Estudio de factibilidad: Detección de aguja

En este documento detallamos la investigación que realizamos sobre la posibilidad de detectar la aguja, el momento y su posición a la hora de punzar, y por qué decidimos dejarlo fuera del alcance del proyecto.

Contexto

Nuestra primer idea del simulador consistía en un muñeco y una Kinect. Esta Kinect se utilizaría para trackear las manos del practicante y la aguja utilizada durante la práctica.

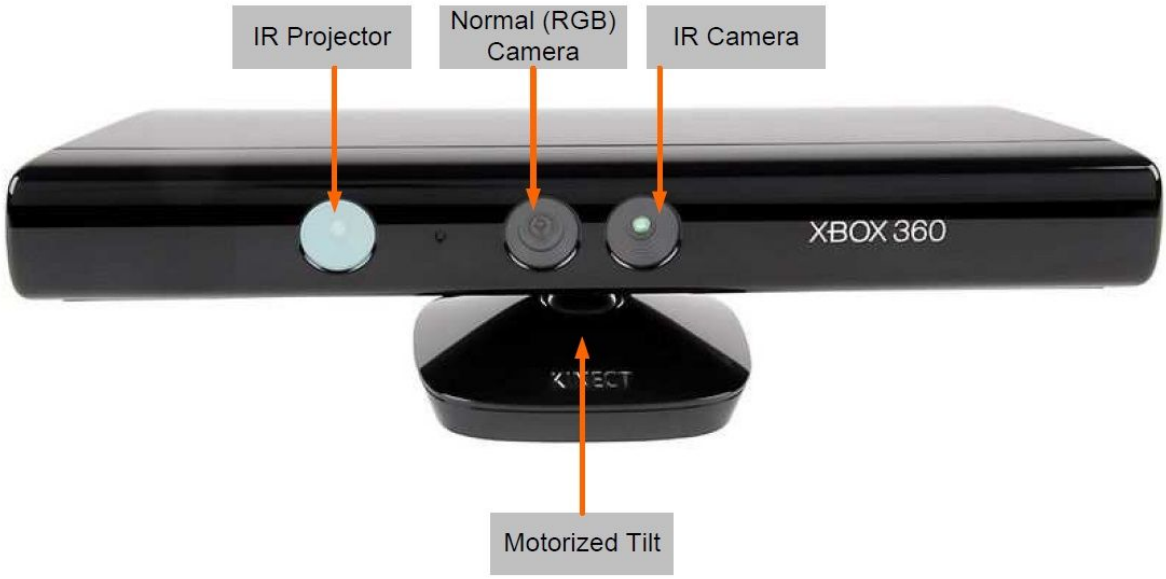
Sobre el tracking de las manos, esperábamos conseguir la posición de las manos en el espacio y en todo momento, definir un área de zona estéril e indicar si las manos se encuentran dentro o fuera de dicha zona, para luego armar las alertas sobre contaminación de la zona.

Sobre el tracking de la aguja esperábamos conseguir su modelo en tres dimensiones, a partir del cual podríamos obtener la posición y ángulo, deducir el momento en que comienza una punción e indicar si se encuentra o no en la posición y ángulo correctos.

Sobre la Kinect

La Kinect XBOX 360 cuenta con una cámara a color (RGB, 1280x960), una cámara infrarroja (IR, 640x480) y un proyector IR.

