Implementación de un Control Gestual para Robots

P. González, A. Brumovsky y M. Anigstein
Universidad de Buenos Aires. Facultad de Ingeniería.
Paseo Colón 850 (1063) Buenos Aires
pgonzal@fi.uba.ar, abrumov@fi.uba.ar

Resumen— **Se describe la implementación de un controlador de posición y orientación de un robot usando el sensor *Leap Motion*, planteando una alternativa al movimiento desde el joystick de la *teach pendant*. El objetivo de este trabajo es generar un método más intuitivo y directo para definir puntos destinos en el espacio. Se estudian además aspectos relativos a la calibración, y se exponen los detalles de implementación.**

Palabras clave : robot, control gestual, integración, visión.

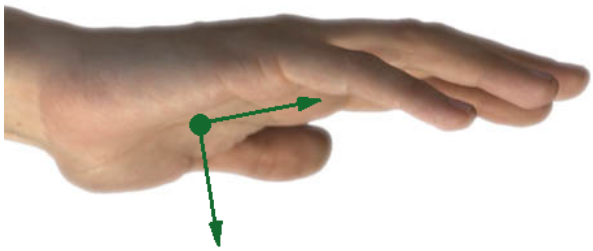

Fig. 1. Definición de la terna asociada a la palma. Se muestra el vector normal a la palma y el vector dirección.

I. Introducción.

La utilización de dispositivos para control gestual se ha incrementado fuertemente en los últimos años y ha encontrado numerosos nichos de aplicación, incluyendo teléfonos inteligentes[1], la tecnología del juego[2] y la domótica [1][2], entre otros.
La aparición de este tipo de dispositivos, disponibles en el mercado a un costo relativamente bajo, hizo que su utilización se vuelva cotidiana, introduciéndose inclusive en áreas no necesariamente masivas, como ser el guiado de robots[3] [3]. Al ser una tecnología relativamente nueva, si bien es necesario aún avanzar tanto en el hardware como en el software para aumentar la calidad de detección y la robustez en las identificaciones gestuales, se están realizando múltiples esfuerzos en dicha dirección y es de esperar que surgan sensores y software de procesamiento[4] confiables que permitan integrarse directamente a la celda de trabajo.

A. Organización del artículo.

En la sección II se describen los componentes de la solución. Luego, en la sección III, se muestra un método para referenciar las mediciones del sensor a una terna conocida por el robot. En IV se detalla el software de la implementación. Finalmente, en la sección V se presentan las conclusiones.

[1]http://www.nextbigwhat.com/control-phone-with-gestures-297/
[2]http://www.microsoft.com/en-us/kinectforwindows/
[3]http://spectrum.ieee.org/automaton/robotics/robotics-hardware/thalmic-myo-armband-provides-effortless-gesture-control-of-robots
[4]http://www.nickgillian.com/wiki/pmwiki.php/GRT/ GestureRecognitionToolkit

II. Componentes del sistema.

El sistema está compuesto por un robot industrial comunicado por red inalambrica con una computadora portátil donde se ejecuta un software de cálculo de coordenadas. Un sensor *Leap Motion*[5] detecta la posición y orientación de la palma de la mano del sujeto que controla el robot, y envía las mediciones a la computadora. Se pueden usar computadoras con el sensor integrado [6], lo que permitiría reducir el número de componentes.

A. Robot.

Se trata de un robot industrial antropomorfo de seis ejes, 5 kg. de capacidad de carga, aprox. 810 mm. de alcance máximo y 0,03 mm de repetibilidad. Su controlador tiene canales serie RS 232/422 y dos interfases Ethernet, una de ellas conectada a un ruteador *WiFi*. El lenguaje de programación *RAPID* permite incorporar ternas de referencia y transformaciones asociadas a cada herramienta en la definición de posiciones y movimientos, motivando y facilitando la elaboración de un código estructurado y reutilizable para las aplicaciones.

