ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO DE TELECOMUNICACIÓN

Título del proyecto:

PLATAFORMA DE SIMULACIÓN PARA ALGORITMOS DE ESTIMACIÓN DE LA POSICIÓN 3D: ESTUDIO DE VARIABILIDAD

Cristian Ordoyo Casado

Tutores: Rafael Cabeza Laguna

Mikel Ariz Galilea

Pamplona, 12 de Septiembre de 2014

1 ÍNDICE

2. Introducción

Las aplicaciones de Eye-Tracking y de Gaze-Estimation están siendo el objetivo de diversas vías de investigación en los días en los que nos encontramos ya que se tratan de técnicas muy poco invasivas y que pueden proporcionarnos mucha información o posibilidad de control de cualquier dispositivo de una forma totalmente autónoma y sin prácticamente entrenamiento.

Una de las líneas de investigación del departamento de Ingeniería Eléctrica y Electrónica de la Universidad Pública de Navarra se centra en determinar con exactitud la posición de la cabeza de una persona y tras ello estimar la mirada de ésta. Es por ello que el presente trabajo de investigación se enmarque en esta línea de trabajo.

Estimar la posición y orientación de un sujeto no es tarea sencilla y menos aún en condiciones no controladas. El presente trabajo toma importancia ya que ofrecerá la posibilidad de determinar la estimación del error de forma rápida y con sólo el uso de un ordenador. A partir del conocimiento del error esperado, podremos determinar si en un caso real estamos obteniendo la mayor precisión posible o tenemos diversos factores que dificultan dicho objetivo.

Por ello se va a implementar un simulador en MatLab que nos ofrezca la posibilidad de realizar experimentos de forma ágil y precisa para determinar qué afecta a dichas estimaciones y valorar de antemano cuales deben de ser las mejores opciones para solucionar posibles problemas conocidos o que queden por conocer.

A lo largo del proyecto se hablará de valores admisibles de estimación, estos son los que se han venido obteniendo en las pruebas realizadas en el laboratorio y que en rotación están sobre los dos grados en promedio de error.

3 Aplicaciones

La estimación de la posición y orientación de un objeto es el origen de numerosas aplicaciones, alguna de ellas serán enunciadas a continuación.

3.1 Estimación de la mirada

Para la estimación de la mirada (Gaze estimation) es necesario conocer previamente la posición de la cabeza y a partir de esta de los ojos. La principal aplicación de este sistema se encuentra en la posibilidad de interacción con una máquina sin contacto físico. Esto es por ejemplo el control de un ordenador utilizando únicamente la mirada, muy interesante en el ámbito global y concretamente en personas discapacitadas a las cuales les permite el uso de sistemas los cuales sin estas herramientas no son posibles.

3.2 Aplicaciones médicas

En el ámbito médico cada vez más sistemas estiman con precisión la posición de instrumentos o radiofármacos los cuales permiten acelerar el tratamiento o intervención del médico. Además evitan un tratamiento intrusivo que alargue los plazos de recuperación del paciente.

3.3 Videojuegos y cine

En el sector del ocio se está llevando a cabo diferentes usos de la estimación de la posición de un usuario y la tendencia en el mundo de los videojuegos es realizarlo sin un mando que controle los movimientos, si no que el propio usuario con sus movimientos interactúe y disfrute sin necesidad de nada más que su propio cuerpo.

En el cine se está aplicando en post-producción y realidad virtual en películas de animación, mediante lo cual se puede asignar al personaje animado de unos rasgos faciales muy realistas.

4 Objetivos

El primer objetivo de este trabajo será la realización de un simulador en MatLab escalable e intuitivo, el cual presente un código limpio y cualquier trabajo posterior pueda basarse en él. Se implementará una simulación de un modelo 3D frente a una cámara web monocular y diferentes funciones que modifiquen el estado ideal del experimento, dando valores más realistas y pruebas objetivas de lo que ocurriría en una aplicación real.

Posteriormente se utilizará dicha implementación para obtener resultados de la estimación de posición y orientación con diferentes modelos de cabeza y de ruidos 2D, 3D, así como con variaciones de modelos y pruebas realistas obtenidas de experiencias previas usando sistemas de *tracking*.

El algoritmo de estimación de la posición y orientación a evaluar será POSIT, en sus diferentes implementaciones (ClassicPosit, ModernPosit y SoftPosit).

5 Fundamentos teóricos

5.1 POSIT algorithm. D. Dementhon [1]

POSIT es un algoritmo rápido e iterativo para la estimación de la pose 6D (translación y rotación) de un objeto respecto a una cámara monocular con los puntos del objeto bajo un sistema de coordenadas. Para poder utilizarlo es necesario que se tengan cuatro o más puntos del plano imagen y que estos puntos no sean muy coplanares.

Combina dos algoritmos, el primero con el cual aproxima la proyección perspectiva del objeto y determina la translación y rotación del objeto (Pose from Orthography and Scaling) y el segundo POSIT (POS with ITerations). Esto es utilizado para a través de varias iteraciones obtener una proyección con menor error.

Probaremos las implementaciones de D. Dementhon del classicPosit, modernPosit y softPosit.

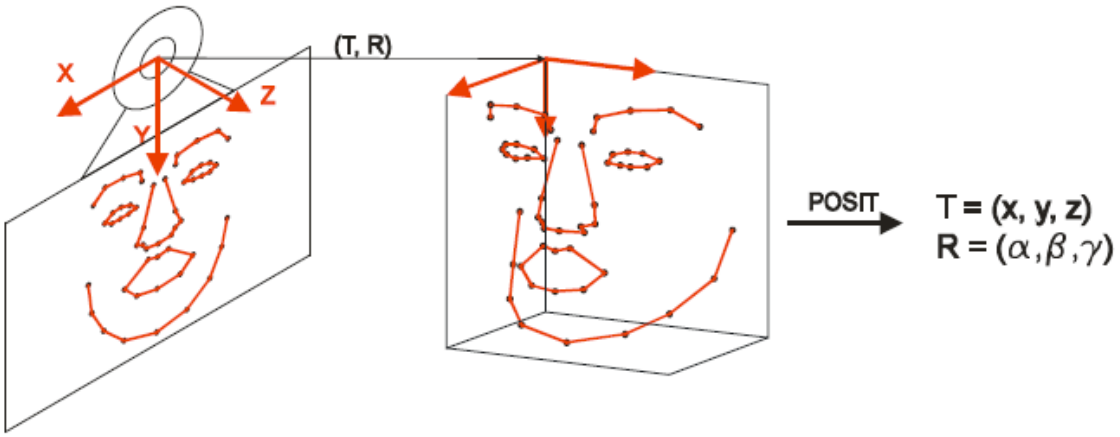


Figura 1. Esquema funcionamiento de POSIT

5.1.1 ClassicPosit y ModernPosit [1]

Estas dos implementaciones de POSIT no necesitan una estimación inicial de la *pose* pero si necesitan la relación de correspondencia de los puntos de la imagen 2D y el modelo 3D. La precisión que nos ofrecen es alta y además ambas pueden ser utilizadas en tiempo real ya que el tiempo de computación es muy reducido.

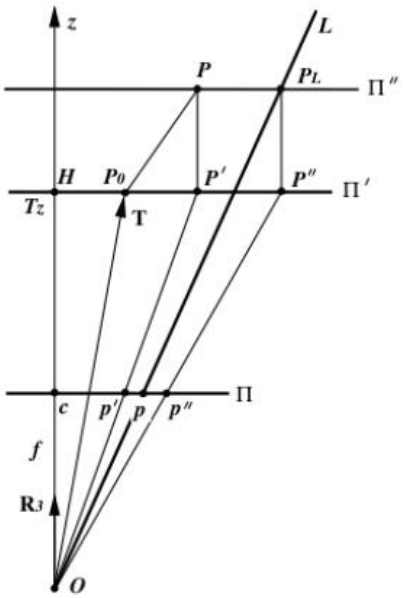


Figura 2. Interpretación geométrica de POSIT

La principal diferencia entre ambos algoritmos radica en la elección del origen de coordenadas del modelo, respecto al cual estima la distancia. ClassicPosit si no se escoge ningún punto como origen elige el primer punto del conjunto, mientras que ModernPosit lo hace respecto al origen de coordenadas del modelo 3D. Por tanto tendremos que tener en cuenta esto para en el caso de comparar ambos algoritmos recalcular la estimación dada respecto al origen de coordenadas del modelo.

5.1.2 SoftPosit [2]

Esta implementación de POSIT va a presentar la ventaja respecto a las dos anteriores explicadas de que no es necesario conocer la relación entre los puntos de la imagen y del modelo. En cambio necesita de una estimación de la *pose* inicial y también necesita de un tiempo de computación superior que actualmente no permite su utilización en tiempo real en un ordenador de prestaciones media-alta.