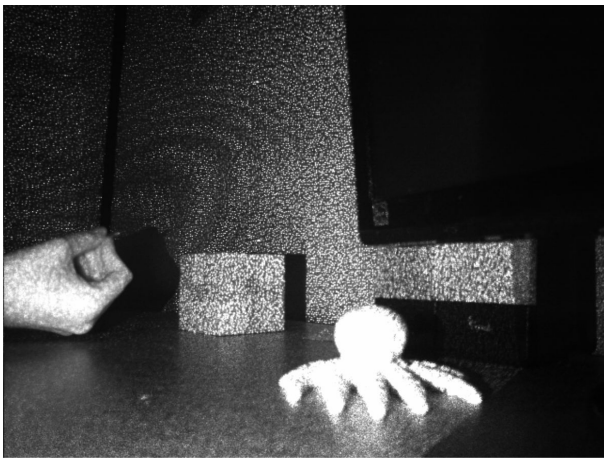
También cuenta con lo que se llama sensor de profundidad, que consiste en proyectar una grilla infrarroja con el proyector IR y verla con la cámara IR, la cual se encuentra convenientemente lejos del proyector y puede ver la grilla desde otro ángulo. La diferencia entre una grilla perfecta y la grilla que efectivamente vea la cámara indica la distancia en el eje Z (profundidad). Esto implica que, aunque tenemos 640x480 píxeles en la cámara IR, no tenemos esa definición en el eje Z, sino que es mucho menor.

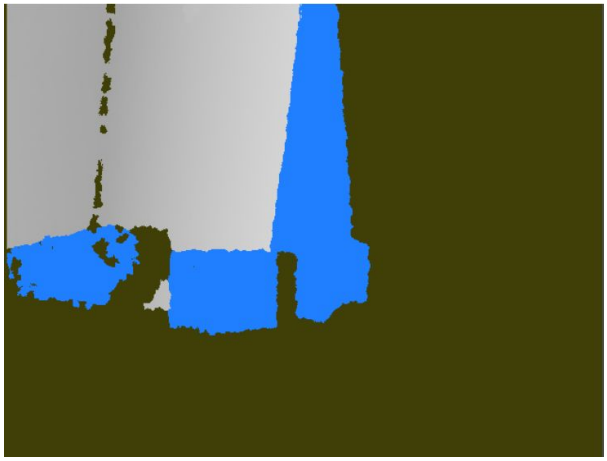


Nuestros intentos

Primera idea: detectar la aguja con la Kinect

Acá vemos, en orden, la cámara RGB, la cámara IR y el sensor de profundidad.



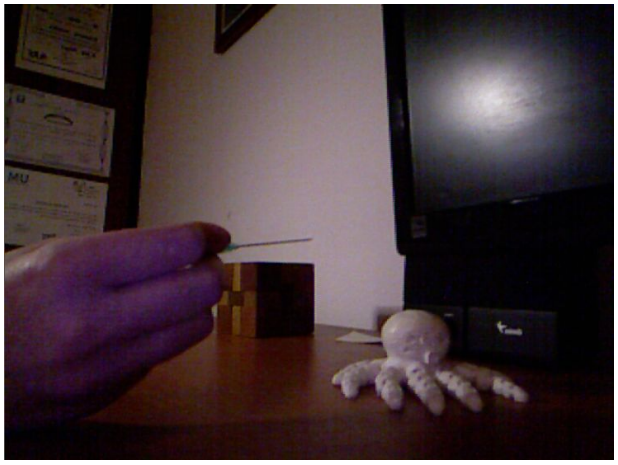


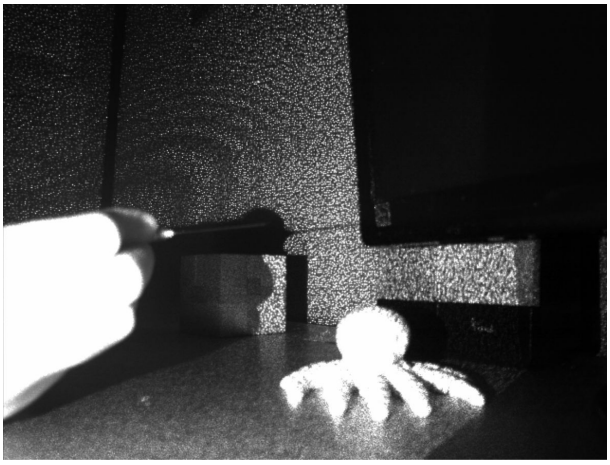
Como vemos, el sensor de profundidad no detecta cosas demasiado cerca. Por ejemplo, detecta al cubo de madera, pero no al pulpo blanco. Ubicamos la aguja a la altura del cubo, pero no la detecta. Esto se debe a que no cuenta con suficiente definición para encontrar cosas tan chicas.