B. Sensor Leap Motion.

El sensor está compuesto por dos cámaras y un proyector de patrones integrados con un procesador que detecta y realiza el seguimiento de manos, dedos, y objetos alargados como lápices. El campo de visión (*Field of View*) cubierto consiste en un volúmen cónico que se extiende desde los 25 mm hasta los 600 mm en altura (eje $\mathbf{Y}$).

La mano se modeliza definiendo la palma con una terna como la mostrada en la Fig. 1, y los dedos según la Fig.2.

[5]https://www.leapmotion.com/product
[6]https://www.leapmotion.com/blog/hp-expands-leap-motion-integration-into-broad-range-of/

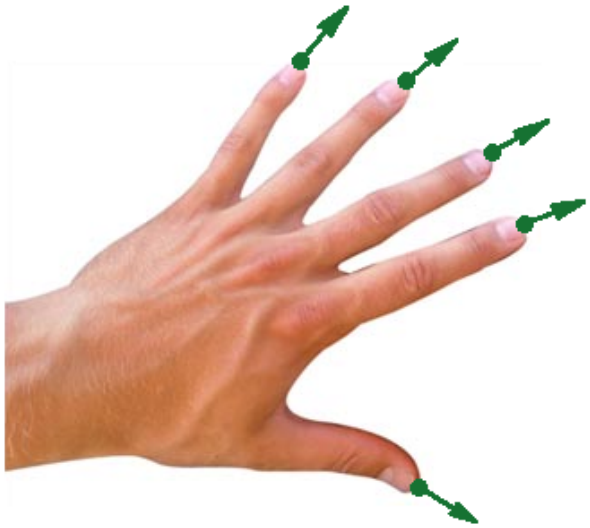

Fig. 2. Modelo de los dedos. Se define un vector asociado a cada dedo con origen en la punta del dedo y la dirección dada por el mismo.

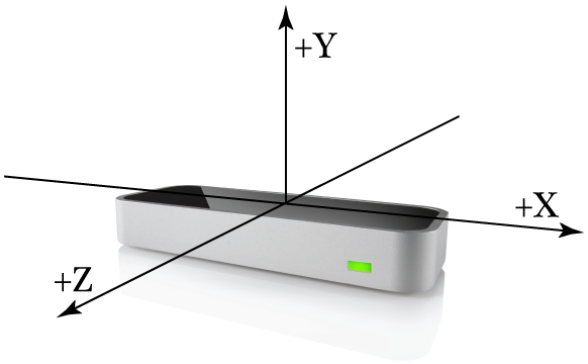

Fig. 3. Terna de referencia del sensor. El controlador del dispositivo permite rotar 180°internamente las mediciones sobre el eje **Y**

Las mediciones se expresan según la terna mostrada en la Fig. 3, y consisten en :

- Posición de la palma. Origen de la terna asociada a la palma (Fig. 1).
- Velocidad de la palma. Vector de desplazamiento obtenido entre sucesivas muestras.
- Vector normal a la palma. Definido saliente de la palma (Fig. 1).
- Dirección de la mano. Vector que apunta hacia los dedos (Fig. 1).
- Centro y radio de curvatura de la mano. Cuantifican que tan cerrada se presenta la mano.
- Lista de dedos descriptos por los vectores de la Fig. 2.

En cada medición o *frame* se pueden determinar varios objetos del tipo mano, los que se encontrarán completos en la medida de lo posible, ya que si la mano está cerrada probablemente no se pueda detectar alguno de los dedos.

El sensor provee facilidades adicionales como establecer una secuencia de movimiento entre *frames* utilizando esta información para eliminar falsas detecciones y dar mediciones más robustas, o a su vez la detección de gestos que pueden ser utilizados en actividades como la navegación por páginas web [7] [8].