Este algoritmo se inicia con una suposición de que las matrices iniciales de puntos son aleatorias, para ir corrigiendo el error iterativamente

5.2 Aplicación de pesos a Posit

La aplicación de pesos a Posit consiste en dar un mayor valor de repetividad a los puntos con menor error respecto a su *pose* ideal. Esto es aplicar redundancia a dichos puntos, incluso dividiendo el nivel de redundancia en varios grados, dependiendo del grado de confianza (precisión) del punto. Lo que se pretende con esta técnica es reducir el error de estimación ofreciendo al algoritmo Posit mayor cantidad de puntos correctos y minimizando los puntos que son considerados "erróneos".

5.3 IntraFace[3]

IntraFace es un novedoso programa desarrollado por equipos de trabajo del Carnegie Mellon y la Universidad de Pittsburgh. Está disponible públicamente para propósitos de investigación y se encuentra pendiente de patente.

Este algoritmo está implementado en C++ y MatLab y detecta puntos característicos en la cara (cejas, ojos, nariz y boca) y también analiza expresiones faciales (sorpresa, alegría, tristeza…).

Para este proyecto resulta interesante el marcado de puntos del tabique nasal y se realizarán varias pruebas para comprobar si a efectos de estimación de la posición y rotación añade efectividad y precisión.

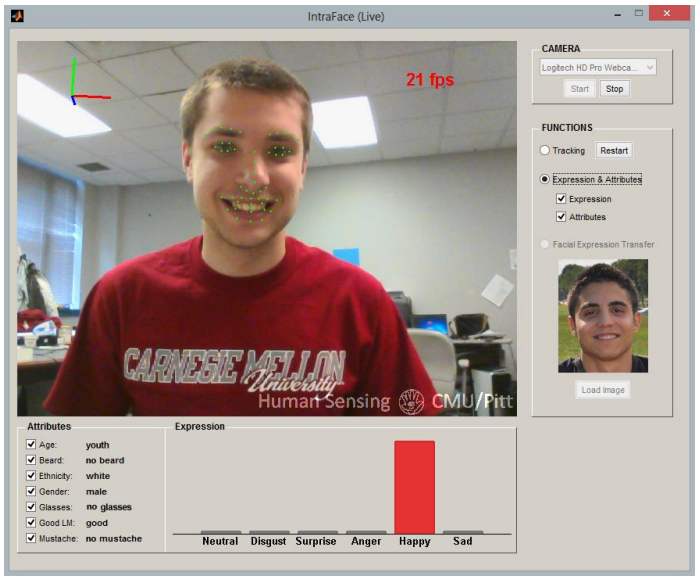


Figura 3. Ejemplo de uso de IntraFace

5.4 Hidden Point Removal (HPR) [4]

Algoritmo simple y rápido que determina los puntos visibles de una nube de puntos desde un determinado punto de vista. Esto es llevado a cabo sin necesidad de reconstruir la superficie que puede ser un reto complicado de realizar sobre todo si no hay una nube de puntos muy densa.

El algoritmo se compone por dos pasos: inversión y construcción de la envolvente convexa.

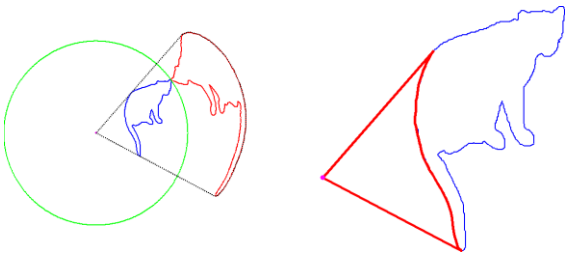


Figura 4. Inversión aplicada por HPR

5.5 Distribución normal. Normrnd.m

Se genera una distribución normal o gaussiana, una de las distribuciones de probabilidad que con más frecuencia aparece en fenómenos reales. Vendrá dada por la función:

$$\Phi_{\mu,\sigma^2}(x) = \int_{-\infty}^{x} \varphi_{\mu,\sigma^2}(u)\,du$$
$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(u-\mu)^2}{2\sigma^2}}\,du, \quad x \in \mathbb{R}$$

Explicada de forma gráfica, tenemos que al 68.2% de los puntos se le añadirá un ruido en el intervalo $(\mu, \pm\sigma)$.

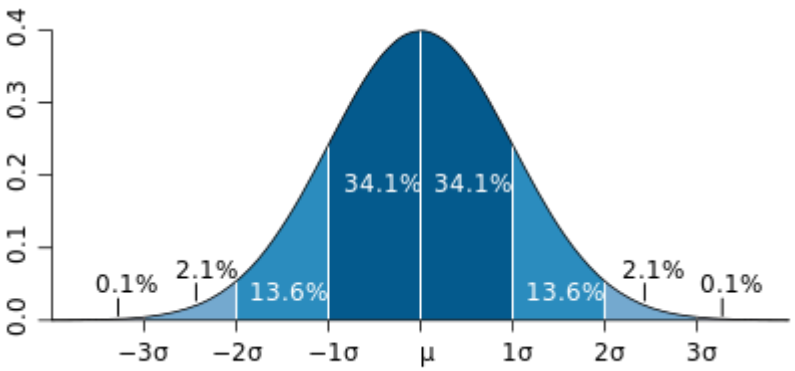


Figura 5. Representación gráfica de la distribución normal

5.6 Ruido derivado de la rotación 3D

Para determinar la cantidad de ruido a añadir a cada punto hemos implementado una función que determina el ángulo que hay entre el vector normal al punto y la recta que une el punto con el centro de proyección. Este ángulo va a estar comprendido entre ±90º y será utilizado para determinar la desviación típica que genere el ruido aleatorio con media nula.

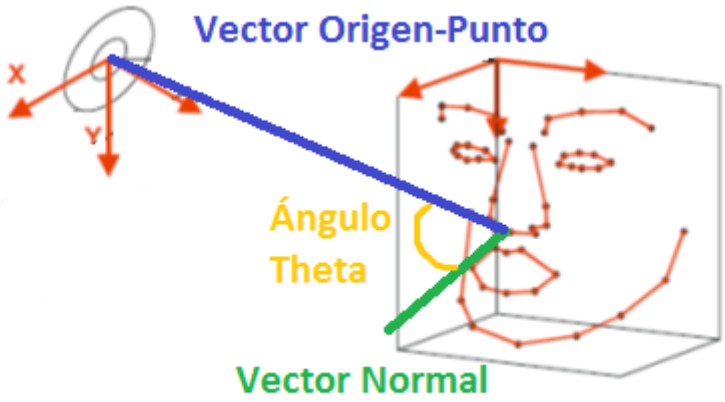


Figura 6. Esquema ruido derivado de la rotación 3D

El ángulo vendrá dado por la ecuación siguiente:

$$\theta = \cos^{-1}\frac{\boldsymbol{Vector\ Normal} \cdot \boldsymbol{Vector\ Origen - Punto}}{|\boldsymbol{Vector\ Normal}| \cdot |\boldsymbol{Vector\ Origen - Punto}|}$$

5.7 Cámara web. Logitech HD pro Webcam C920

Para el desarrollo de este proyecto únicamente son importantes los valores de calibración de la cámara, los cuales se detallará como se obtienen más adelante.

5.8 Rotación en ángulos. GetEulerAngles.m

A continuación se muestran los ángulos de giro elegidos y las matrices de rotación para cada punto.

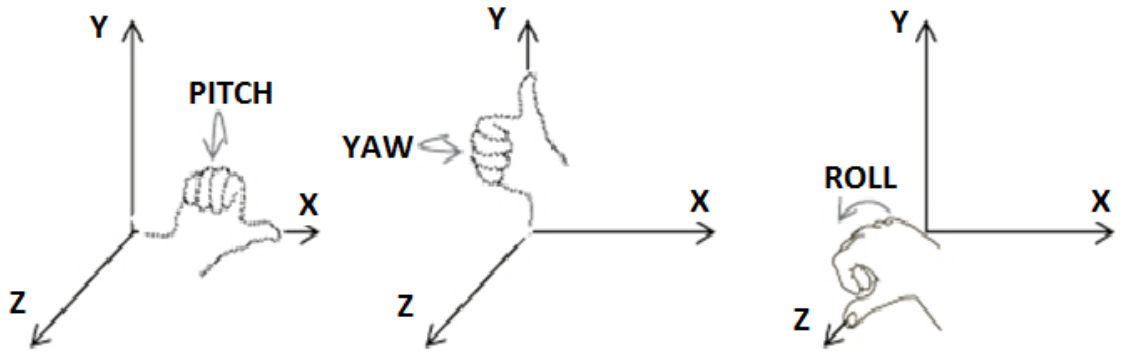


Figura 7. Ángulos de giro 3D

$$ROT_{ROLL} = \begin{bmatrix} Cos(\varphi) & Sen(\varphi) & 0 & 0 \\ -Sen(\varphi) & Cos(\varphi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ROT_{YAW} = \begin{bmatrix} Cos(\theta) & 0 & -Sen(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ Sen(\theta) & 0 & Cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$ROT_{PITCH} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos(\gamma) & Sen(\gamma) & 0 \\ 0 & -Sen(\gamma) & Cos(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz de rotación del objeto será $M_{ROT} = ROT_{ROLL} * ROT_{YAW} * ROT_{PITCH}$

Muy importante el orden de multiplicación ya que en otro orden no estaremos calculando la rotación del objeto correctamente.