Segunda idea: Kinect + OpenCV

Ante este problema, pensamos en descartar el sensor de profundidad y usar ambas cámaras con OpenCV para detectar la aguja por software. Como no dependemos del sensor, ahora podemos poner la aguja más cerca.

A continuación vemos, en orden, la cámara RGB y la cámara IR.

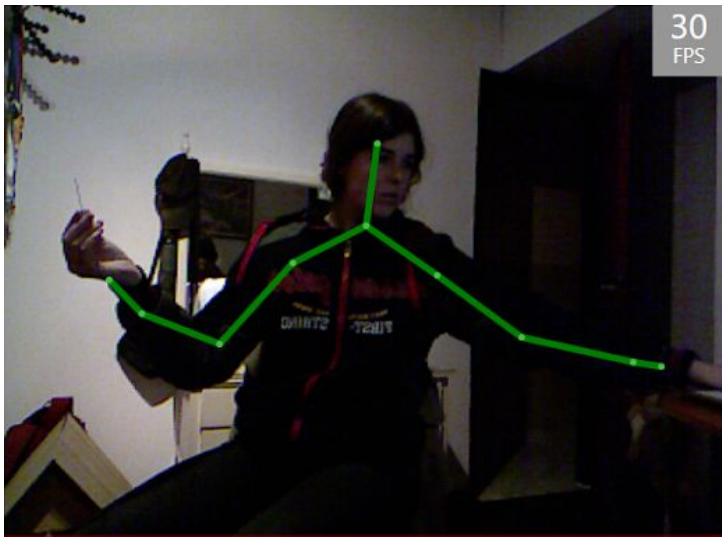


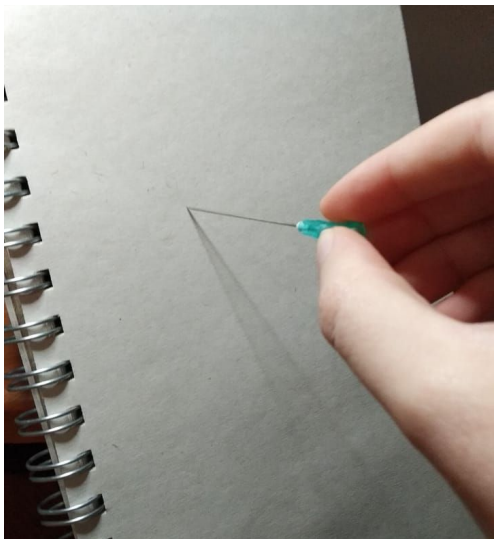


Como vemos, la cámara IR es bastante ruidosa y tiene poca definición. Además, al ser infrarroja, la fuente de iluminación es diferente y las sombras se vuelven muy marcadas. Lo que es un escenario ideal para la cámara RGB se complica para la cámara IR.

Pensamos seguir con esta idea de todas formas, pero nos encontramos con más problemas. El sensor de profundidad sí es necesario (y funciona) para el tracking de manos, por lo que la distancia mínima sigue siendo un requisito. Para explicarlo simple, si la persona a trackear estira el brazo y llega a tocar la kinect, está demasiado cerca.

Además, el trackeo de manos necesita ver desde adelante y todo el torso de la persona. Puede estar levemente de costado y desde arriba, pero no a 90°. Mientras que el trackeo de la aguja necesita ver la espalda del muñeco, lo que implica ver desde atrás.