C. Computadora.

Se utiliza una computadora portátil pues permite al operador ubicarse de la forma más conveniente para guiar la tarea. Además se utiliza la posibilidad de conexión inalámbrica con el controlador del robot para evitar el cableado innecesario.
El sistema operativo utilizado es *Ubuntu 12.04LTS* y el software se desarrolló en `C++` utilizando el IDE *QtCreator 2.4.1* con *Qt 4.8.1*. Por lo tanto no existiría limitación para compilar el proyecto en una plataforma *Windows*.

III. Calibración.

En robótica se busca programar las aplicaciones definiendo la estructura de la tarea con ternas de referencia. El robot tiene definida una terna en su base que sirve de referencia al resto. Para que el robot pueda utilizar las mediciones del sensor *Leap Motion* en forma directa debe conocer la terna de referencia que se muestra en la Fig. 3, o bien se deben proyectar las mediciones a una terna conocida del robot (por ejemplo, la base). En cualquiera de los dos casos se necesitará realizar un procedimiento de calibración.

Se ha trabajado en dos procedimientos alternativos. El primero, denominado método manual, consiste en utilizar al robot para medir una serie de puntos sobre el sensor y definir así una terna asociada al mismo que resulta semejante a la terna que utiliza el sensor para referir los resultados. El método automático, consiste en mostrar al sensor un objeto de calibración sostenido por el robot desde distintas posiciones y orientaciones; para cada una de ellas se vinculan los datos de posición de la herramienta del robot con las mediciones que realiza el sensor y se calcula la relación entre la terna base del robot y la terna del sensor.

A. Procedimiento manual de calibración.

El procedimiento manual es similar a la facilidad que incorpora el controlador del robot para definir las ternas de trabajo. En este caso el usuario debe mostrar en forma interactiva (Fig. 4) los tres puntos fresados en el soporte del sensor (Fig. 5) utilizando el joystick de la *teach pendant*. El primero de ellos indica el origen de la terna asociada al soporte, el segundo la dirección **X** y el tercero la **Y**. La dirección **Z** se calcula con el producto vectorial de $\mathbf{X} \times \mathbf{Y}$.

Con los vectores normalizados se puede armar una matriz que debido a los errores en el posicionamiento manual sobre los conos del calibrador, no es exactamente una matriz de rotación. Por lo tanto se puede calcular la matriz de rotación más cercana según la norma de Frobenius haciendo $R_0^{soporte} = UV^T$, donde las matrices U y

[7] http://leaptouch.com/
[8] Leap Motion Chrome Extension

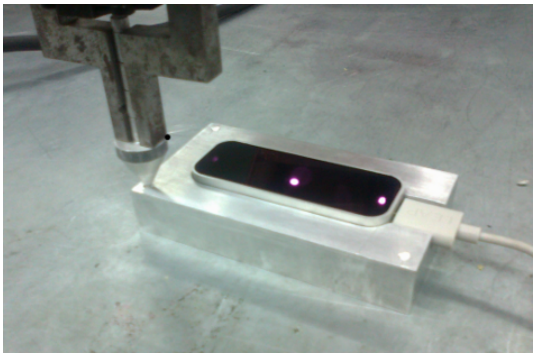

Fig. 4. Acercamiento de la herramienta al primer punto del soporte calibrador.

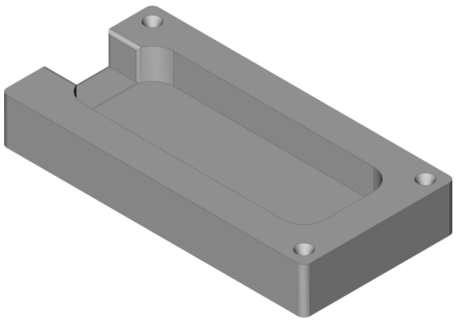

Fig. 5. Vista en 3D del soporte calibrador. Se observan las tres conos fresados que definen el origen de la terna del soporte y los ejes **X** sobre el lado más largo e **Y** sobre el otro. El lado abierto permite insertar el conector y determina una única forma de instalar el sensor dentro.

