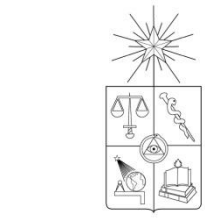
Universidad de Chile

Facultad de Ciencias Físicas y Matemáticas

Departamento de Ingeniería Mecánica



Informe pierna prostética

"Captura de movimiento"

Elaborado por:
Orlando Campos B.

Profesor: Mónica Zamora.

Índice

1 Introducción

Las prótesis son extensiones artificiales que reemplazan o mejoran la función de una parte del cuerpo, ya sea porque esta ha fallado o ha sido extraída. Para el estudio y diseño de una prótesis es necesario hacer diversas investigaciones respecto a los centros instantáneos de movimiento y ángulos que se generan entre distintos puntos del cuerpo, un método efectivo para lograr esto es la Captura de movimiento.

La captura de movimiento, por sus siglas en ingles MOCAP (Motion Capture), es una técnica de grabación de movimiento de personas, animales u objetos para registrarlos y ser llevados a un modelo computacional para su procesamiento.

Para el registro de movimiento se necesitan a lo menos dos cámaras, dado que con más de una cámara es posible detectar la profundidad de un objeto con respecto al eje de su punto de mira.

Para que el movimiento sea detectado por una cámara, es necesario utilizar marcadores corporales, la cámara es sensible a estos marcadores por lo cual entrega a la computadora la posición de cada uno de ellos con respecto a un punto de referencia, de esta manera el movimiento de una persona se puede estudiar como el movimiento de N partículas en el espacio. Las cámaras utilizadas son llamadas Optitrack de la empresa NaturalPoint, estas envían la información detectada al software Motive para su procesamiento.

En el presente informe se mostrará los pasos para la grabación de partículas y el método para procesar los cambios de posiciones vía MATLAB, a fin de poder obtener un gráfico en tres dimensiones de la trayectoria de N partículas solas y unidas por eslabones, con esto poder calcular el ángulo que se va formando entre cada eslabón.

2 Objetivos

Para poder realizar una investigación de los centros instantáneos de rotación y ángulos entre varios puntos del cuerpo, obtenidos de la caminata de un sujeto normal y grabándola utilizando motive, es necesario contar con un software que facilite la interacción usuario-máquina y entregue sólo la información necesaria.

Para realizar lo anterior se necesita lograr un dominio del software MOTIVE y obtener las coordenadas de movimientos de N partículas, esto permite graficar las trayectorias correspondientes a las N partículas y eslabones entre ellas, una vez obtenidos es posible calcular el ángulo entre eslabones y los centros instantáneos de rotación.

Los objetivos principales son:

- Familiarizarse con el uso del software MOTIVE, para la detección de movimiento de N partículas y obtención de las coordenadas de estas en el espacio.
- Graficar en MATLAB el movimiento de estas partículas, ya sea en forma de trayectoria como en posición respecto al tiempo (en fotogramas).
- Para más de una partícula, crear eslabones entre dos partículas, y realizar el paso anterior con el eslabón.
- Obtener centros instantáneos de rotación y ángulos entre eslabones.
- Creación de un GUI que permita un fácil manejo entre el usuario y el software.

3 Introducción al uso de MOTIVE:

3.1 Inicio

El Software Motive es una herramienta dinámica capaz de procesar y trayectorizar el movimiento de varias partículas en sus tres coordenadas, entregando los recursos necesarios para el modelado y estudio de los movimientos capturados.

Primero se ejecuta el programa, una vez abierto está habilitado para capturar el movimiento (imagen 3.1), para esto hay que pulsar el botón rojo REC, de esta manera el software ira guardando el movimiento de las partículas. Si se desea terminar la grabación se pulsa el botón STOP.

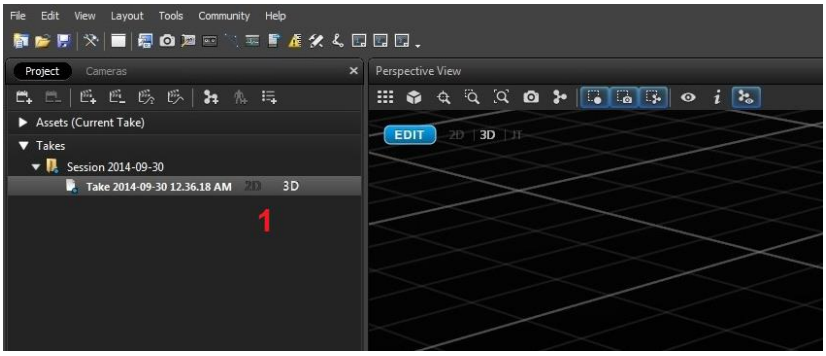


Imagen 3.1 Método para trayectorizar.

3.2 Trayectorización

Una vez hecha la grabación se puede trayectorizar la partícula, esto es, pedirle al programa que entregue en matrices las coordenadas de cada partícula (imagen 3.1), se debe hacer clic con el botón derecho en (1) y luego seleccionar “Trayectorizar”. Para poder utilizar MATLAB con las matrices es necesario exportar el archivo en formato ***CSV*** (imagen 3.1), se debe hacer clic con el botón derecho en (1) y luego seleccionar “Exportar”. Realizados estos pasos se deja de utilizar el software MOTIVE para trabajar con los resultados en MATLAB.

3.3 Errores y opciones avanzadas en MOTIVE

Es frecuente que existan ciertos errores en el programa al momento de detectar objetos, el problema más común es la incapacidad de detectar marcadores de gran tamaño, por lo cual

es preferible que los marcadores sean pequeños, menores a 1 cm^2, los cuales son detectados sin dificultad.

Puede ocurrir que en mitad de grabación alguna partícula “parpadee” en el monitor, esto es una alerta de que la cámara no detectó el movimiento por un instante, cuando ocurre esto es posible continuar grabando. Una vez finalizada la grabación se debe trayectorizar, luego ir a la ventana “***TAKE***” (imagen 3.2) donde se muestran las trayectorias de todas las partículas. Cuando una partícula “pestañea” el programa asume que esa partícula desapareció y que en su lugar se agregó una nueva, por lo tanto si tuviéramos tres partículas en una grabación y una de ella parpadeó una vez, al trayectorizar el programa nos lanzará cuatro trayectorias. Para solucionar esto basta tomar la trayectoria extra y colocarla encima de la trayectoria inicial, MOTIVE unirá las dos trayectorias, luego el archivo puede ser exportado normalmente.

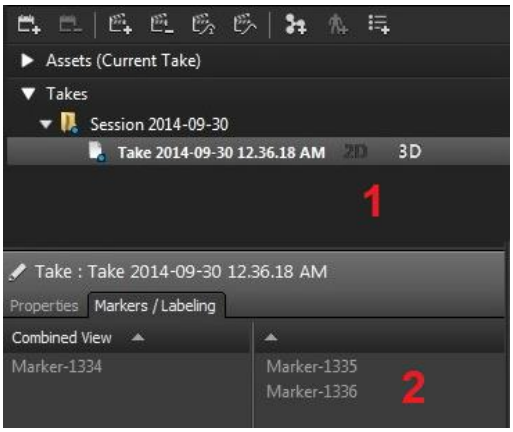


Imagen 3.2 Sección para unir dos trayectorias.

