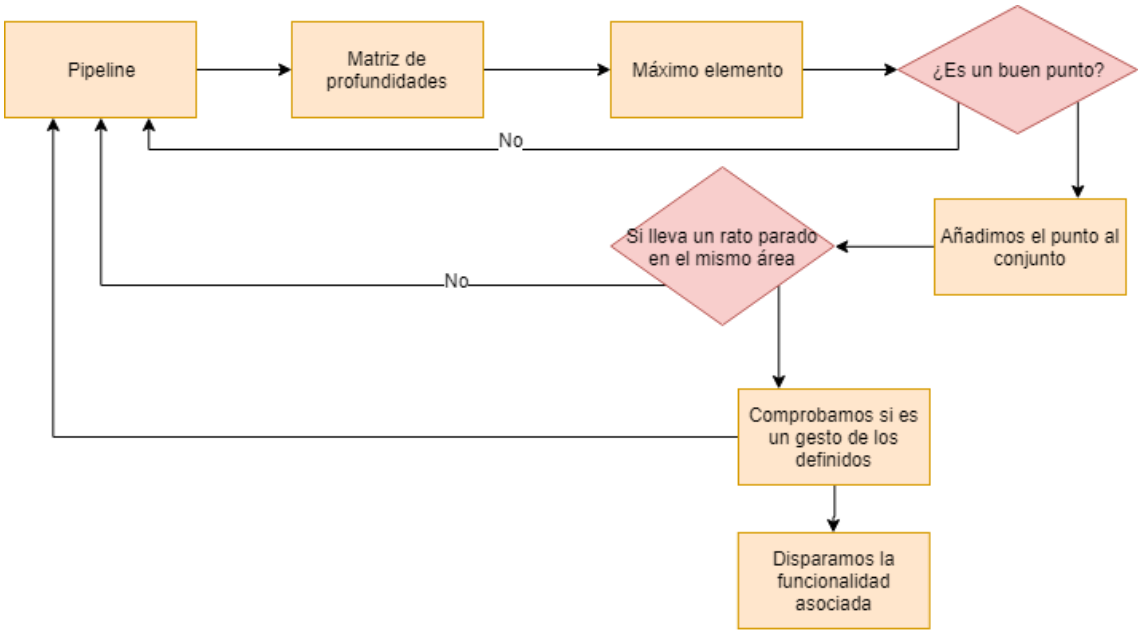
- $Point$: representa un punto en el espacio 2D.
- $PointT$: representa un punto en el tiempo en un espacio 2D.
- $Matrix$: representa la profundidad del plano.

El esquema de clases que representa nuestro proyecto es el siguiente:



Esquema de flujos

Este es el esquema de flujos que seguimos cuando recibimos la información:



Este es el esquema utilizado para seguir el movimiento de la mano es codificado mediante un punto.

Sabemos que es un buen punto cuando se encuentra a una distancia razonable del último punto almacenado. De cada punto se almacenan sus coordenadas y el momento en el que fue detectado.

Como almacenamos el tiempo de cada punto podemos inferir el intervalo de tiempo que lleva en un área. Esto es de gran utilidad para determinar si se ha completado un gesto o un subgesto.

Para comprobar si un gesto pertenece al conjunto de gestos definidos comprobamos su descomposición en gestos menores. En el siguiente apartado veremos cómo se define un gesto en nuestro sistema.

Definición de gestos

De forma nativa podemos codificar 4 gestos básicos:

- Movimiento izquierda
- Movimiento derecha
- Movimiento arriba
- ñMovimiento abajo

Para determinar un gesto hacemos uso de la siguiente estructura de datos ordenada en función del tiempo:

$$vector < PointT > gesturePath;$$

Ejemplo:

Si quisiéramos sabemos si la mano se mueve a la izquierda durante un periodo x de tiempo comprobamos si las coordenadas de inicio y fin se corresponde con ese movimiento. Posteriormente determinamos una probabilidad de aceptación de dicho movimiento con el número de puntos que su adyacente se mueve a la izquierda. Si dicha probabilidad supera un umbral fijado aceptamos el movimiento y devolvemos la distancia en ese eje recorrida.

Así es como definimos lo que llamamos gestos fáciles con lo que luego definimos componemos los gestos difíciles.

Nota: un error muy común en la industria de la interacción por gestos es fijar la posición o el escalado del gesto. En nuestro caso es completamente libre y solo limitada por las capacidades de la cámara.

Usando lo anteriormente dicho y dos propiedades más podemos definir los 4 gestos básicos más, que son los siguientes:

- Movimiento a la izquierda y abajo
- Movimiento a la izquierda y arriba
- Movimiento a la derecha y abajo
- Movimiento a la derecha y arriba

Las dos propiedades usadas son las siguientes:

- La media aritmética de las probabilidades supere un umbral fijado
- La distancia recorrida en las dos direcciones sea proporcional y no supere nunca el doble que la otra. Conclusión de un movimiento en diagonal.