V resultan de la descomposición en valores singulares de $R_{aprox} = [\check{\mathbf{x}}\check{\mathbf{y}}\check{\mathbf{z}}] = U\Sigma V^T$. El origen de la terna $\mathbf{p}_0^{soporte}$ queda definido por el primer punto enseñado en la secuencia. Finalmente se puede armar una matriz homogenea $A_0^{soporte}$ que define la terna asociada al soporte calibrador respecto de la base del robot (Eq. 1).

$$A_0^{soporte} = \left[\begin{array}{c|c} R_0^{soporte} & \mathbf{p}_0^{soporte} \\ \hline \mathbf{0^T} & 1 \end{array}\right] \tag{1}$$

Como el sensor tiene una posición única respecto al soporte de calibración, la matriz $A_{soporte}^{leap}$ se puede calcular analíticamente según la Ec. 2.

$$A_{soporte}^{leap} = \left[\begin{array}{ccc|c} -1 & 0 & 0 & 50 \\ 0 & 0 & 1 & 25 \\ 0 & 1 & 0 & 5 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2}$$

En el lenguaje de programación *RAPID* un *work-object* incluye la definición de una terna usuario respecto de la base del robot, y una terna objeto respecto de esta última. En este caso se define un *work-object* llamado `leap_wobj` donde la terna usuario tomar el valor de la matriz $A_0^{soporte}$, y la objeto el de la matriz $A_{soporte}^{leap}$. Finalmente los movimientos se programan en el controlador del robot como instrucciones referidas a `leap_wobj`.

B. Procedimiento automático de calibración.

Siguiendo este método, el usuario debe correr una rutina programada en el robot que muestra al sensor un objeto de calibración desde distintas posiciones y con distintas orientaciones. El objeto de calibración es un elemento sobre el cual el sensor puede realizar las mismas determinaciones que hace normalmente sobre las manos. Un bloque de 130 mm x 65 mm x 7,5 mm sostenido por el gripper del robot se usa como elemento de calibración ya que no se requiere la información de la posición de los dedos, ni la curvatura de la palma.

En la estructura de la tarea se definen un conjunto de ternas asociadas al robot, al calibrador y al sensor. Las siguientes matrices homogéneas definen sus relaciones:

- A_0^6: relaciona la terna asociada al último eslabón del robot con la terna base del mismo,
- A_6^{cal}: relaciona la terna asociada al objeto calibrador con la del último eslabón del robot,
- A_{cal}^{leap}: expresa la inversa de la medición del sensor,
- A_0^{leap}: matriz de calibración, relaciona la terna del sensor con la terna base,

y cumplen

$$A_0^6(i)A_6^{cal}A_{cal}^{leap}(i) = A_0^{leap} \tag{3}$$

Para cada una de las muestras i, tomadas con el robot ubicado en una posición predefinida, son conocidas las matrices $A_0^6(i)$ pues se obtienen del robot, y las $A_{cal}^{leap}(i)$ pues son la inversa de la mediciones que arroja el sensor. Durante la tarea el robot mantiene firmemente sostenido el objeto de calibración, y además el sensor no se mueve; por lo tanto las matrices A_6^{cal} y A_0^{leap} son constantes aunque desconocidas. El problema consiste entonces en despejar de la Ec. 3 la matriz A_0^{leap}, y la solución se obtiene en dos pasos: primero resolviendo la parte de orientación y luego la traslación.

B.1 Obtención de la orientación del sensor respecto de la base.