Las cámaras son sensibles a la luz, ambientes con luz artificial son más adecuados que con luz natural, la luz de los tubos fluorescentes también tiende a causar “parpadeos” al momento de grabar. Se puede solucionar esto ajustando los parámetros EXP, THR y LED (imagen 3.3).

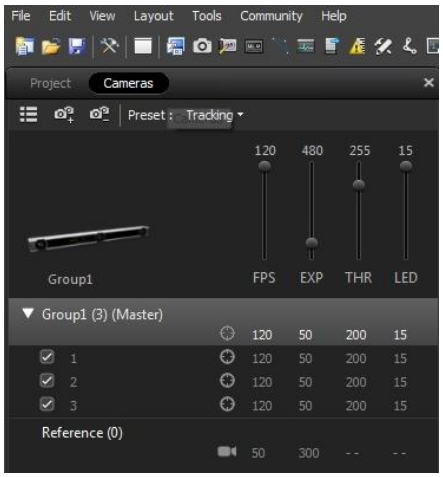


Imagen 2.3 Ajustes de la cámara.

4 Resultados

4.1 Una partícula

4.1.1 Trayectoria

Realizando las pasos nombrados en la introducción a el uso de MOTIVE, se trayectorizó el movimiento de una partícula, esta se exporta a MATLAB como una matriz de *N x 3*, donde *N* corresponde a la cantidad de fotogramas tomados en la grabación. Si la matriz la guardamos en un objeto, como ejemplo M, entonces para graficar su trayectoria en MATLAB se invoca la función plot3, en este caso sería:

$$plot3(M(:,1), M(:,2), M(:,3))$$

Función 4.1.1: Función para graficar trayectoria de una partícula.

Donde M(:, i) corresponde a la i columna de la matriz.

Los resultados de un movimiento aleatorio de una partícula se muestran en la imagen 4.1.1

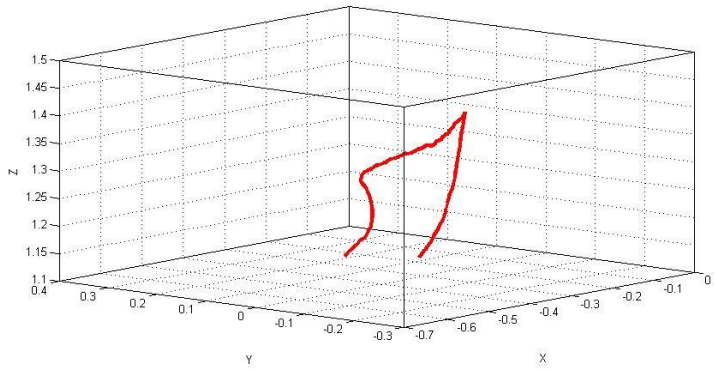


Imagen 3.1.1 Trayectoria de una partícula.

4.1.2 Partícula en movimiento

Si se desea graficar la posición de una partícula con respecto a los fotogramas, se obtendrá una animación de la partícula, para ello basta utilizar la siguiente función:

```
for i=1:length(M)
X=M(i,1);
Y=M(i,2);
Z=M(i,3);
plot3(X,Y,Z,'k-x')
pause(0.001)
end
```

Función 4.1.2: Función para animar la partícula

Si se desea ver en conjunto la trayectoria y la posición de la partículas con respecto al tiempo se debe llamar a "***hold on***" antes de la función 4.1.1 y 4.1.2 y luego ejecutar el programa.

4.2 Dos partículas, un eslabón

4.2.1 Trayectoria partículas

En esta sección se midieron dos marcadores unidos a una barra, por lo tanto la distancia entre estos es siempre la misma. Ahora se tienen dos matrices, M1 y M2, utilizando el procedimiento de la función 4.2.1 se escribe:

```
hold on
plot3(M1(:,1),M1(:,2),M1(:,3));
plot3(M2(:,1),M2(:,2),M2(:,3));
for i=1:length(M1)
X1=M1(i,1);
Y1=M1(i,2);
Z1=M1(i,3);
X2=M2(i,1);
Y2=M2(i,2);
Z2=M2(i,3);
plot3(X1,Y1,Z1,'k-x')
plot3(X2,Y2,Z2,'k-x')
pause(0.001)
end
```

Función 4.2.1: Función para graficar la trayectoria de dos partículas.

El resultado es la gráfica de la trayectoria de dos partículas, se puede apreciar en la imagen 4.2.1 que la distancia entre los puntos no varía.

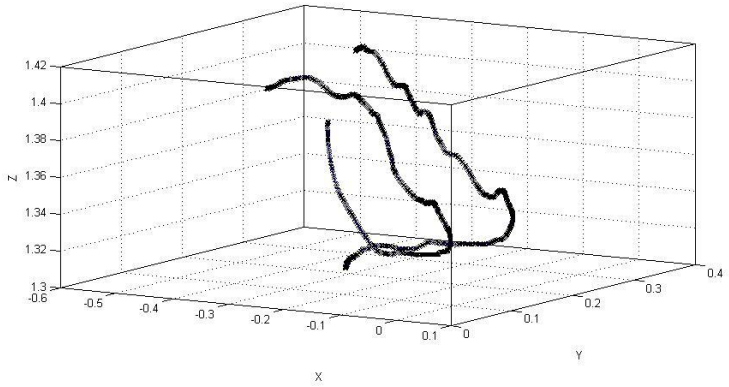


Imagen 4.2.1 Gráfico de la trayectoria de dos partículas.

4.2.2 Trayectoria eslabón

Para poder dibujar la trayectoria del eslabón que une los dos puntos se debe escribir la siguiente función:

```
hold on
plot3(M1(:,1),M1(:,2),M1(:,3));
plot3(M2(:,1),M2(:,2),M2(:,3));
for i=1:length(M1)
X1=M1(i,1);
Y1=M1(i,2);
Z1=M1(i,3);
X2=M2(i,1);
Y2=M2(i,2);
Z2=M2(i,3);
X=[X1 X2];
Y=[Y1 Y2];
Z=[Z1 Z2];
plot3(X,Y,Z,'k-x','LineWidth',2)
xlim([-0.7 0.1])
ylim([-0.2 0.5])
zlim([1.2,1.6])
pause(0.001)
end
```

Función 4.2.2: Función para graficar la trayectoria de un eslabón que une dos partículas

Si se une un eslabón entre los dos puntos tendría que surgir una imagen similar a una cinta, como se muestra en la imagen 4.2.2

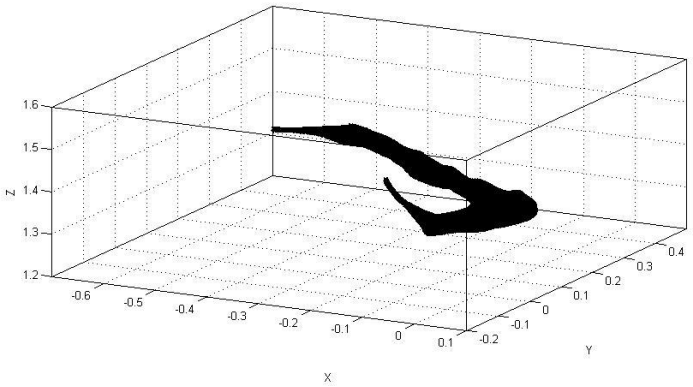


Imagen 2.2.2: Grafico de la trayectoria del eslabón.