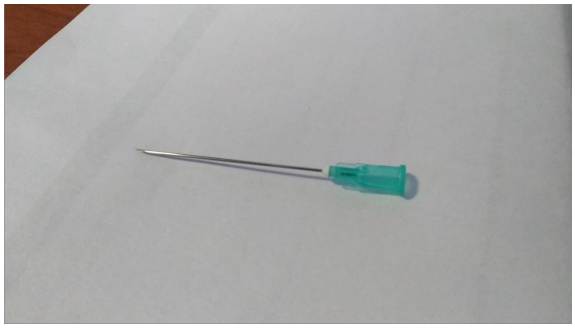
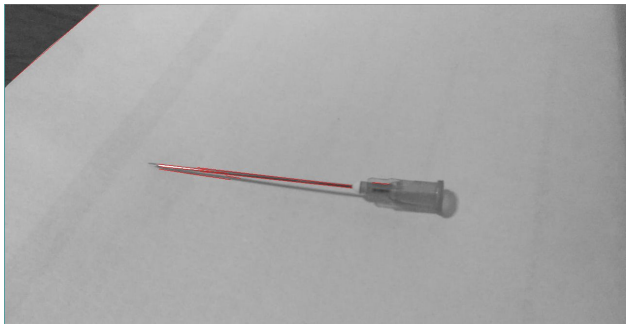
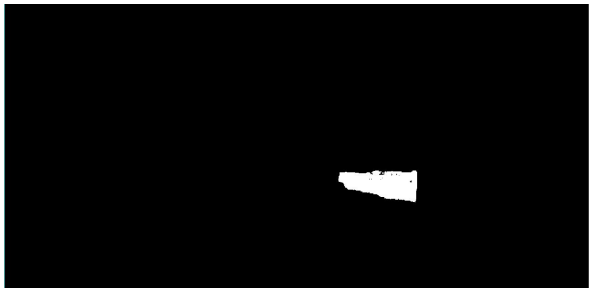


Llegamos a la conclusión de que ambos puntos de vista son incompatibles. Necesitamos dos dispositivos distintos.

Tercer idea: OpenCV + cámaras nuevas

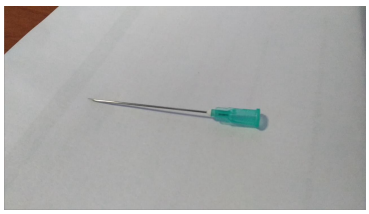
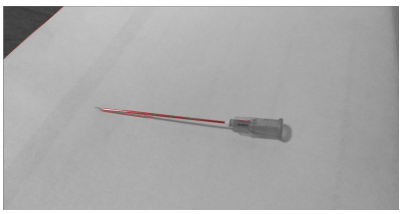
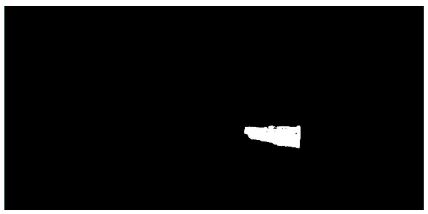
Como tercera opción pensamos en dejar la Kinect solo para el trackeo de manos y conseguir otras cámaras para trackear la aguja. Necesitamos dos cámaras para ver en 3D.