Se puede escribir la Ec. 3 en función de la componente de rotación según la Ec. 4

$$R_0^6(i)R_6^{cal}R_{cal}^{leap}(i) = R_0^{leap} \tag{4}$$

Calculando la Ec. 4 para dos puntos distintos $i = 1, 2$

$$\begin{array}{l} R_0^6(1)R_6^{cal}R_{cal}^{leap}(1) = R_0^{leap} \\ R_0^6(2)R_6^{cal}R_{cal}^{leap}(2) = R_0^{leap} \end{array} \tag{5}$$

e igualando y reacomodando términos, se obtiene

$$R_6^{cal}R_{cal}^{leap}(1)R_{cal}^{leap}(2)^T - R_0^6(1)^T R_0^6(2)R_6^{cal} = 0 \tag{6}$$

En la Ec. 6, la matriz de rotación R_6^{cal} es la única incógnita. El robot sabe dónde se encuentra respecto de la base, $R_0^6(i)$, y el sensor mide $R_{cal}^{leap}(i)$ en cada punto. Para despejar R_6^{cal}, la Ec. 6 se expresa como el producto de una matriz Q de dimension $(3x9)$ con los datos, y un vector columna $\mathbf{x}$ de 9 elementos que contiene las incógnitas: las tres columnas de R_6^{cal}.

$$R_6^{cal} R_{cal}^{leap}(1) R_{cal}^{leap}(2)^T - R_0^6(1)^T R_0^6(2) R_6^{cal} = Q\mathbf{x} = 0 \quad (7)$$

Para resolver la Ec. 7, se requiere tomar al menos 3 puntos; es decir que es necesario ampliar el rango de Q apilando las ecuaciones planteadas con otro par de puntos. La solución de esta ecuación es el vector singular derecho asociado al valor singular más pequeño de Q [4]. La matriz R_x obtenida a partir de las incógnitas despejadas en $\mathbf{x}$ no será ortonormal debido a los errores de medición, entonces se procede como en la sección III-A para obtener la matriz de rotación R_6^{cal} más cercana a R_x.

Finalmente, la matriz R_0^{leap} se obtiene reemplazando en la Ec. 4 la matriz R_6^{cal} hallada .

C. *Cálculo del origen de las ternas del objeto de calibración y del sensor.*

Una vez calculadas las componentes de rotación de las matrices homogéneas, pueden obtenerse las componentes de traslación. Según la definición de la matriz homogénea, de la Ec. 3 se puede despejar $\mathbf{p}_0^{leap}$

$$R_0^6(i)(R_6^{cal}\mathbf{p}_{cal}^{leap}(i) + \mathbf{p}_6^{cal}) + \mathbf{p}_0^6(i) = \mathbf{p}_0^{leap} \quad (8)$$

Llamando $\mathbf{p}(i) = R_0^{6T}(i)\mathbf{p}_0^6(i)$, y reemplazando en la Ec. 8 resulta

$$R_6^{cal}\mathbf{p}_{cal}^{leap}(i) + \mathbf{p}(i) = R_0^{6T}(i)\mathbf{p}_0^{leap} - \mathbf{p}_6^{cal} \quad (9)$$

que puede escribirse como

$$\left[\begin{array}{c|c} R_6^{cal} & I \end{array}\right] \left[\begin{array}{c} \mathbf{p}_{cal}^{leap}(i) \\ \hline \mathbf{p}(i) \end{array}\right] = \left[\begin{array}{c|c} R_0^{6T}(i) & -I \end{array}\right] \left[\begin{array}{c} \mathbf{p}_0^{leap} \\ \hline \mathbf{p}_6^{cal} \end{array}\right] \quad (10)$$

En la Ec. 10 quedan agrupadas las incógnitas en los vectores $\mathbf{p}_0^{leap}$ y $\mathbf{p}_6^{cal}$, que contienen la información de la posición de la terna del sensor respecto de la base y del calibrador respecto de la terna 6 del robot, respectivamente. La solución se encuentra apilando las ecuaciones de al menos tres puntos ($i = 1, 2, 3, ...$) y resolviendo por mínimos cuadrados.

IV. Software.

El software desarrollado para la computadora es el encargado de tomar las mediciones del sensor, proyectarlas en la terna de calibración, y enviarlas al robot. Por otro lado el software escrito para el robot mueve la herramienta según se actualizan las mediciones. El sistema está diseñado para que el usuario acceda a todas las funciones desde la *teach pendant*, por lo que el programa del robot puede disparar las rutinas de calibración o de seguimiento de manos que están programadas en la computadora.