5.9 Proyección en el plano imagen. Project_points3.m

Esta función contenida en el toolbox_calib de MatLab nos permite a partir de la nube de puntos 3D del modelo, los parámetros de una cámara, la relación de translación y rotación entre el modelo 3D y el centro de proyección obtener la proyección 2D en el plano imagen. El modelo físico de cámara implementada corresponde con una cámara Pinhole a la cual se le añade la distorsión radial y tangencial correspondiente al modelo de cámara elegido para el desarrollo de este trabajo.

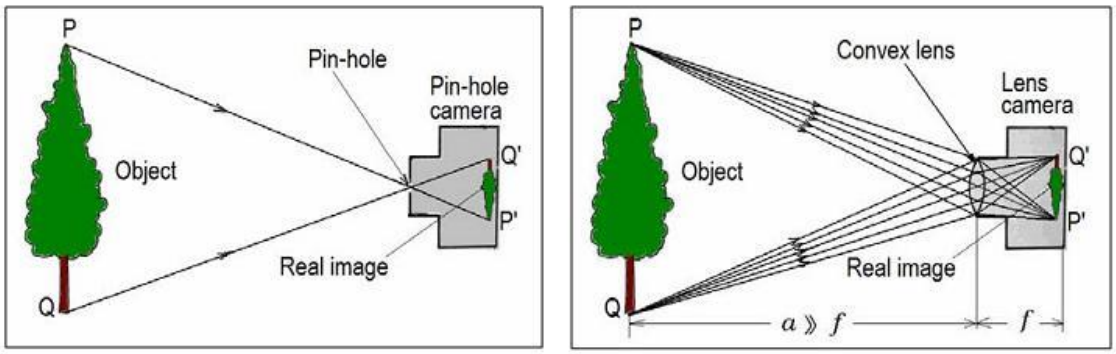


Figura 8. Proyección Pin Hole (Izq.) - Proyección de una lente normal (Dcha.)

En el proceso de obtener la proyección en el plano imagen del objeto es necesaria realizar la transformación rígida en translación y rotación entre el objeto y la cámara. La cual vendrá determinada por la matriz de cambio de coordenadas siguiente:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} & Matriz\ Rotación & & T_x \\ & [3x3] & & T_y \\ 0 & 0 & 0 & T_z \\ & & & 1 \end{bmatrix} * \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{bmatrix}$$

5.10Función dsearchn [5]

Se deseaba determinar cuál era el punto más cercano a una determinada nube de puntos, concretamente el punto más cercano de los puntos visibles. Para ello se ha utilizado está función en su versión más ligera computacionalmente. Dicha versión busca el punto más próximo sin la necesidad de realizar una triangulación la nube de puntos. Además nos va a proporcionar el valor de la distancia entre el punto inicial y el punto final obtenido.

5.11Modelos 3D

5.11.1Modelo Dasha

Se trata de una cabeza-maniquí de tamaño y apariencia humana con relieve como se puede apreciar en párpados y labios.

Figura 9. Modelo Dasha

5.11.2Modelo de los Suizos. Basel Face Model (BFM)

El *Basel Face Model* (BFM) publicado por el departamento de Matemática y Ciencias de la Computación de la Universidad de Basel, es un Modelo de caras deformables 3D construido a partir de 100 ejemplos masculinos y 100 ejemplos femeninos de caras. En nuestro proyecto lo utilizaremos para generar diferentes modelos de usuario 3D.

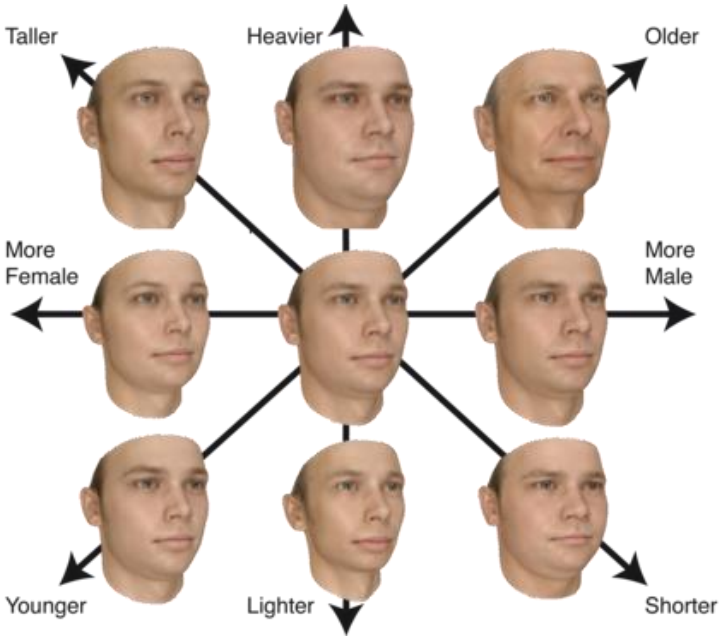


Figura 10. Ejemplos de variabilidad de caras

5.11.3Modelo cilíndrico

Se trata de una aproximación a una cabeza humana donde los puntos característicos forman parte de la cara lateral de un cilindro. Dicha aproximación se puede conseguir generando un cilindro de dimensiones realistas y proyectando en su cara lateral los puntos característicos de una cara humana. En nuestro caso se proyectará tanto Dasha como el modelo BFM.

6 Desarrollo del trabajo

Para el desarrollo del trabajo de investigación se necesita una preparación previa, consistente en conocer los dos elementos físicos que debemos virtualizar: la cámara y el modelo de cabeza o usuario.

La virtualización de la cámara consistirá primeramente en determinar sus parámetros intrínsecos y extrínsecos mediante un proceso de calibración. Posteriormente para obtener el plano imagen nos servirá la aplicación de la función *"project3points.m"*

Para virtualizar el modelo de Dasha se ha procedido a su escaneado con un láser para así poder obtener una nube de puntos 3D que es la realmente interesante desde el punto de vista de este trabajo. Otro de los modelos que utilizaremos es un promediado estadístico de varios usuarios que ya se encuentra virtualizado. En ambos se procederá también al marcado de los puntos característicos que se desean estudiar.

6.1 Calibración de la cámara

Para la calibración de la cámara se ha usado el Camera Calibration Toolbox for Matlab de Jean-Yves Bouguet el cual calcula los parámetro intrínsecos y extrínsecos de la cámara utilizando un grupo de imágenes.

Para ello se capturan grupos de 20 imágenes con la cámara fija de un damero con número de cuadros y dimensiones conocidas en diferentes posiciones sobre los ejes X, Y, Z y rotaciones sobre estos ejes. Se determina las esquinas del damero y el Toolbox determina las esquinas del damero a partir de las dimensiones de cada cuadro y del número de cuadros en horizontal y vertical. Si en algún caso comete un error, se puede modificar manualmente. De esta forma el Toolbox determina la distancia y rotación real entre la cámara y el plano (damero). Interesa tener imágenes con varias rotaciones y posiciones para obtener unos resultados más exactos. Finalmente se obtienen los parámetros intrínsecos de la cámara: distancia focal, punto principal, coeficientes de inclinación que definen el ángulo entre los pixeles de los ejes x e y, y sus distorsiones.

También se obtienen los parámetros extrínsecos a la cámara: rotación y translación del origen de cada plano (damero) al sistema de referencia de la cámara.

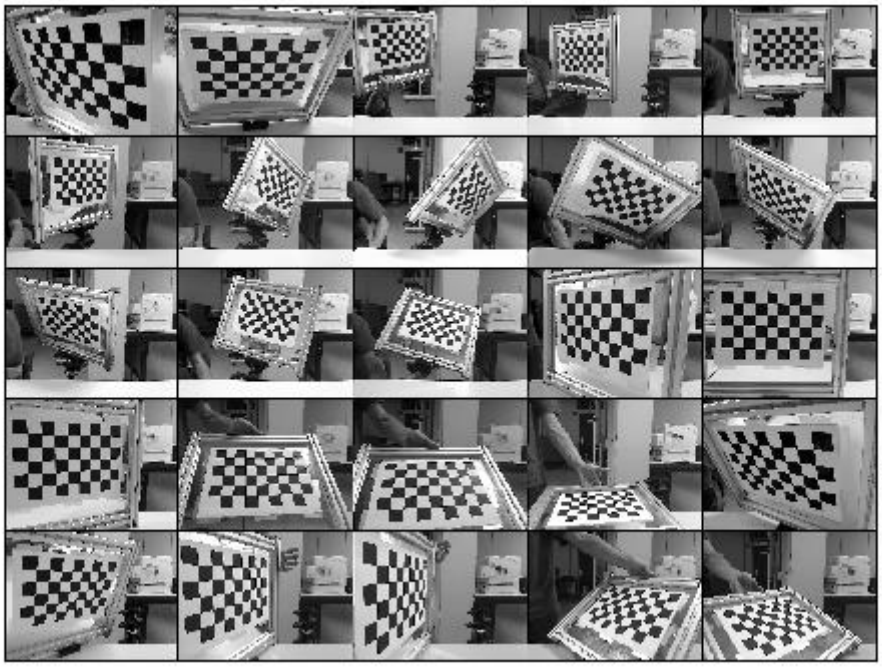


Figura 11. Conjunto de imágenes utilizadas en la calibración

6.2 Modelo Dasha

Se procedió al escaneado 3D de dicho maniquí mediante el software David LaserScanner. El proceso detallado se puede seguir en el Proyecto Fin de Carrera realizado por David Roncal. [6]

Tras el escaneado y procesado de la nube de puntos, un aspecto a tener en cuenta era determinar adecuadamente el origen del sistema de coordenadas, ya que nuestra estimación va venir determinada respecto de dicho origen por lo que es vital establecerlo nosotros mismos.