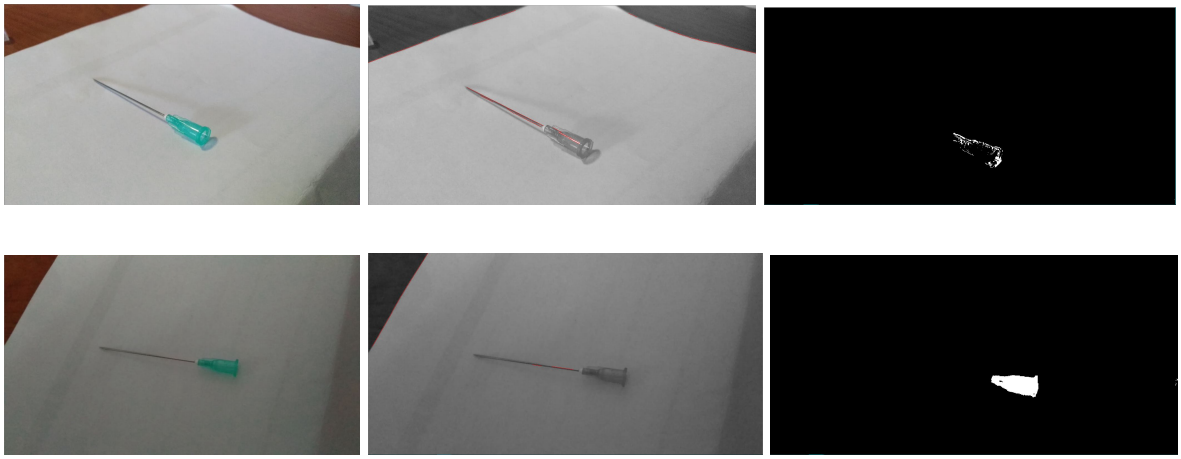
Con OpenCV podemos detectar líneas para encontrar la parte metálica de la aguja, aplicar filtros de color para encontrar el mango y así entender cuál es la punta que va a pinchar. Hicimos una prueba de concepto con fotos. A continuación vemos, en orden, la imagen original, la detección de líneas y el filtro de color.

Como vemos, el detector de líneas se confunde fácilmente con cualquier detalle del fondo, como el borde de la hoja, y con la misma sombra de la aguja. El filtro de color parece ir bastante bien.

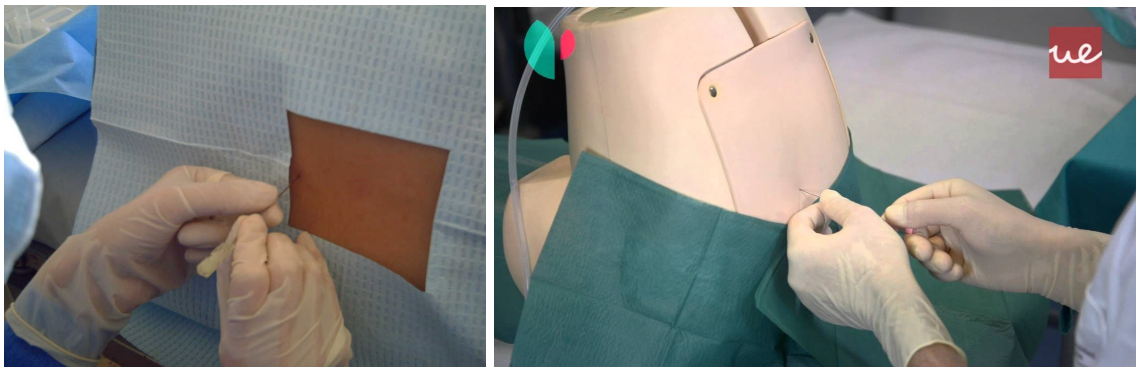
Probamos aplicar los mismos parámetros en un grupo de imágenes.



Como vemos, incluso con imágenes muy similares entre sí los detectores empiezan a fallar. Esto requiere un trabajo fino de calibración, que implica mucha prueba y error hasta encontrar los parámetros óptimos que cubran todos los casos. Y será necesario recalibrar cada vez que cambiemos de entorno, ya que este sistema es sensible a cualquier variación de iluminación y del fondo.

Además, serán frecuentes los casos donde la mano tape la parte metálica de la aguja, o cubra completamente el mango, perdiendo temporalmente el trackeo de la aguja.



Creemos que esta idea es factible, pero presenta demasiadas complicaciones para incluir en nuestro proyecto y terminar a tiempo.