A. *Arquitectura.*

El sotfware de sensado está programado en lenguaje `C++`, basándose en la *API* (Application Programming Interface) que provee el fabricante del sensor.

El programa resuelve la comunicación con el robot y la adquisición del sensor mediante las clases `TComunicator` y `leapListener` que implementan internamente un hilo de procesamiento, por lo que el robot está leyendo constantemente las mediciones del sensor y resolviendo simultáneamente los comandos enviados desde el robot.

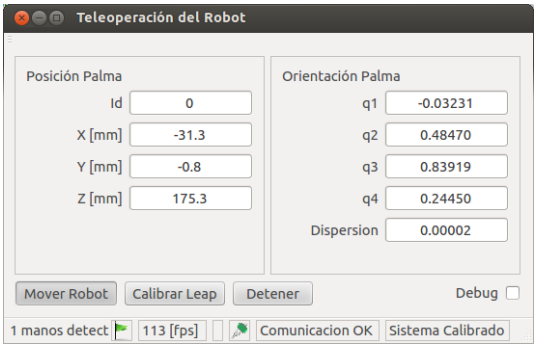

Fig. 6. Pantalla de la aplicación de la PC. Se observa la medición actual (posición y orientación dada por un cuaternión), los botones para realizar las acciones y los indicadores de estado del sensor y el robot.

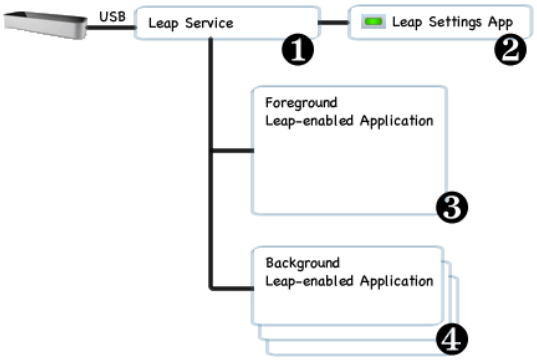

Fig. 7. Arquitectura del software del sensor. Un servicio resuelve la comunicación por *USB* con el sensor. Luego distintos programas pueden conectarse simultáneamente por *TCP*, como por ejemplo la aplicación para configurar el sensor y el sistema de control gestual.

Ambos objetos envían datos de estado, valores de mediciones, etc. al hilo que maneja la pantalla mediante el mecanismo asincrónico de señales y conectores provisto por la biblioteca *Qt*. Del mismo modo reciben comandos generados al presionar los botones de la pantalla (Fig. 6).

B. *Comunicación PC-sensor.*

El software del sensor corre como un servicio en las plataformas *Windows* y como un *daemon* en *Linux*. El sensor se conecta a la PC por medio de *USB*. Las aplicaciones programadas en la PC acceden a los servicios del software del sensor a través de la *API*, que implementa la comunicación por *TCP* (Fig. 7).

La *API* provee una clase `Controller` que sirve de interfase. Entre sus varias funciones destacamos la de detectar los sensores conectados a la PC, establecer la comunicación con ellos y adquirir los distintos tipos de mediciones. Mediante el método `frame` se puede obtener una medición, pero resulta más adecuado para esta aplicación usar la facilidad de definir y anexarle al controlador un objeto de la clase `listener`.