4.2.3 Eslabón en movimiento

Es de interés ver la posición a través del tiempo del eslabón, y no la trayectoria de la barra completa, por lo tanto si se escriben las líneas de código de la función 4.2.3 sobre la función 4.2.2 se puede visualizar el movimiento del eslabón a través del tiempo como en la imagen 4.2.3, esto es debido a que la posición anterior es borrada cuando se dibuja la nueva posición, dado la sensación de movimiento.

```
clear all;
close all;
clc;
```

Función 4.2.3: Función que permite visualizar el eslabón en movimiento.

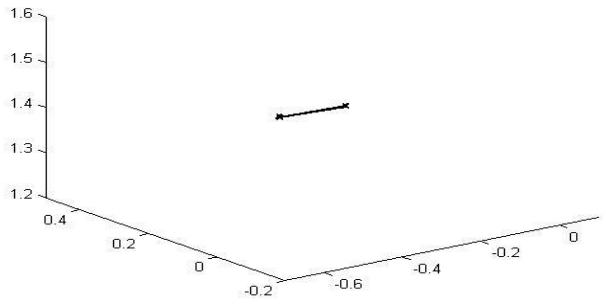


Imagen 4.2.3: Un fotograma del movimiento del eslabón.

Con esto obtenemos finalmente el movimiento de la barra tal cual como se movió frente a las cámaras Optitrack.

4.3 Tres partículas, dos eslabones unidos con un ángulo variable en el tiempo

4.3.1 Posición de marcadores

El siguiente experimento contempla tres marcadores en un brazo, uno unido a la muñeca (marcador 1), otro unido al codo (marcador 2) y otro unido en el brazo, sobre el codo (marcador 3), como se muestra en la imagen 4.3.1.

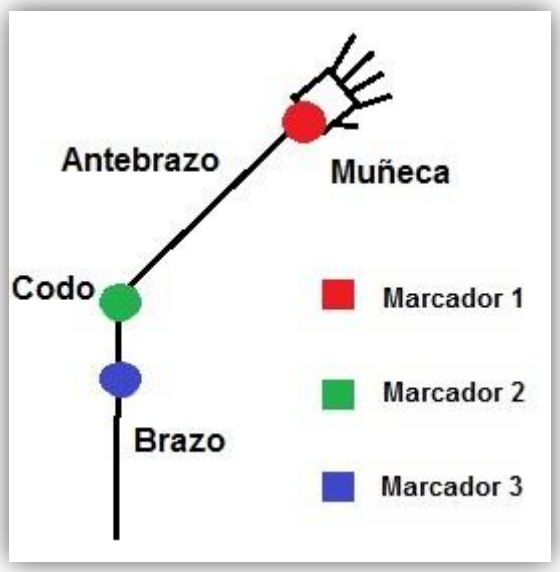


Imagen 4.3.1: Referencia de las posiciones de los marcadores en el brazo.

4.3.2 Trayectoria

Al ser tres marcadores se importa una matriz de N X 9, llamada matriz M, la cual se subdivide en tres matrices; M1 (marcador 1), M2 (marcador 2) y M3 (marcador 3). Utilizar este método permite ahorrar tiempo, dado que si no se utiliza, cuando se tengan T marcadores, habrá que crear T objetos, donde cada objeto será una matriz de N X 3.

Para realizar lo anteriormente mencionado se utiliza la siguiente función:

```
M;
M1=M(:,1:3);
M2=M(:,4:6);
M3=M(:,7:9);
```

Función 4.3.1: Código para dividir la matriz total en tres matrices, cada una representa un marcador.

Para este experimento el brazo se movió de forma perpendicular a las cámaras, realizando la siguiente trayectoria mostrada en la Imagen 4.3.2

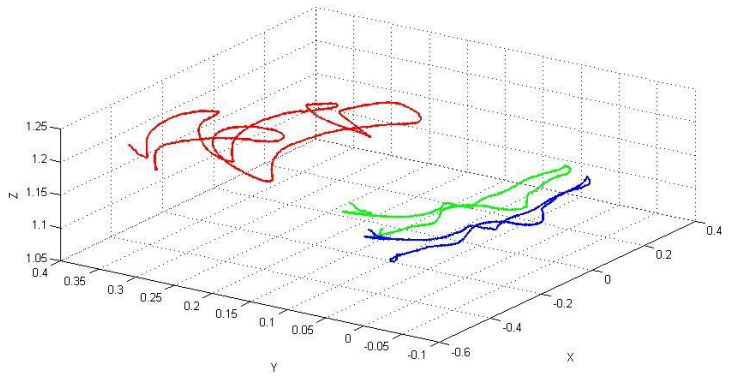


Imagen 4.3.2: Trayectoria de movimiento de los tres marcadores, los colores corresponden a la Imagen 4.3.1.

El movimiento realizado fue rotar lo menos posible el brazo y el codo y moverlos juntos, de forma recta hacia la derecha (eje positivo x), mientras que el antebrazo se extendía completamente (cerca de los 180º con respecto al brazo) y luego se cerraba (cerca de los 75º con respecto al brazo) intermitentemente, el movimiento se hizo de ida y de vuelta.

Para poder realizar el gráfico se escribe el siguiente programa:

```
hold on
plot3(M1(:,1),M1(:,2),M1(:,3),'red');
plot3(M2(:,1),M2(:,2),M2(:,3),'green');
plot3(M3(:,1),M3(:,2),M3(:,3),'blue');
```

Función 4.3.2: Código para graficar la trayectoria de los tres marcadores.