Recomendaciones de mejora

El simulador de punción lumbar se puede mejorar agregando el trackeo de la aguja. Con este trackeo se podría detectar la posición de la aguja en 6 puntos de libertad (posición y ángulo en 3 dimensiones). Si además se mapea la espalda del muñeco se puede detectar

cuándo lo punza, dónde y con qué ángulo. Esta información se puede utilizar para indicar correcciones en tiempo real.

A continuación, dejamos una serie de recomendaciones para el desarrollo del tracking de la aguja.

Cámaras

Serán necesarias al menos dos cámaras para obtener una visión tridimensional.

Recomendamos que todas las cámaras sean del mismo tipo: tengan el mismo sensor y misma definición.

Diferentes sensores implican distinta sensibilidad a la luz y distintos colores, lo que varía la calibración de cada filtro. Mejores sensores obtienen mejores imágenes en condiciones de poca iluminación o movimientos rápido. Por el contrario, peores sensores obtienen imágenes ruidosas y "movidas" si no hay iluminación suficiente.

Distintas definiciones necesitan distintas transformaciones matemáticas para pasar de las coordenadas de un píxel al espacio tridimensional que queremos interpretar. Mayor definición permite notar más detalles, pero también implica más procesamiento.

Recomendamos ubicar las cámaras a 90° respecto de la otra. Por ejemplo, una mirando exactamente de arriba y otra mirando exactamente de la izquierda. Esto debería simplificar algunas cuentas. Es posible ponerlas en cualquier posición y ángulo, mientras ambas vean lo que necesitan. Tener en cuenta que si una cámara no detecta la aguja, perderemos el tracking completo, por lo que puede ser útil ubicar varias cámaras para redundancia. Cuánto más difieran sus posiciones, menor será el error del tracking y habrá menos puntos ciegos. Por el contrario, si están una al lado de la otra, la diferencia que permite calcular la profundidad será mínima, lo que resulta más propenso a errores.

No recomendamos usar las cámaras de la Kinect. Tienen distintas definiciones y sensores, ambas tienen poca definición y no están muy alejadas entre sí. La ventaja que tiene la Kinect es que ya está calibrada, la distancia entre cámaras es conocida y no cambiará.

Entorno

Como primer versión, recomendamos un entorno lo más controlado posible.

Este entorno debe estar bien iluminado. Se deben minimizar las sombras para que no confundan al sistema y se deben minimizar los reflejos. Tener en cuenta que la aguja

refleja cualquier luz puntual.

El fondo debe ser liso y simple. Evitar líneas que puedan confundir al detector de líneas. Buscar un color que contraste con la aguja para diferenciarla fácilmente.

Procurar que el entorno no cambie frecuentemente. Cada cambio en el entorno requerirá una nueva calibración del sistema.

Para las siguientes versiones, se podrá trabajar en remover alguna de estas restricciones. Recomendamos trabajar de a una a la vez, ya que cada restricción resuelve un problema complejo.

Calibración

El sistema necesita conocer la posición exacta de las cámaras, tanto la posición relativa entre ellas como respecto al mundo. Cualquier cambio en esta posición cambiará drásticamente los resultados.

Recomendamos desarrollar scripts o partes del sistema que asistan en esta calibración.

Recomendamos asistir la calibración con marcadores. Los marcadores son objetos visuales que cada cámara puede reconocer y definir su posición. Luego, el sistema puede juntar la información de cada cámara para definir la posición de cada una respecto al marcador.

Para dichos marcadores recomendamos ChArUco. ArUco es un tipo de QR mínimo, más liviano para procesar, más versátil y muy utilizado en realidad aumentada. Por otro lado, se suelen utilizar patrones de ajedrez para la calibración de cámaras, ya que ofrecen la precisión necesaria. ChArUco es una combinación de ambos. Dejamos un link a un tutorial de OpenCV sobre la detección de charucos
https://docs.opencv.org/3.4/df/d4a/tutorial_charuco_detection.html