La clase `listener` es abstracta, lo que significa que tiene un grupo de métodos virtuales que deben ser redefinidos por el programador, permitiendo así establecer la respuesta para los distintos eventos. En particular se define una clase `leapListener` que en los métodos virtuales para manejar eventos tales como la conexión, inicialización y puesta en foco interactúa con la pantalla, y que para el método `onFrame` sigue los siguientes pasos:

- si no estuviera siguiendo una mano en particular toma la primer mano detectada y almacena su identificación,
- calcula la normal a la palma y el vector dirección de la mano identificada,
- arma una terna usando el producto vectorial para obtener el vector transversal,
- obtiene la posición del centro de la palma, y arma la matriz homogénea $A_{leap}^{hand}(i)$
- con las N últimas matrices $A_{leap}^{hand}(i)$, calcula una matriz promedio para limitar el efecto de los errores de medición. En [5] se describen varios métodos para promediar rotaciones.
- obtiene la dispersión en el conjunto de matrices promediadas,
- si la dispersión es muy grande deja de seguir la mano actual y borra la historia; en caso contrario y al cabo de tener una cierta cantidad de mediciones exitosas, publica el resultado,
- mide el radio de curvatura de la palma y usa esta información para abrir y cerrar la pinza; otros gestos se pueden reconocer, tal como el movimiento de alguno de los dedos (*key tap*), que no interfieren con la posición y orientación de la palma medida, y pueden asignarse a ellos otras funciones en el programa del robot.

Como el sensor envía a la computadora solo las mediciones obtenidas sobre las manos, se ahorran importantes recursos que supondrían la transmisión de las imágenes que utiliza para determinarlas. La velocidad de adquisición llega así a valores tales como 120 Hz, por lo que la respuesta dinámica del sistema no se deteriora al usar filtros de promedio móvil con ventanas de 10 muestras.

C. Comunicación robot-PC.

La comunicación entre el sistema de visión y el robot está implementada con comandos *RAP* (Robot Application Protocol) de ABB, sobre *RPC* (Remote Procedure Calls). Tanto para sistemas operativos *Linux* como para *Windows* se encuentra desarrollada la biblioteca dinámica *RAPLIB* que permite acceder desde programas escritos en `C/C++`, o *matlab*, a las funciones de comunicación sin necesidad de programar ninguno de los comandos de RAP [6].

La clase `TComunicator` implementa la comunicación con el robot encapsulando los comandos de *RAPLIB* y resolviendo las solicitudes del robot. Mediante un hilo de procesamiento queda a la espera de recibir algún mensaje del robot y dispara las funciones *callback* (una pieza de código que es pasado como argumento a otro código) que definen el comportamiento para cada uno de ellos, actuando de esta forma como un servidor.

Alguna de los métodos públicos de ésta clase permiten adicionalmente escribir y leer:

- posiciones y orientaciones,
- ternas de trabajo,
- herramientas,
- variables numéricas,
- y arreglos de variables.

Los comandos resueltos por medio de *callbacks* permiten en esta aplicación arrancar o detener la secuencia de medición y resolver la calibración automática. Cuando la secuencia se habilita, el programa comienza a actualizar la posición y orientación destino (llamada `pleap` en el robot) en función de la mano detectada.

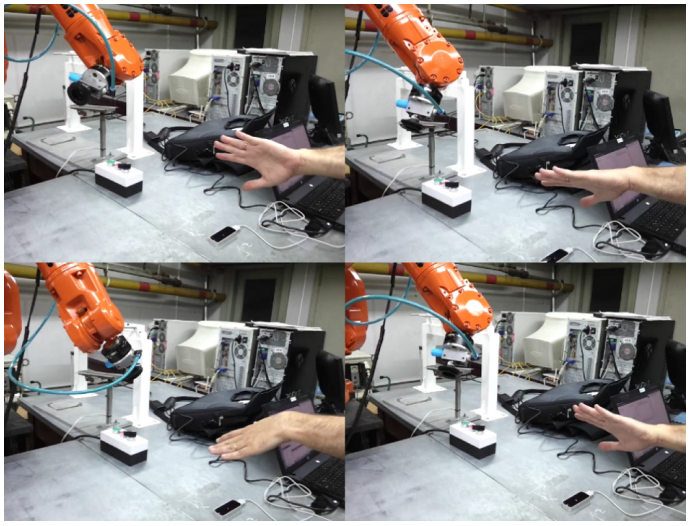

Fig. 8. Distintos instantes captados durante un movimiento de reorientación. Se observa la mano que guia la herramienta del robot y el sensor bajo la misma.