4.3.3 Trayectoria eslabones

En esta ocasión no se mostrará la trayectoria que siguen los eslabones, puesto que se superponen y no se contempla una imagen clara. Para poder ver el movimiento de los dos eslabones y obtener el ángulo entre ellos se escribe el siguiente código:

```
%se inicializan las variables para el calculo de ángulos
M11=M2-M1;M22=M2-M3;
ANGULOS=zeros(length(M1),1);

for i=1:length(M1)

%%%%calculo de cada ángulo
producto_Escalar=sum(M11(i,:).*M22(i,:));
producto_Modulos=sqrt(sum(M11(i,:).*M11(i,:)))*sqrt(sum(
M22(i,:).*M22(i,:)));
ANGULOS(i)=acosd(producto_Escalar/producto_Modulos);
%%%%fin calculo de ángulo, se guarda en el vector
Angulos

X1=M1(i,1);Y1=M1(i,2);Z1=M1(i,3);
X2=M2(i,1);Y2=M2(i,2);Z2=M2(i,3);
X3=M3(i,1);Y3=M3(i,2);Z3=M3(i,3);

X=[X1 X3 X2];Y=[Y1 Y3 Y2];Z=[Z1 Z3 Z2];

plot3(X,Y,Z,'k-x','LineWidth',2)
%view(-66,82);
view(-32,90);
xlim([-0.5 0.4])
ylim([-1 1.4])
zlim([-0.2,1.4])
pause(0.0001)

end
%%%%Grafico del ángulo vs los fotogramas

plot(linspace(1,length(M),length(M)),ANGULOS)[1]
title('Angulo vs Fotograma');
xlabel('Fotograma');
ylabel('Angulo (°)');
```

Función 4.3.3: Código para animar el movimiento de los eslabones y graficar el Angulo vs los fotogramas entre los eslabones.

4.3.4 Eslabones en movimiento y ángulo entre ellos

El procedimiento para simular la escena es el mismo que para dos partículas, para calcular el ángulo se sabe que el producto escalar de dos vectores, dividido en el producto de sus módulos es igual al coseno del ángulo que hay entre ellos. Motive nos entrega los vectores que hay desde el origen hasta la posición de cada marcador, pero se necesitan los vectores entre los marcadores, por lo tanto el vector que hay del marcador 2 al marcador 1 es M11=M2-M1 y el vector que va desde el marcador 2 al marcador 3 es M22=M2-M3, con esto aplicado a lo mencionado anteriormente se obtiene el coseno del Angulo entre los dos vectores, solo falta llamar la función inversa del coseno

[1] `linspace(1,length(M),length(M))` es una matriz de que va desde el 1 al N, donde N corresponde al largo de la matriz M, representa los números de los fotogramas.

definida en MATLAB como acosd[2]. Un fotograma del eslabón en movimiento se puede ver en la Tabla 4.3.1, y el ángulo entre los eslabones en la imagen 4.3.3.

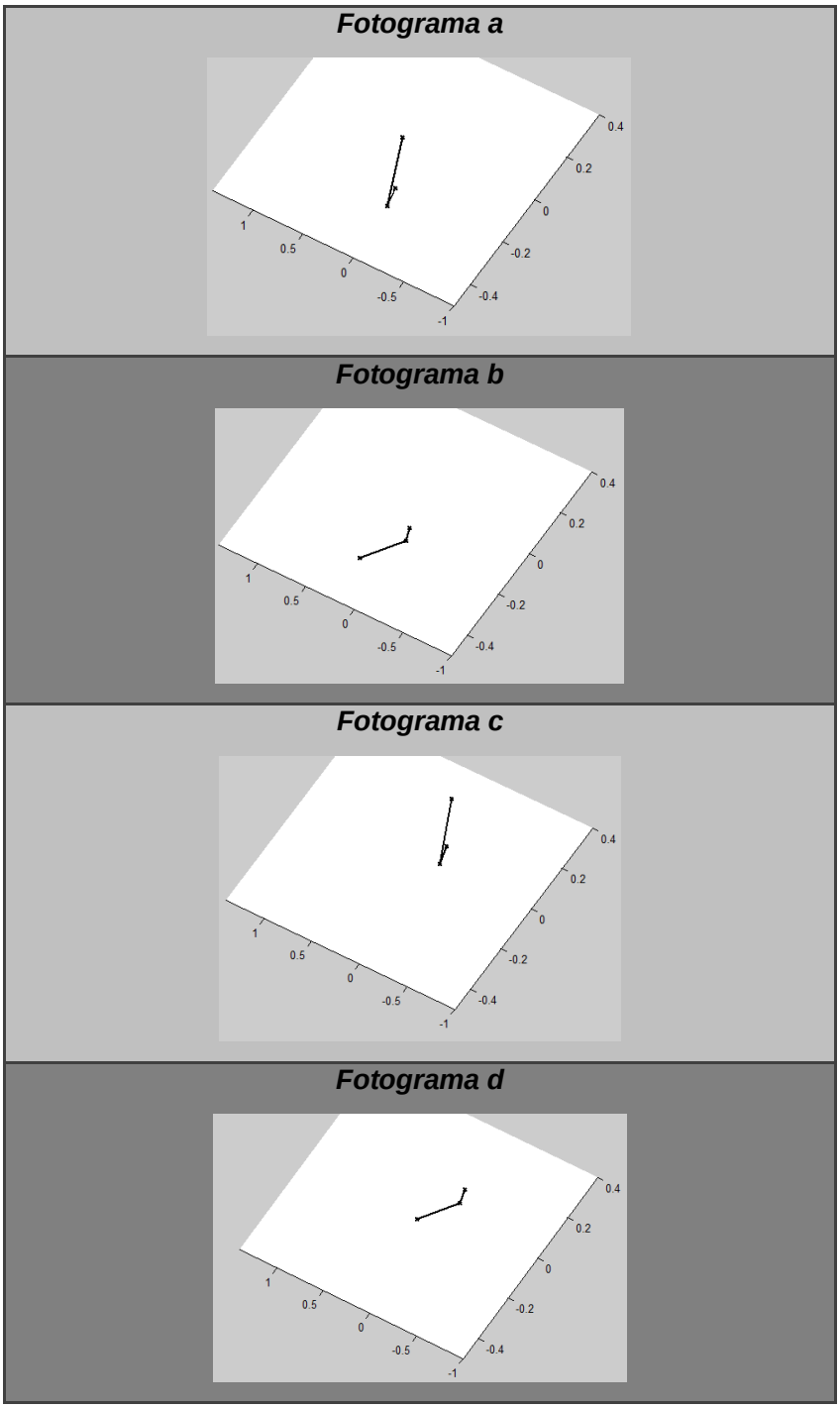


Tabla 4.3.1: Fotogramas de los eslabones avanzando en dirección positiva del eje X, en los fotogramas; a<b<c<d.

[2] Acosd es Arcoseno en grados, Acos es Arcoseno en radianes.