Se elige, para los modelos usados en el laboratorio, situarlo dentro de la cabeza a la altura de la nariz y en la sección de las orejas.

6.3 Modelo 3D Basel Face Model (BFM)

El modelo 3D *Basel Face Model*, (a partir de ahora lo llamaremos BFM), publicado por el departamento de Matemática y Ciencias de la Computación de la universidad de Basel (Suiza), es un modelo escaneado 3D de 100 caras masculinas y otras 100 caras femeninas de diferentes edades. Para nuestro estudio utilizaremos el promedio de esas 200 caras, de esta forma creemos adecuado utilizarla como modelo para la estimación de la posición y orientación de la cabeza.

Al igual que con el modelo de Dasha ha sido necesario el marcado de los puntos característicos y la colocación del origen de coordenadas.

6.4 Modelo Cilíndrico

El modelo cilíndrico es una aproximación teórica de una cabeza humana, en la cual se proyectarán los puntos característicos de un usuario. Realmente se puede aproximar una cabeza humana a la cara exterior de un cilindro con base elíptica. Esto nos permite con únicamente 3 variables, el eje mayor y menor del cilindro y la altura, conseguir diferentes nubes de puntos 3D de usuarios.

6.5 Marcado de puntos

Se deben determinar unos puntos característicos de la cara, denominados landmarks. Estos puntos son los realmente importantes en la estimación de la “pose”. En el trabajo se han determinado los siguientes 54 puntos, a los que se han añadido 4 del tabique nasal. El marcaje no es algo baladí porque lo que se para el modelo de Dasha se utilizó un sensor magnético con 6 grados de libertad y para el modelo BFM se procedió a un análisis exhaustivo del punto característico. Se pueden ver detallados los procesos de marcaje en los proyectos realizados por David Roncal y José Javier Bengoechea respectivamente. [7]

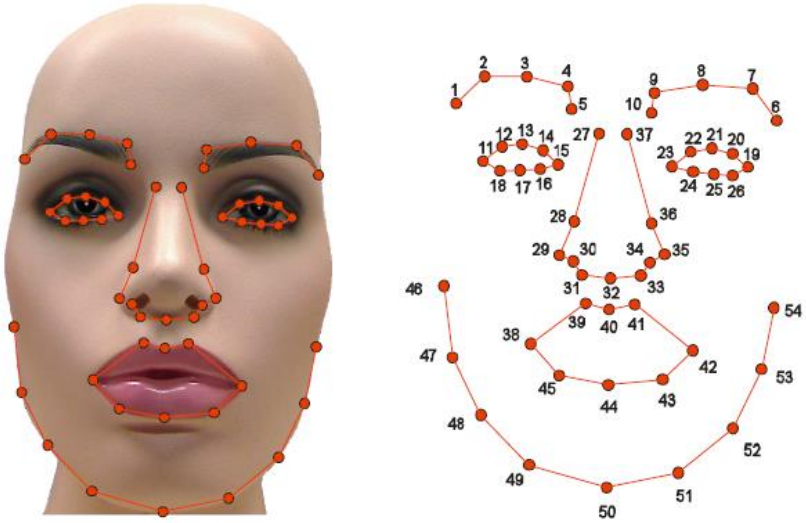


Figura 12. Marcaje de 54 puntos en Dasha

6.6 Simulador

Una vez disponemos de los modelos 3D y la calibración de la cámara podemos empezar con la plataforma de simulación. En ella se implementarán las diferentes modificaciones o funciones que nos permitan reproducir los diferentes estudios que se desean hacer. Además hemos considerado conveniente el establecer una base de videos que sirva para todas las pruebas y de esta forma conocer la posición y orientación del modelo de forma exacta.

6.6.1 Descripción de videos

Se van a realizar las pruebas con unos videos ya definidos para la grabación de la base de videos de la UPNA y que son los que están siendo usados en las pruebas del laboratorio. Esto significa que se van a simular 12 videos, de los cuales 6 tienen un grado de libertad (traslaciones y rotaciones puras) y los restantes vídeos son libres, lo que quiere decir que combinan movimiento en los 6 grados de libertad.

El sistema de coordenadas de la cámara será:

- Eje X: horizontal, positivo a hacia la derecha.

- Eje Y: vertical, positivo hacia abajo.
- Eje Z: profundidad, positivo alejándose.

Los ángulos de giro vendrán determinados de la siguiente manera:

- Roll: giro de cabeza lateral, intentando tocar los hombros con las orejas.
- Yaw: giro de cabeza hacia los lados, gesto de negación.
- Pitch: giro de cabeza vertical, gesto de afirmación.

Figura 13. Definición de los sistemas de coordenadas

Los vídeos tendrán una duración de 10 segundos, lo que suponen 300 frames, si son grabados con una frecuencia de adquisición de 30 frames por segundo. La resolución que emplearemos para ellos también será de 1280x720 pixeles, lo que mantiene una relación de aspecto de 16:9.

Video 01	Translación en el eje X pura, movimiento izquierda-derecha
Video 02	Translación en el eje Y pura, movimiento arriba-abajo
Video 03	Translación en el eje Z pura, movimiento acercarse-alejarse
Video 04	Rotación Roll pura, balanceo lateral de la cabeza
Video 05	Rotación Yaw pura, negación con la cabeza
Video 06	Rotación Pitch pura, afirmación con la cabeza
Video 07 - 12	Movimientos libres con combinaciones de los 6 movimientos anteriores.

6.6.2 Implementación del bloque principal

En este bloque se van a detallar la funcionalidad básica de esta plataforma de simulación. Esto engloba la carga del modelo, de la cámara, la generación del movimiento, la proyección 2D del modelo y la estimación de su posición y orientación.

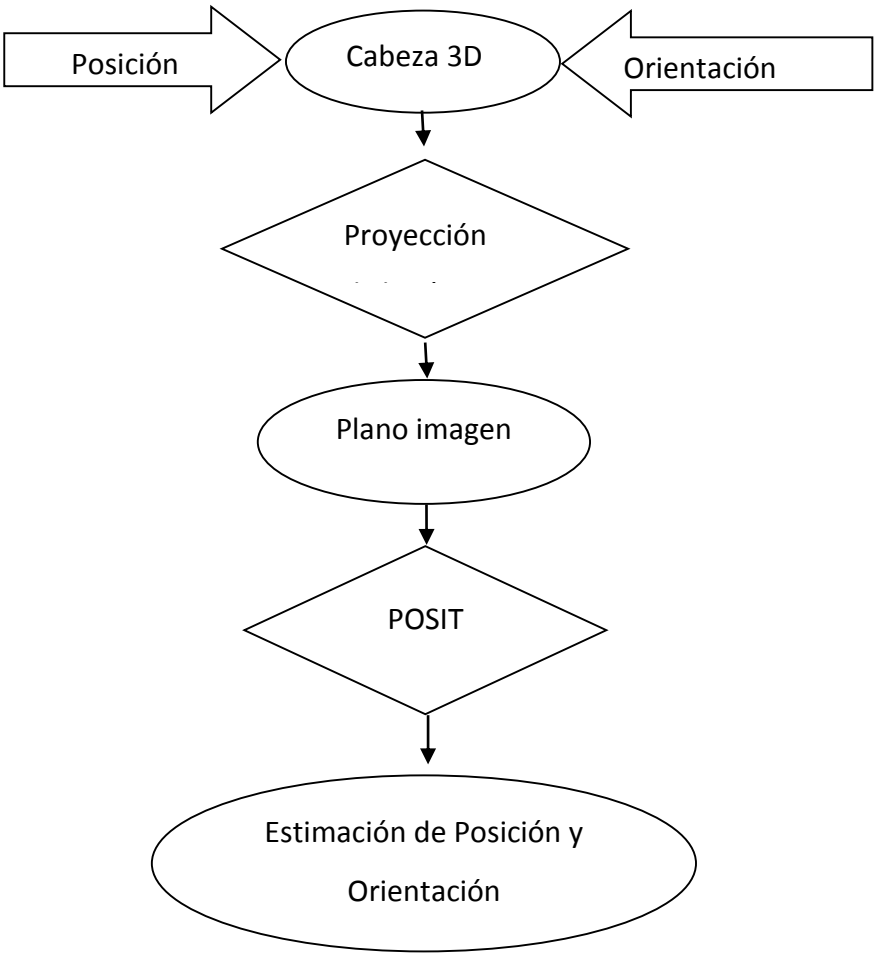


Figura 14. Esquema de funcionamiento global del simulador

- **Selección de modelo**

En vista a ampliar las posibilidades de combinación de diferentes modelos se va a permitir seleccionar el modelo de cabeza (*model_choice*), los puntos característicos del modelo (*worldpoints_choice*) y los puntos del usuario a seguir (*points_ choice*). Esto nos va a permitir el juego de combinaciones en las pruebas posteriores.