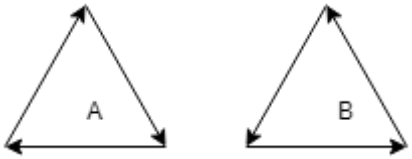
A continuación, se muestra la tabla de codificaciones de los movimientos básicos usada en la tabla:

Nombre	Codificación
Nulo	0
Derecha	1
Izquierda	2
Arriba	3
Abajo	4
Derecha Arriba	5
Izquierda Arriba	6
Derecha Abajo	7
Izquierda Abajo	8

Ahora tenemos las herramientas necesarias para definir una amplia gama de gestos. Para la práctica hemos definido 4 gestos relacionados con el movimiento que hace una mano.

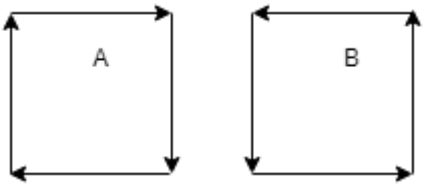
Para definir un gesto complejo solo es necesario dar el conjunto de codificaciones del mismo. Veamos los ejemplos de nuestra práctica:

Triángulo



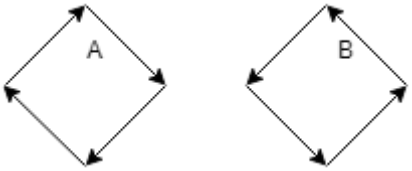
Nombre	Codificación
Triángulo A	725
Triángulo B	527

Cuadrado



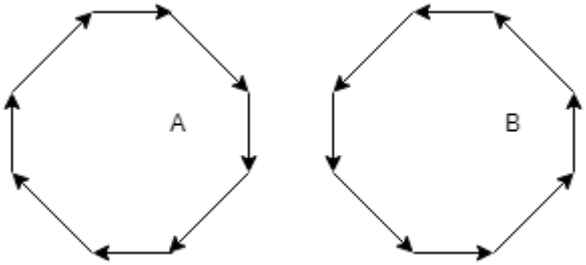
Nombre	Codificación
Cuadrado A	1423
Cuadrado B	4132

Rombo



Nombre	Codificación
Rombo A	7865
Rombo B	5687

Circulo



Nombre	Codificación
Circulo A	17482635
Circulo B	53628471

Parámetros libres

Nombre	Tipo de dato	Valor por defecto	Función
MAX_POINT	$unsigned\ int$	**1000**	**Número máximo de puntos que almacenamos**
MAX_DISTANCE	$float$	**200**	**Diferencia máxima que puede tener un punto nuevo con el anterior**
TIME_INTERVAL	$double$	**1.2**	**Tiempo entre gesto y gesto**
ERROR_SQUARE	$Point$	**(75,75)**	**Error máximo admitido en forma de cuadrado**
ERROR_CIRCLE	$float$	**45.0**	**Error máximo admitido en forma de circulo**
PERCENTAGE	$float$	**0.725**	**Portage mínimo de veracidad de un gesto para ser admitido**

Conclusión

En el desarrollo de la práctica hemos constatado que la $INTEL\ REALSENSE\ SDK\ 2.O$ presenta graves errores de implementación, por lo menos cuando se usa en conjunto con la cámara $SR300$. Los propios ejemplos de implementación realizados por la empresa $INTEL$ presenta errores que provocan cierres inesperados.

Realizamos numerosas pruebas siguiendo la documentación oficial del mismo:

https://software.intel.com/sites/landingpage/realsense/camera-sdk/v1.1/documentation/html/index.html?doc_devguide_introduction.html

Teniendo éxito a la hora de integrar la API y el motor gráfico $UNITY$ siendo capaz de iniciarse pero en cuanto hacía $matching$ con un gesto definido, $UNITY$ se cierra inesperadamente. También probamos ejemplo realizados por externos llegando al mismo resultado.

Puesto que se mencionó en clase que no podíamos tener todos los mismos dispositivos, viendo que mayoritariamente los grupos habían elegido los otros y la ocupación de los mismo decidimos continuar con el elegido.

Nos hubiera gustado explotar las capacidades a las que hemos visto que han llegado el resto de nuestros compañeros, implementando gestos muchísimo más complejos fruto de proyectos mucho más elaborados que el nuestro.

Ha sido un trabajo duro de diseño e ingeniería haber creado esta pequeña capa de abstracción con el objetivo de cumplir los requisitos de la práctica con este dispositivo.