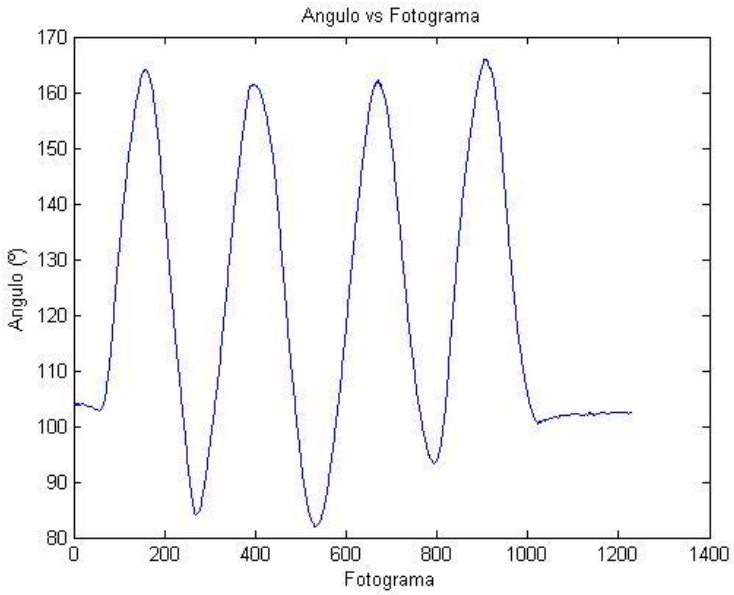


Imagen 4.3.3.: Angulo entre los eslabones respecto a los fotogramas, el inicio de la grabación comienza con el antebrazo a 90º del brazo.

4.4 Cinco partículas, tres eslabones, cálculo de polos de velocidad en 3D[i]

4.4.1 Posición de los marcadores

Para este experimente se utilizan cinco marcadores como se muestra en la imagen 4.4.1

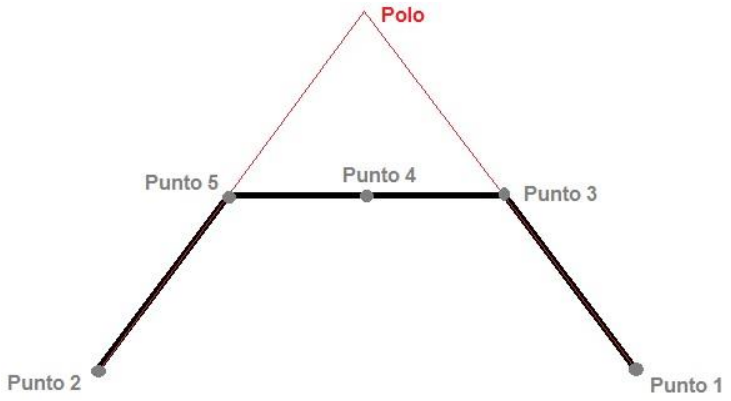


Imagen 4.4.1 Referencia de la posición de los marcadores, los puntos 3 y 5 son puntos articulados.

4.4.2 Eslabones en movimiento

Al ser cinco marcadores, la matriz total M será de NX15, para poder graficar el movimiento de este objeto, los ángulos

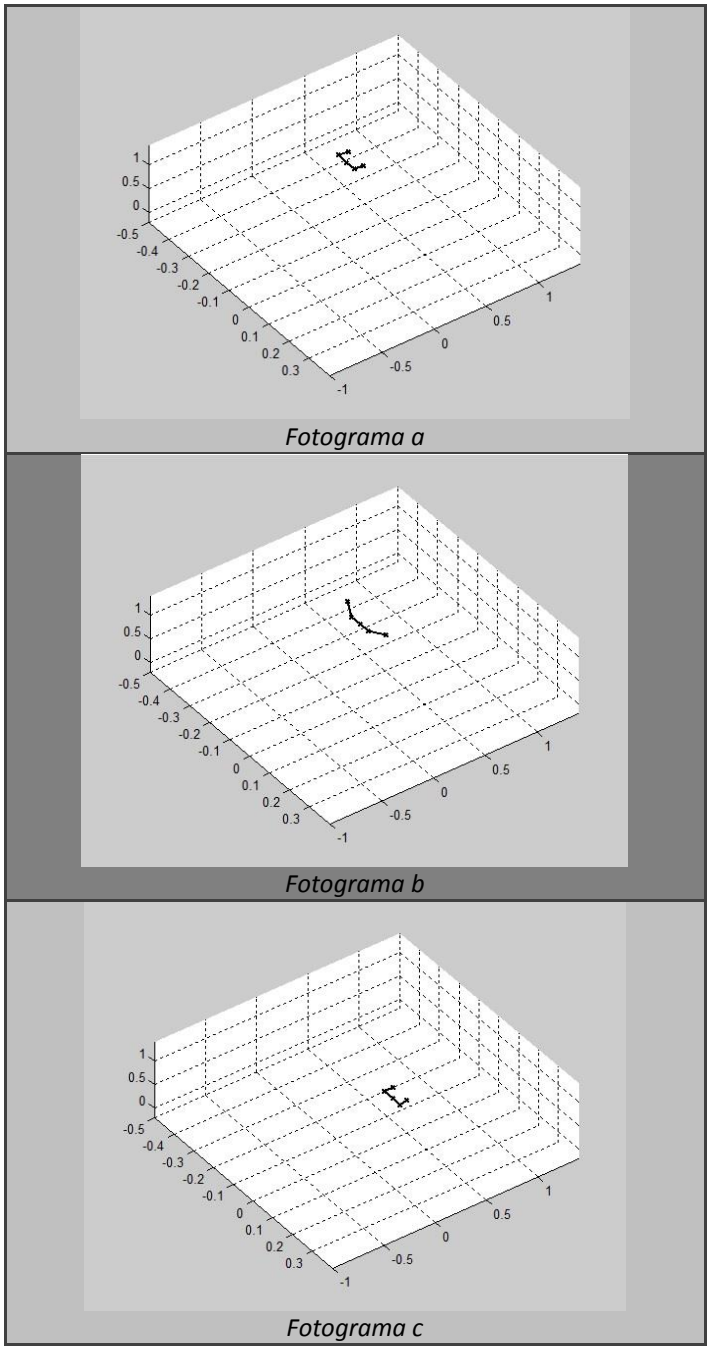
entre los eslabones 1,3 y 3,4 y entre los eslabones 2,5 y 5,4 se llama la siguiente función:

```
M;
M1=M(:,1:3); M2=M(:,4:6); M3=M(:,7:9); M4=M(:,10:12);
M5=M(:,13:15);
%se inicializan las variables para el calculo de ángulos
M11=M1-M3; M22=M4-M3; M33=M4-M5; M44=M2-M5;
M55=M5-M4; M66=M3-M4;
ANGULOS1=zeros(length(M1),1);
ANGULOS2=zeros(length(M1),1);
ANGULOS3=zeros(length(M1),1);
POLO=zeros(length(M1),1);
for i=1:length(M1)
%%%%calculo de cada ángulo

producto_Escalar=sum(M11(i,:).*M22(i,:));
producto_Modulos=sqrt(sum(M11(i,:).*M11(i,:)))*sqrt(sum(
M22(i,:).*M22(i,:)));
ANGULOS1(i)=acosd(producto_Escalar/producto_Modulos);
producto_Escalar=sum(M33(i,:).*M44(i,:));
producto_Modulos=sqrt(sum(M33(i,:).*M33(i,:)))*sqrt(sum(
M44(i,:).*M44(i,:)));
ANGULOS2(i)=acosd(producto_Escalar/producto_Modulos);
producto_Escalar=sum(M55(i,:).*M66(i,:));
producto_Modulos=sqrt(sum(M55(i,:).*M55(i,:)))*sqrt(sum(
M66(i,:).*M66(i,:)));
ANGULOS3(i)=acosd(producto_Escalar/producto_Modulos);
%%%%%fin calculo de ángulo, se guarda en el vector
Angulos

X1=M1(i,1);Y1=M1(i,2);Z1=M1(i,3);
X2=M2(i,1);Y2=M2(i,2);Z2=M2(i,3);
X3=M3(i,1);Y3=M3(i,2);Z3=M3(i,3);
X4=M4(i,1);Y4=M4(i,2);Z4=M4(i,3);
X5=M5(i,1);Y5=M5(i,2);Z5=M5(i,3);


X=[X1 X3 X4 X5 X2]; Y=[Y1 Y3 Y4 Y5 Y2];
Z=[Z1 Z3 Z4 Z5 Z2];

plot3(X,Y,Z,'k-x','LineWidth',2)
grid on

view(54,68);
xlim([-0.5 0.4]); ylim([-1 1.4]);zlim([-0.2,1.4])
pause(0.00001)
end
%%%%Grafico del ángulo vs los fotogramas
figure(1)
plot(linspace(1,length(M),length(M)),ANGULOS1)
title('Angulo vs Fotograma');
xlabel('Fotograma');
ylabel('Angulo (°)');

figure(2)
plot(linspace(1,length(M),length(M)),ANGULOS2)
title('Angulo vs Fotograma');
xlabel('Fotograma');
ylabel('Angulo (°)');

figure(3)
plot(linspace(1,length(M),length(M)),ANGULOS3)
ylim([0 360])
title('Angulo vs Fotograma');
xlabel('Fotograma');
```

```
ylabel('Angulo (°)');

figure(4)
hold on
plot3(M1(:,1),M1(:,2),M1(:,3),'red');
plot3(M2(:,1),M2(:,2),M2(:,3),'green');
plot3(M3(:,1),M3(:,2),M3(:,3),'blue');
plot3(M4(:,1),M4(:,2),M4(:,3),'black');
plot3(M5(:,1),M5(:,2),M5(:,3),'cyan');
grid on
```