Además se debe establecer la posición y orientación en cada frame de la cabeza, esta información será proporcionada por la función *checkpoints.*

- **Proyección 2D**

Los parámetros de calibración de la cámara ya han sido incluidos en el código, y el único dato que puede afectarle es la resolución a utilizar en los videos. Esto es debido a que para un correcto funcionamiento del sistema se deben auto-ajustar los valores entre la resolución de calibración de la cámara (1920x1080 pixeles) y los valores de resolución de video.

Posteriormente la función *project_points3* obtendrá el plano imagen correspondiente a la escena y todavía hay una función más implicada, que es la encargada de determinar si los puntos proyectados se encuentran en la resolución de video (pantalla) o fuera de ella y es necesario eliminarlos. Esta función es *camara_resolucion.*

- **Estimación de la *pose***

Como ya se ha dicho en este bloque el algoritmo con el que vamos a trabajar es POSIT.

6.6.3 Implementación de funcionalidad

- **Pérdida generada por el *tracker***

Puede ser simulado el efecto de pérdida o "enganchamiento" de puntos durante un *tracking*. Esto puede ser determinado por el usuario si quiere perder algún punto en algún frame determinado o puede hacerse de forma más automática permitiendo que el algoritmo HPR determine qué puntos están siendo ocluidos y en cada momento. Además a los datos proporcionados por el HPR se pueden ofrecer simulaciones donde esos puntos no desaparecen realmente sino que se van desplazando en el rostro del usuario.

- **Recuperación de puntos**

Durante el transcurso de un video algún punto de seguimiento puede perderse, ya sea por oclusión o por un error forzado en este caso. Se habilita un comando que permite recuperar puntos en cada frame o no recuperar ningún punto perdido a lo largo del video. Esto se realiza de manera sencilla habilitando todos los puntos al inicio del procesamiento de cada frame.

- **Aplicación de pesos a Posit**

Se podrá optar por aplicar pesos en cada simulación o mediante un script una vez obtenidos todos los videos deseados. En este trabajo he optado por usar la versión offline ya que se busca determinar el peso óptimo que se debe aplicar. En futuros usos si se conoce el peso concreto a aplicar será mucho más sencillo añadirlo en la opción de pesos.

6.6.4 Implementación de funciones específicas

- **Ruido 2D**

Se aplicará ruido gaussiano de media 0 y varianza controlada por el usuario a los puntos del plano imagen.

- **Redondeo a pixeles enteros**

Aplicación de redondeo del valor del punto proyectado en el plano imagen a un pixel entero.

- **Ruido 3D**

Se aplicará ruido gaussiano de media 0 y desviación estándar controlada por el usuario a los puntos 3D del modelo.

- **RuidoRotacion3D**

Implementación de la función para añadir ruido dependiendo de la posición y orientación del usuario.

- **Error_abs**

Calcula el error absoluto entre la posición real (Ground Truth) y la estimada por Posit.

- **Generar_cilindro**

Función creada para generar un cilindro con base elíptica y altura determinada por el usuario.

- **Imagen2D**

Función que controla la impresión por pantalla de la proyección 2D

- **Num_aleatorio**

Generación de N números aleatorios no repetidos.

- **PesosSigmaPosit**

Aplica pesos mayores a los puntos más cercanos, (puede usarse tanto una distancia como una desviación estándar de ruido). Además también procede a crear varios niveles de pesos no sólo dar el valor máximo y mínimo.

- **Profundidad**

Determina el valor de la componente Z de los puntos proyectados en el modelo cilíndrico.

- **Proyectar_cilindro**

Proyecta los puntos de un modelo determinado sobre la cara lateral del cilindro.

- **Quita_puntos**

Elimina el punto indicado a través de su índice en el frame i, ambas variables son controladas por el usuario.

- **Rot_matrix**

Obtiene la matriz de rotación a partir de los ángulos de Euler dados.

- **Video_maker**

Función que procesa todos los algoritmos necesarios en cada frame.

6.7 Simulaciones

Dentro de las pruebas a realizar vamos a poder separarlas en 4 grandes bloques:

- ✓ Evaluación del error intrínseco del Posit.
- ✓ Reproducción de efectos observados en diferentes métodos de tracking.
- ✓ Evaluación del uso de pesos.
- ✓ Aplicación de diferentes usuarios respecto a un modelo predeterminado.

Todas las pruebas se realizarán sobre el conjunto total de video, por lo que tendremos un total de 12 videos y 300 frames/video para analizar cada prueba.

6.7.1 Bloque 1:

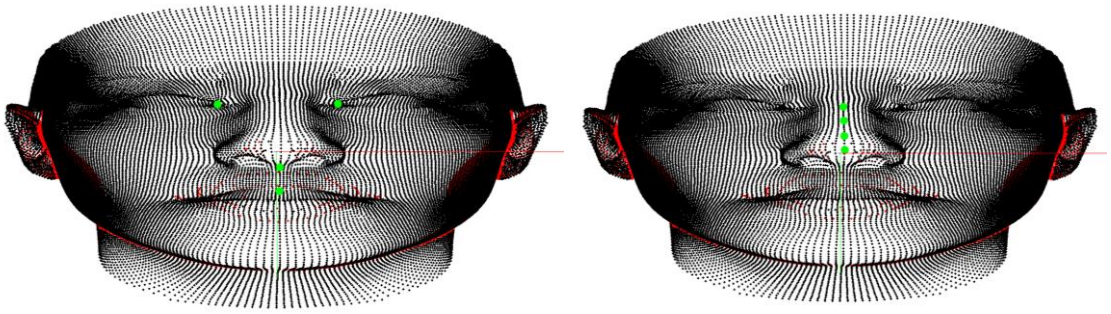
- **Posit aplicado a diferente número de puntos**

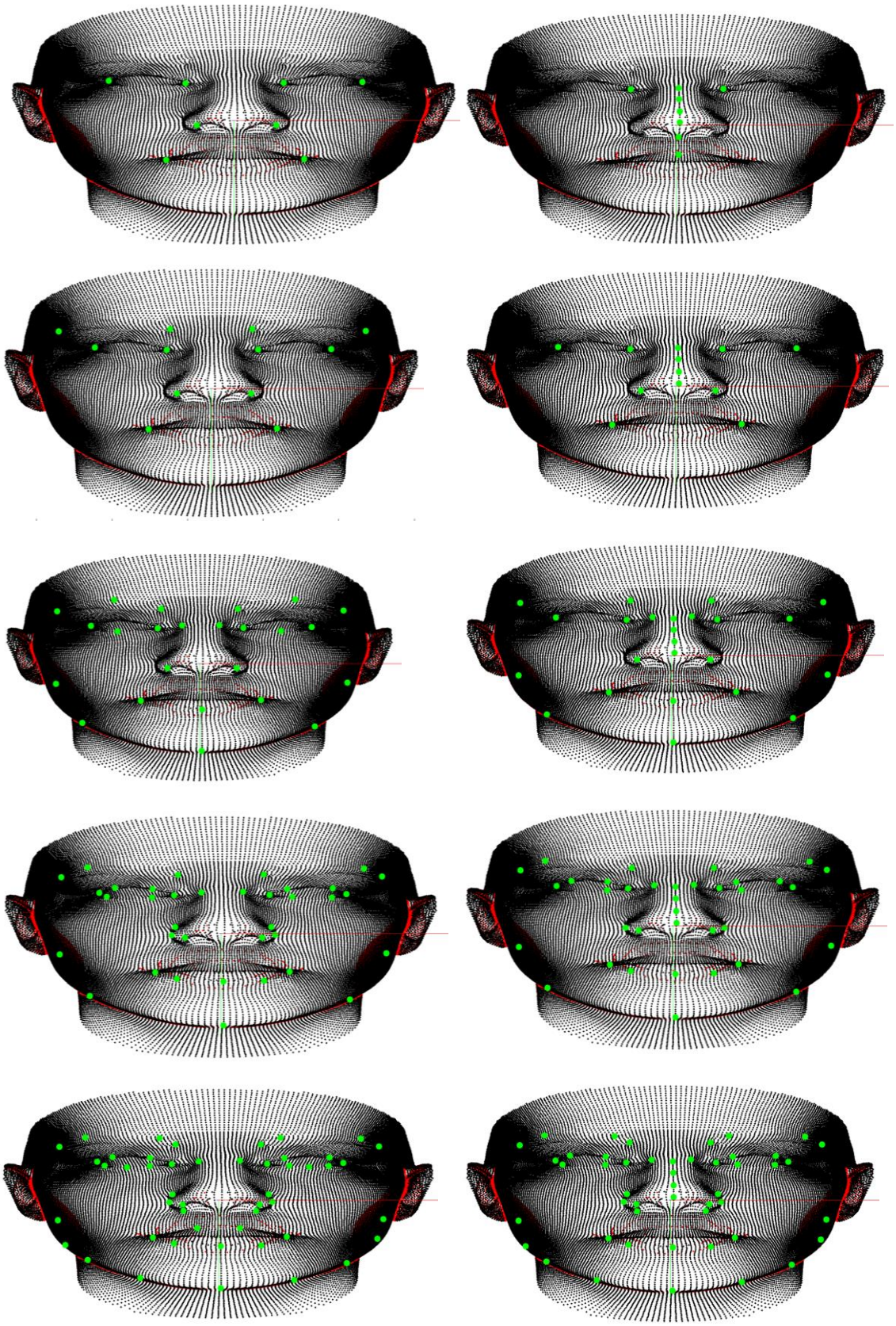
Mediante la realización de esta prueba se espera determinar cómo afecta el aumento del número de puntos al error intrínseco del Posit.