D. Software del robot.

El programa que corre en el controlador del robot permite acceder a los comandos implementados en la aplicación de la PC. Consiste en un ciclo que realiza un movimiento *joint* a un punto destino llamado `pdest` que es actualizado continuamente por el software de la PC con el valor medido por el sensor `pleap` más un desplazamiento fijo para evitar que el robot se posicione exactamente donde está la mano que lo guía.

En la instrucción de movimiento se especifica un tiempo de realización de 500 ms que es aproximadamente el tiempo en que se actualizan las mediciones. Además se fija una zona de acercamiento de 15 mm para la posición y 10°para la rotación, consiguiendo así un encadenamiento natural de los movimientos. En la Fig. 8 se observan distintos instantes en la realización de una tarea de reorientación.

Como las coordenadas destino no son conocidas al momento de escribir el programa, no se pueden fijar los índices de cuadrantes de los ejes 1, 4 y 6 (coeficientes cf_1, cf_4 y cf_6 del vector de configuración del robot *ABB*). Por lo tanto se deben calcular en forma dinámica, luego de actualizar `pdest` y antes de realizar el movimiento, mediante el módulo de corrección de cuadrantes [7].

Para desarrollar una *teach gestual*, se deben incluir va-

rias facilidades en el programa del robot, como por ejemplo mostrar las coordenadas de la herramienta durante el movimiento, permitir hacer el movimiento respecto de la base del robot o de alguna terna de trabajo y permitir seleccionar distintas herramientas entre otras.

V. Conclusiones.

Se han mostrado las bases para realizar un control gestual para robots basado en un sensor óptico sin contacto (*Leap Motion*). Tanto el funcionamiento del sensor como su calibración fueron abordadas, proponiendo dos métodos, uno manual y otro automático para calibrar el sistema. Dicha calibricación es necesaria para hacer coincidir las ternas de referencia del robot y del sensor *Leap Motion*. Se ha detallado a su vez la estructura de software utilizado para poder interactuar tanto con el sensor gestual como con el robot de manera de poder realizar aplicaciones a posteriori. El software desarrollado es lo suficientemente general y sirve como punto de partida para la implementación de un control gestual en el robot que permita definir instrucciones gestuales más complejas.

Referencias

[1] H. H. Ana Carla de Carvalho Correia, Leonardo Cunha de Miranda, *Human-Computer Interaction INTERACT 2013*, ch. Gesture-based interaction in domotic environments: State of the art and HCI framework inspired by the diversity. Springer, 2013.

[2] S. G. Qifan Pu, Sidhant Gupta and S. Patel, "Whole-home gesture recognition using wireless signals," in *MobiCom '13*, pp. 27–38, 2013.

[3] J. N. Kun Qian and H. Yang, "Developing a gesture based remote human-robot interaction system using kinect," *International Journal of Smart Home*, vol. 7, no. 4, pp. 335–340, 2013.

[4] G. G.H. and V. L. Ch.F., *Matrix Computations*. London: Johns Hopkins, 1989.

[5] M. R. Inna Sharf, Alon Wolf, "Arithmetic and geometric solutions for average rigid-body rotation," *Elsevier, Mechanism and Machine Theory*, vol. 45, pp. 1239–1251, 2010.

[6] M. Roaux, P. Ilardi, and M. D'Ascanio, "Comunicación con el controlador del robot abb irb 140." Trab. Final de Taller de Prog. III y Robótica, FIUBA, Nov 2003.

[7] C. J. Cartelli and M. Anigstein, "Sobre la programación off-line de robots industriales y las múltiples soluciones para una posición objetivo," in *VII Jornadas Argentinas de Robótica*, (Olavarría, Buenos Aires), JAR012, Oct 2012.