Función 4.4.1 Función para animar los eslabones, graficar la trayectoria de las partículas y el ángulo entre eslabones.

El movimiento observado es como el de una "*tijera podadora*", como se puede ver en la tabla 4.4.1



Fotograma a

Fotograma b

Fotograma c

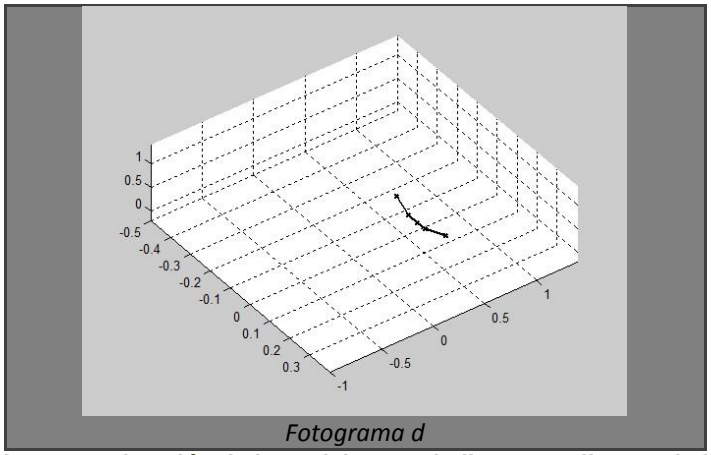


Tabla 4.4.1 Animación de los eslabones similar a una tijera podadora.

4.4.3 Trayectoria de partículas

La trayectoria de los cinco puntos se puede ver en la imagen 4.4.2:

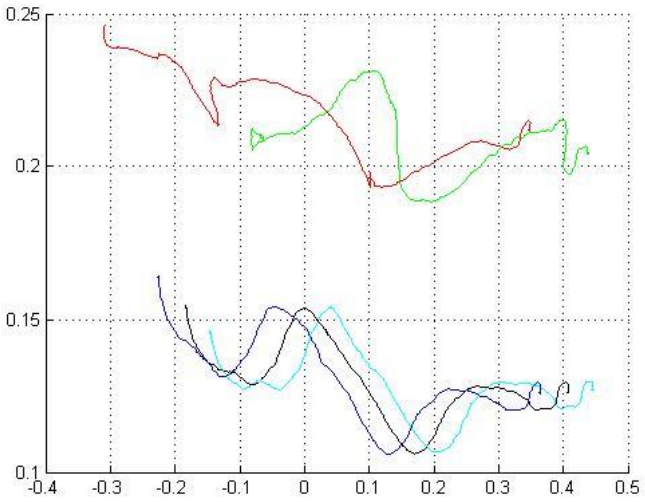


Imagen 4.4.2 Trayectoria de las partículas

Donde el rojo, verde, azul, negro y celeste corresponde a las partículas 1, 2, 3, 4, 5 respectivamente.

4.4.4 Angulo entre eslabones

Los ángulos que se generan entre los eslabones 1 y 2 (que corresponde a las partículas 1,3 y 3,4) vienen dado por la imagen 4.4.3:

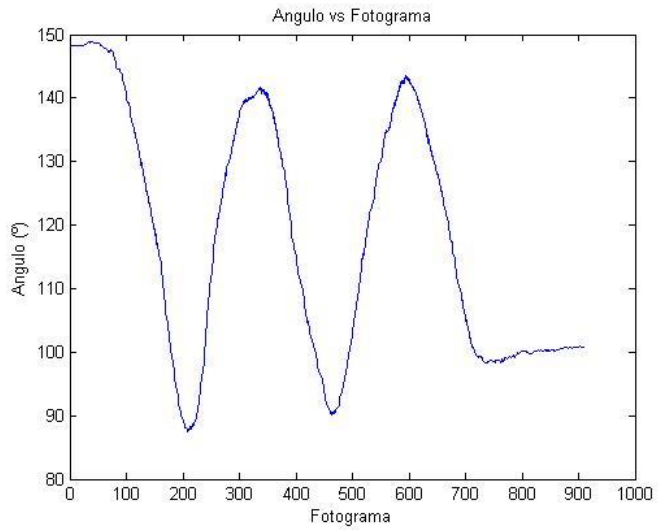


Imagen 4.4.3 Ángulo entre eslabones.