También nos servirá para determinar si la elección de puntos es un aspecto importante y a tener en cuenta si es independiente para el cálculo.

Además de ello, habrá que tener en cuenta la variación del tiempo de computación siendo importante que dicho aumento no imponga una restricción para el procesado en tiempo real.

Posteriormente procederemos a realizar las pruebas con 12 y 54 puntos.





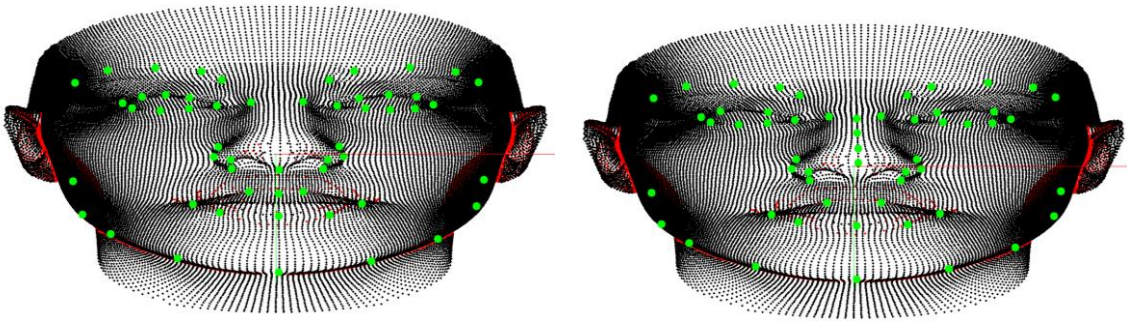


Figura 15. Puntos utilizados en las simulaciones (4,8,12,24,36,48,54).

A la izquierda en la definición de puntos coplanes y a la derecha con la aplicación de Intraface (4 puntos del tabique nasal)

- **Distancia de trabajo y tamaño de la cabeza**

Valoraremos en esta simulación cómo se comporta Posit respecto a dos factores del usuario, la distancia de trabajo y el tamaño de la cabeza.

La distancia de trabajo promedio está en unos 55 cm respecto a la cámara, por ello vamos a estudiar el error entorno a esa posición inicial de usuario.

Para determinar cómo afecta el tamaño de la cabeza valoraremos distintas pruebas realizadas sobre escalados del modelo BFM. Se ha realizado el cálculo en torno a la distancia adecuada de trabajo y utilizando también 3 tipos diferentes de usuario con su respectivo modelo de puntos. Esto lo hemos realizado modificando en un ±10% el modelo BFM que disponemos.

- **Estudio de Softposit**

Se comprobará como de preciso y robusto es el algoritmo *Softposit* y para ello debido a que necesita inicializarse con una translación y rotación se utilizará el *modernPosit* en el primer frame, y a continuación ya seguirá con su estimación anterior. Además para evitar cualquier posible correspondencia se añadirá aleatoriedad a los puntos del modelo.

- **Desalineamiento entre usuario y modelo**

Se utilizará la plataforma de simulación para determinar cómo afecta un desalineamiento entre el usuario y el modelo 3D. Para ello se va a rotar el modelo 3D unos pocos grados en cada grado de libertad.

6.7.2 Bloque 2:

- **Resolución de imagen**

Se determinará la importancia de la resolución de imagen teniendo en cuenta el redondeo producido en cada valor de pixel. Esto nos va a añadir un ruido 2D en el plano de proyección.

Estudiaremos varias resoluciones típicas utilizadas en grabación de video.

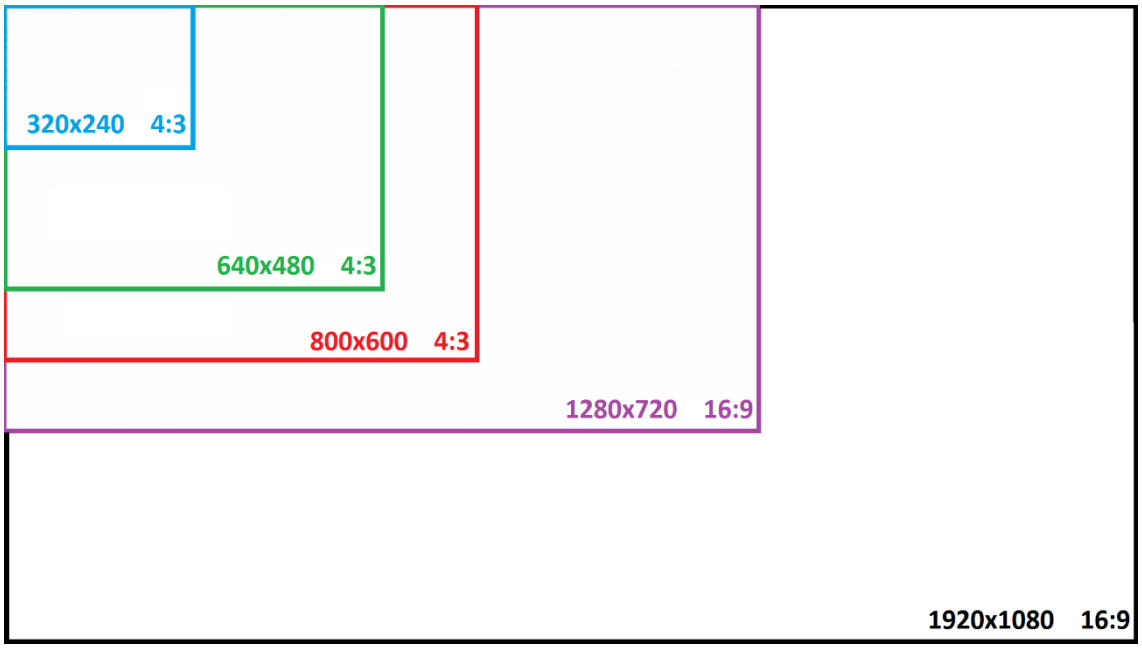


Figura 16. Diferentes resoluciones utilizadas en las simulaciones

- **Ruido aleatorio en el plano imagen**

Analizaremos como afecta el ruido 2D del plano imagen aplicando una distribución de ruido gaussiano con media 0 y diferentes desviaciones estándar de ruido.

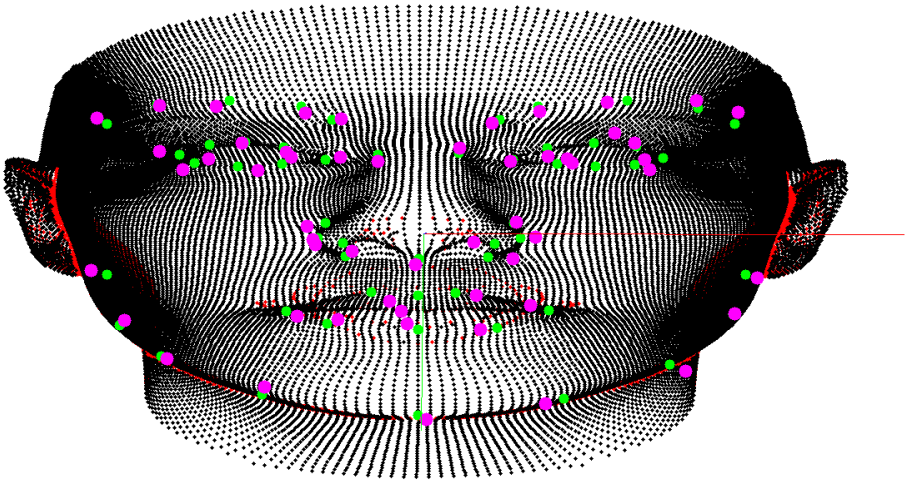


Figura 17. Aplicación de ruido gaussiano con desviación estándar = 5 pixeles En Verde los puntos ideales y en magenta una vez aplicado el ruido.