Los ángulos que se generan entre los eslabones 2 y 3 (que corresponde a las partículas 3,4 y 2,5) vienen dado por la imagen 4.4.4:

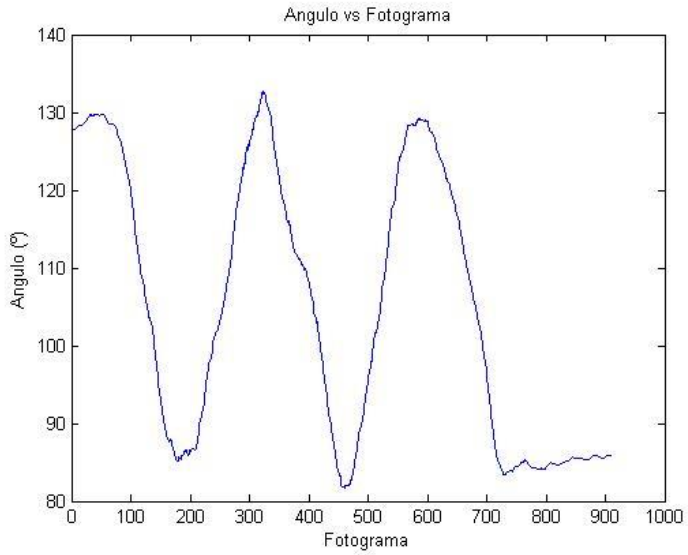


Imagen 4.4.4 Ángulo entre eslabones.

4.4.5 Error en la medición de las cámaras

Es de intuir que en el eslabón 2, constituido por las partículas 3,4 y 5 no tenga un cambio de “ángulo” puesto que es un eslabón rígido, sin embargo, al graficar el ángulo estos puntos se obtiene la imagen 4.4.5:

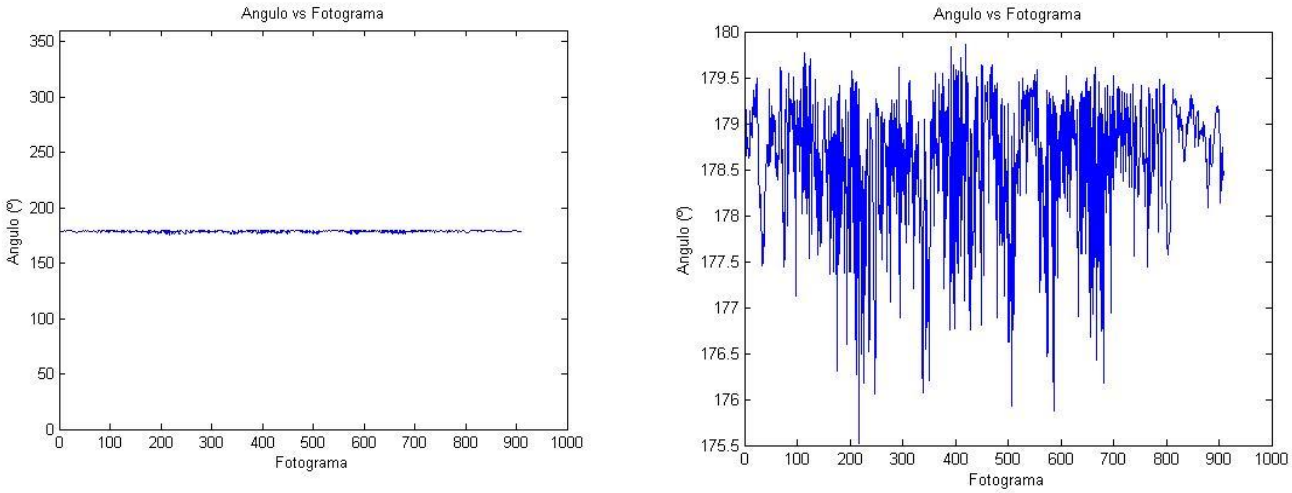


Imagen 4.4.5 Error de las cámaras.

Lo cual nos da cuenta de que existen errores de medición, si graficamos el mismo grafico más de cerca notamos que el error máximo casi alcanza los 4 grados, si se obtiene el promedio del valor absoluto de los ángulos se obtiene que:

$$\overline{X_{ABS}} = 178.59$$

Por lo tanto el error (para este caso) en promedio es de 1.41 grados. Se hace énfasis en este parte debido a que este error no proviene de la configuración de los marcadores, si no que el error esta intrínsecamente relacionado con el error de las cámaras al calcular la posición de los marcadores.

4.4.6 Cáculo de Matriz de rotación y traslación mediante el Algoritmo de *Kabsch para dos fotogramas*

Cuando se tienen tres eslabones en movimiento, se crea un centro instantáneo de movimiento y rotación, también llamado "***Polo***". Calcular polos en 2D es simple comparado al cálculo en 3D, esto se debe a que en dos dimensiones hay que extender los eslabones como dos rectas y encontrar su punto de intersección, mientras que en 3D hay que calcular una matriz de rotación y traslación. Con la matriz de rotación se puede calcular el eje de rotación y el ángulo de rotación.

El Algoritmo de Kabsch es un método para calcular la matriz de rotación más óptima que minimiza la desviación del promedio de la raíz cuadrada entre dos puntos cercanos. Este algoritmo entrega una matriz ***U*** de rotación (una matriz de 3X3),

una matriz ***r*** de traslación (3x1) y ***rmls*** que es el valor del menor promedio de la raíz cuadrada.

Este algoritmo necesita tres matrices, llamados ***P*** y ***Q***, La matriz P contiene la posición de las N partículas[3] en el fotograma f[4], mientras que Q contiene la posición de las N partículas en el fotograma f+1. El tercer vector, llamado ***m***, es un vector fila de longitud N, done ***m(i)*** es el ***peso*** a asignar a la desviación del punto i-ésimo. Si no se proporciona, se toma por defecto el ponderado, entonces:

$$m(i) = \frac{1}{N} \qquad \text{, para } i \in [1, N]$$

De tal forma que:

$$\sum_{i=1}^{N} m(i) = 1$$

El siguiente paso es calcular la matriz covarianza C, la cual está dada por:

$$C = P^T \mathrm{Q} \qquad Matriz\ covarianza$$

Con esto se puede calcular la rotación óptima U mediante la fórmula:

$$U = (C^T C)^{1/2} C^{-1} \qquad Matriz\ de\ rotación$$

Pero esta solución puede ser complicada de obtener, especialmente si la matriz C no es invertible. Por lo tanto se utiliza la función ***singular value decomposition*** **(SVD)**[ii], lo cual es una factorización de una matriz real o compleja. Matlab posee esta función incorporada, si se le llama utilizando la matriz covarianza, esta entregará tres matrices como respuesta, V, S y W, se debe invocar la función de la siguiente manera:

```
[V,S,W] = svd(C)
```

Función 4.4.2 Función para llamar SVD

A continuación, hay que decidir si hay que corregir nuestra matriz de rotación para asegurar un sistema de coordenadas de mano derecha.