- **Ruido aleatorio dependiente de la posición y orientación del usuario respecto a la cámara**

Mediante esta prueba se pretende generar un ruido sobre los puntos de la imagen, relacionado con la posición y orientación. Esto es debido a que los puntos frontales a la cámara tienen menos ruido que los puntos más oblicuos. Por ello utilizaremos la función explicada anteriormente de “RuidoRotacion3D”.

Se realizará el estudio también con varios niveles de ruido tratando de determinar los niveles críticos.

- **Estudio de oclusiones**

En esta simulación se compará la respuesta ideal de Posit ante dos posibles formas de *tracking* de puntos: eliminación de puntos ocluidos y aproximación del punto ocluido.

Para ello debemos determinar qué puntos están siendo ocluidos pos la cabeza respecto a la visión de la cámara. Esto lo determinaremos con la función HPR ("Hidden Point Removal") explicada en apartados anteriores.

Posteriormente una vez conocemos los puntos ocluidos en cada frame se eliminarán o se buscará el punto visible más cercano. Para determinar este punto utilizamos la función dsearchn que también ha sido detallada previamente.

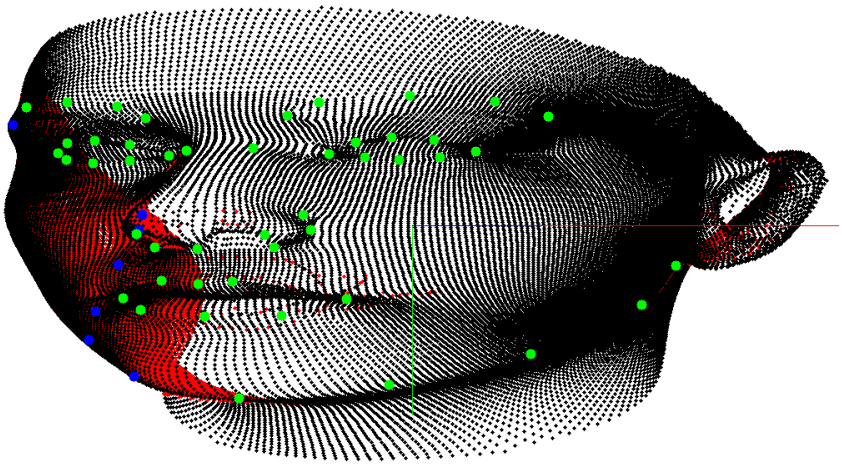


Figura 18. Ejemplo de oclusión en el modelo. En verde puntos visibles y en azul puntos ocluidos

- **Combinación de ruido dependiente de la posición y orientación y oclusiones**

Combinaremos los dos apartados anteriores y así iremos obteniendo resultados más realistas.

6.7.3 Bloque 3:

- **Aplicación de pesos a Posit**

Se realizará un estudio de cómo afecta la aplicación de pesos al Posit. Para ello se repetirán los puntos más próximos a su posición real un mayor número de veces, mientras que los más alejados tendrán un peso menor y por tanto menor número de repeticiones. Se observará la magnitud de mejora en la estimación. Tendremos en cuenta también el tiempo de computación como hemos hecho en pruebas anteriores para que no exceda de lo requerido.

6.7.4 Bloque 4

- **Ruido aleatorio en el modelo 3D**

Realizaremos simulaciones con diferente nivel de aleatoriedad del modelo 3D para comprobar el margen de error que tendremos si consideramos pequeñas diferencias entre usuario y modelo.

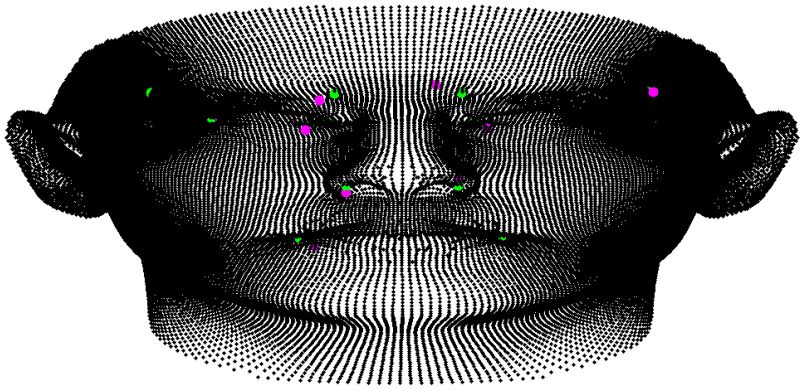


Figura 19. Aplicación de ruido 3D en BFM. Puntos magenta con ruido. En verde los puntos ideales.

- **Combinación de diferentes modelos**

Se comparará la posible relación entre los modelos de Dasha y BFM y posteriormente determinar la precisión del modelo cilíndrico. Esto lo comprobaremos realizando unas simulaciones sin ruido y viendo si podemos determinar una linealidad o constante de diferencia.

USUARIO	MODELO
Dasha	BFM
BFM	Dasha
Modelo cilíndrico	BFM
BFM	Modelo cilíndrico

- **Relación entre usuario y modelo**

En esta simulación se busca recrear un modelo realista de diferentes usuarios referidos a un mismo modelo, para ello se realizarán variaciones del BFM para obtener la proyección en el plano imagen y ese será referido a diferentes escalados del BFM.

USUARIO	MODELO
Cabeza un 10% más pequeña	Estándar
Cabeza estándar	Estándar
Cabeza un 10% más grande	Estándar

- **Diferente usuario respecto al mismo modelo**

Se han extraído diferentes posibles usuarios de la base de datos del modelo BFM y lo que se pretende en esta prueba es comparar la estimación que puede proporcionar POSIT de estos diferentes usuarios respecto al modelo genérico promediado.

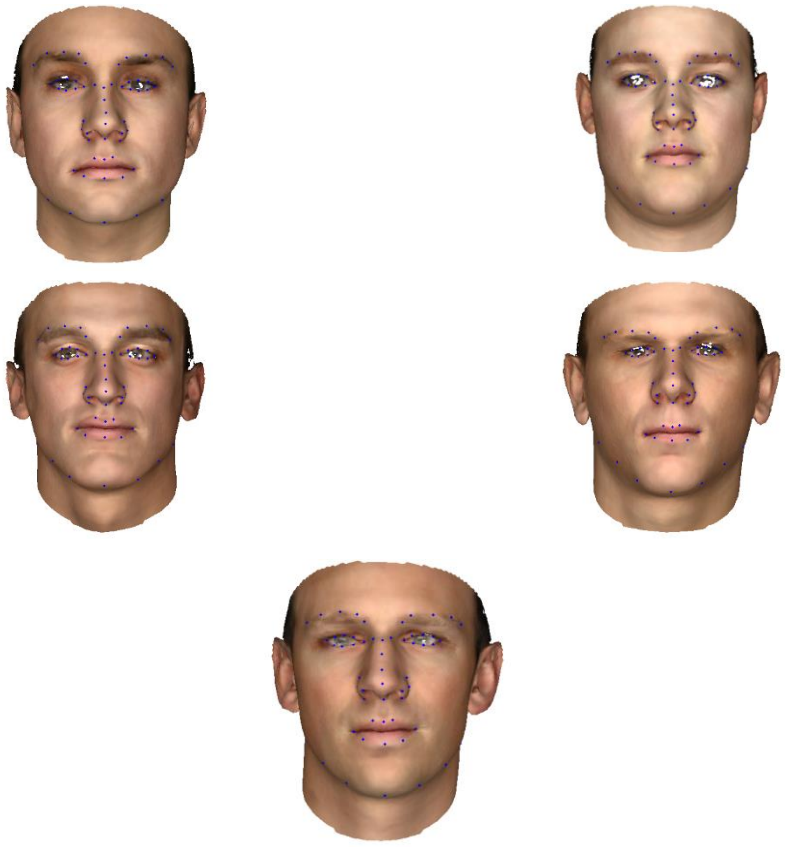


Figura 20. Ejemplo de usuarios diferentes extraídos de la base de datos de BFM

7 Análisis de resultados

En este apartado se explicarán los resultados de las pruebas realizadas.

7.1 Bloque 1:

7.1.1 Posit aplicado a diferente número de puntos

Como podemos apreciar, el error disminuye según aumentamos el número de puntos, si bien cabe resaltar dos asuntos.

El primero de ellos es que aplicando los algoritmos de Posit obtenemos mayores errores utilizando 12 puntos en vez de 8 o 16, por lo que se hará un pequeño análisis más concreto de este conjunto de puntos.