[3] En el eje cartesiano x,y,z.

[4] f es un fotograma cualquiera.

$$d = sign(\det(WV^T))$$

Finalmente obtenemos nuestra matriz de rotación U:

$$U = W \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{pmatrix} V^T \quad Matriz\ de\ rotación$$[iii]

El código en Matlab que realiza estas operaciones es:

```
function[U, r, lrms] = Kabsch(P, Q, m)
    sz1 = size(P) ;
    sz2 = size(Q) ;
    if (length(sz1) ~= 2 || length(sz2) ~= 2)
        error 'P and Q deben ser matrices' ;
    end
    if (any(sz1 ~= sz2))
        error 'P and Q deben ser del mismo tamaño' ;
    end
    D = sz1(1) ;          % dimension of space
    N = sz1(2) ;          % number of points
    if (nargin >= 3)
        if (~isvector(m) || any(size(m) ~= [1 N]))
            error 'm debe ser un vector fila de largo N'
;
        end
        if (any(m < 0))
            error 'm no tiene que tener entradas
negativas' ;
        end
        msum = sum(m) ;
        if (msum == 0)
            error 'm debe contener entradas positivas' ;
        end
        m = m / msum ;      % normalizar, entonces los
pesos son igual a 1.
    else                    % si m no es entregada, usar
por defecto:
        m = ones(1,N)/N ;
    end

    p0 = P*m' ;           % el centroide de P
    q0 = Q*m' ;           % el centroide de Q
    v1 = ones(1,N) ;      % vector fila de N ones
    P = P - p0*v1 ;       % trasladamos P al origen
    Q = Q - q0*v1 ;       % trasladamos Q al origen

    % C es la matriz covarianza
    Pdm = bsxfun(@times,m,P) ;
    C = Pdm*Q' ;

%   C = P*Q' / N ;        % (para el caso sin peso)
    [V,S,W] = svd(C) ;    % singular value decomposition

    I = eye(D) ;
    if (det(V*W') < 0)    % numericamente mas estable que
usar (det(C) < 0)
        I(D,D) = -1 ;
    end
    U = W*I*V' ;
```

```
    r = q0 - U*p0 ;

    Diff = U*P - Q ;      % P, Q ya centrados

%   lrms = sqrt(sum(sum(Diff.*Diff))/N) ; % (para el
caso sin peso)

    % para calcular lrms se usa el siguiente método
eficiente:
    lrms = sqrt(sum(sum(bsxfun(@times,m,Diff).*Diff))) ;
end
```

Función 4.4.3 Función para utilizar el algoritmo de Kabsch en dos fotogramas[iv].

Este algoritmo nos permite obtener la matriz rotación U, la matriz traslación r y el lrms para N puntos en solo dos fotogramas.

4.4.7 El Algoritmo de *Kabsch para varios fotogramas*

Como se debe estudiar todos los fotogramas se llama la función 4.4.2:

```
sz=size(M); %tamaño de la gran matriz con todos
los puntos (1x2)
sz1=sz(1); %equivalente a la cantidad de
fotogramas
sz2=sz(2);% la cantidad de coordenadas para las N
particulas -> N*3 (eje x,y,z)
N=sz2/3; %cantidad de particulas
P=ones(N,3);%se crea una matriz de NX3
Q=ones(N,3);%se crea una matriz de NX3
k=1;
for j=1:(sz1-1)
    for i=1:N
        P(i,:)=[M(j,k:k+2)]; %P esta compuesto por
la posicion de N particulas en un fotograma
        Q(i,:)=[M(j+1,k:k+2)]; %P esta compuesto
por la posicion de N particulas en un fotograma+1
        k=k+3;
    end
    k=1;
    [U, r, lrms] = Kabsch(P', Q', m); %[5]
    [V,D] = eig(U-eye(3));
    traza(j)=trace(U);

end

coseno_angulo=(traza-1)./2;
angulo=acosd(coseno_angulo);
```

Función 4.4.4 Función para utilizar el algoritmo de Kabsch en todos los fotogramas, además en las últimas líneas se calcula el eje y ángulo de rotación.

[5] Observe que está escrito P' y Q' y no P y Q, la comilla representa la matriz traspuesta.

El cual demora medio segundo en procesar cinco partículas para mil fotogramas.

4.4.8 Calculo del eje de rotación y ángulo de rotación

Las últimas líneas de la función 4.4.4 tratan la siguiente parte a tratar, una vez obtenidas la matriz de rotación y traslación es necesario encontrar el eje de rotación del objeto.

Para encontrar el eje de rotación de un objeto, la matriz de rotación U debe cumplir la siguiente condición.

$$Uu = u$$

Donde u es el vector del eje de rotación del objeto.

Lo anterior puede ser escrito de la siguiente manera:

$$Uu = Iu$$

Donde I es la matriz identidad, en este caso de 3x3 y puede ser invocada en Matlab con la función eye(3).

Finalmente para encontrar el vector u observamos que la ecuación anterior puede ser escrita de la siguiente manera:

$$(U - I)u = 0^{v}$$

Donde u tendría que ser el *Kernel* de (U-I), y visto de otro manera u es el vector propio de U correspondiente al valor propio λ=1.

Observamos que para una matriz U de 3x3 la máxima cantidad de vectores que pueden cumplir la condición de ser el Kernel es de tres vectores, necesitaremos sólo un vector aunque en las líneas de código se ha puesto una solución para dos vectores. El código es el siguiente:

```
[V,D] = eig(U-eye(3));
```

Función 4.4.5 Función para cálculos los vectores propios.

Donde se invoca la función eig de Matlab (eigenvector / vector propio). V y D son los vectores que cumplen la condición y dado que son vectores con decimales, la solución puede traer números complejos.

La manera más simple de encontrar el ángulo de rotación es calcular la traza del vector de rotación U, lo cual equivale a:

$$Traza(U) = 1 + 2\cos(\theta)$$

De esta manera se obtiene que:

$$\theta = arcos\left(\frac{Traza(U) - 1}{2}\right)$$

5 Conclusiones

Motive permite obtener resultados rápidos gracias a la facilidad de manejo y exportación de datos, los esencial entregado por este software son las coordenadas de las posiciones de cada marcador, una vez obtenidas se puede realizar cualquier tipo de trabaja en MATLAB.

MATLAB permite un fácil manejo de matrices logrando realizar una gran variedad de simulaciones, dentro de la que destaca el reproducir el movimiento realizado frente a las cámaras Optitrack.

6 Bibliografía

[i] http://www.mathworks.com/matlabcentral/newsreader/view_thread/314432
[ii] http://en.wikipedia.org/wiki/Singular_value_decomposition
[iii] http://en.wikipedia.org/wiki/Kabsch_algorithm

[iv] http://www.mathworks.com/matlabcentral/fileexchange/25746-kabsch-algorithm

[v] http://en.wikipedia.org/wiki/Rotation_matrix