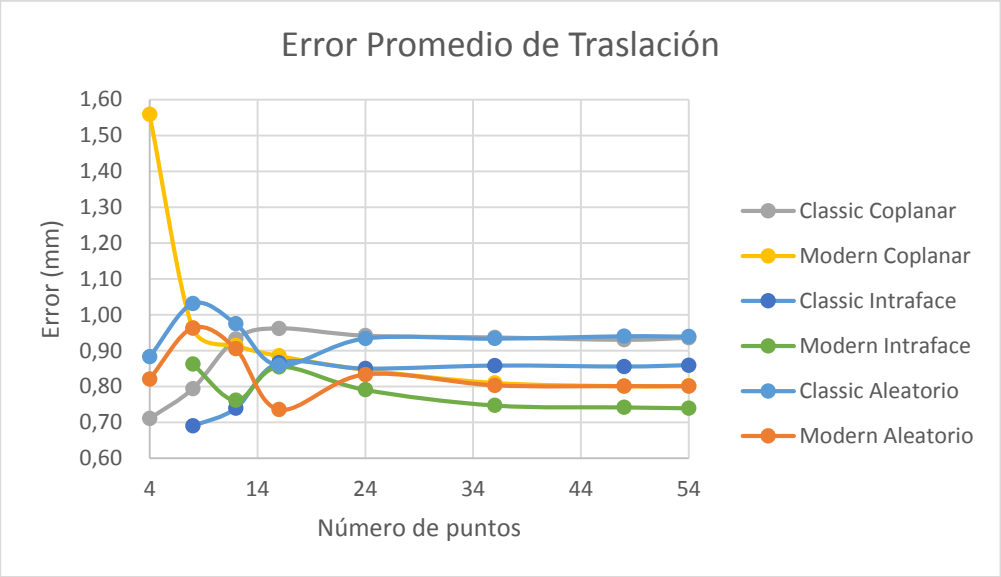
El segundo es detallar cómo la mejora en precisión a partir de la utilización de 24 puntos del modelo no es muy elevada respecto a usar un número mayor de puntos. Aunque el incremento en el tiempo de computación es inapreciable (continúa en el rango de las décimas de milisegundo).

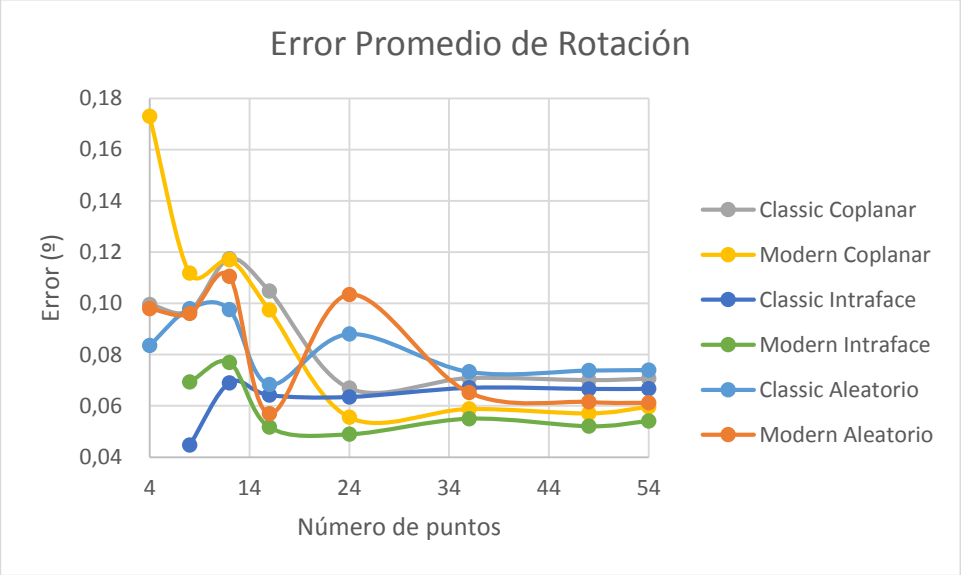
Además para hacer más robusta esta prueba hemos seleccionado los puntos de 3 formas diferentes, una de ellas donde los puntos están cercanos a la coplanaridad, otra donde se incluyen 4 puntos pertenecientes a la nariz para romper la coplanaridad (Intraface) y una tercera forma donde los puntos son seleccionados aleatoriamente del conjunto de 58 puntos explicado anteriormente.

Esto determina que los mejores resultados se obtienen introduciendo en el cálculo puntos no coplanares (técnica utiliza en Intraface).

También podemos apreciar como la elección de los puntos es un tema a tener en cuenta ya que la estimación con puntos aleatorios tiene valores dispares por ejemplo cuando se han usado 24 puntos en la rotación.

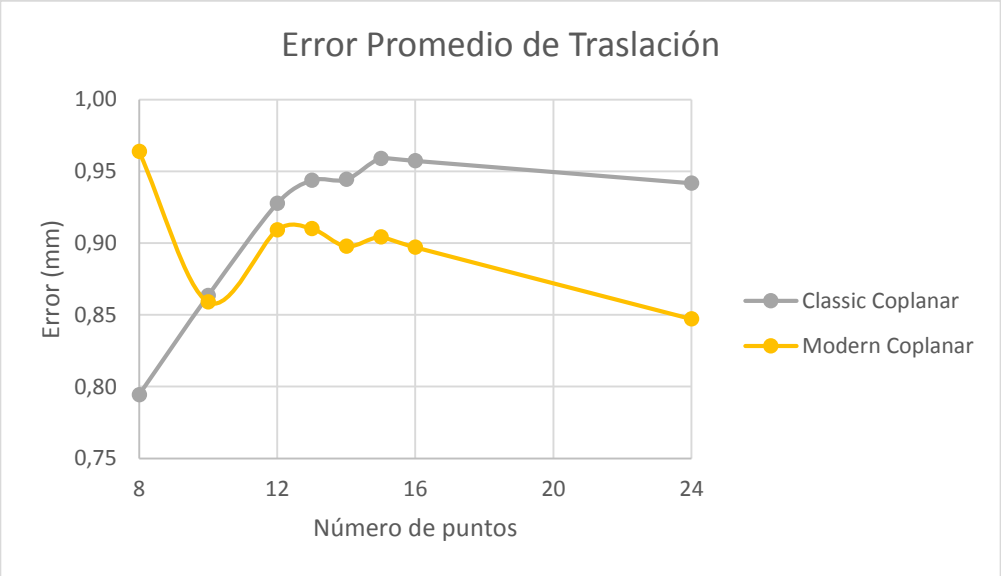
Por último afirmar que *modernPosit* ofrece menores errores que *classicPosit* tanto en rotación como en traslación en término medio.

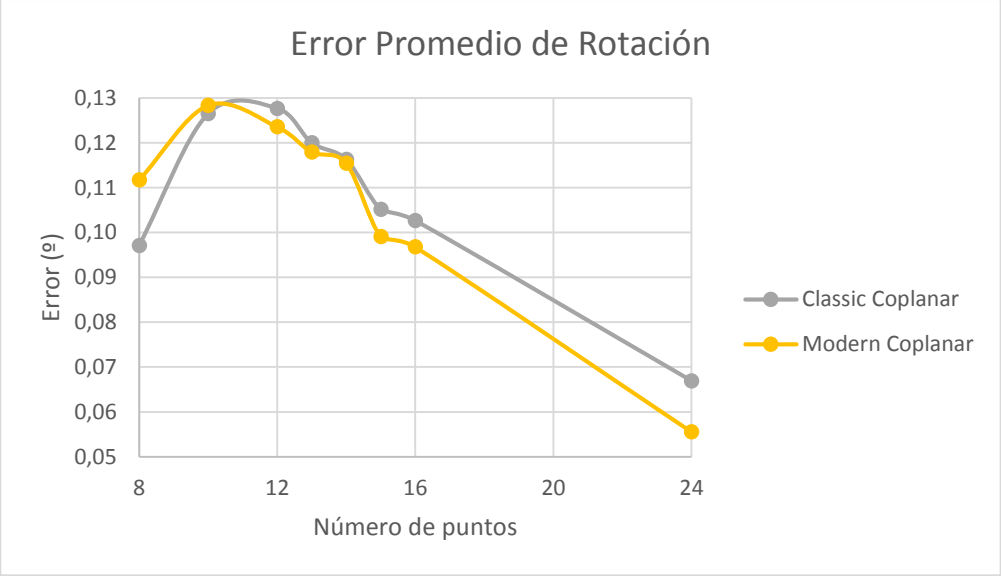

Error Promedio de Traslación
Error (mm)
1,60
1,50
1,40
1,30
1,20
1,10
1,00
0,90
0,80
0,70
0,60
4
14
24
34
44
54
Número de puntos
Classic Coplanar
Modern Coplanar
Classic Intraface
Modern Intraface
Classic Aleatorio
Modern Aleatorio


Error Promedio de Rotación
Error (º)
0,18
0,16
0,14
0,12
0,10
0,08
0,06
0,04
4
14
24
34
44
54
Número de puntos
Classic Coplanar
Modern Coplanar
Classic Intraface
Modern Intraface
Classic Aleatorio
Modern Aleatorio

Intentamos analizar más a conciencia el pico producido en torno a la marca de 12 puntos. Y como vemos debajo de estas líneas nos encontramos con que hasta la utilización de 16 puntos no se reduce el error estimado y vuelve a los valores obtenidos con 8 puntos. Por lo que dados estos resultados deberíamos pensar en utilizar 8 o 16 puntos y ningún valor entre medio de esos límites.

A futuro sería correcto estudiar porque al introducir los puntos de las cejas se produce este aumento del error que posteriormente es corregido.



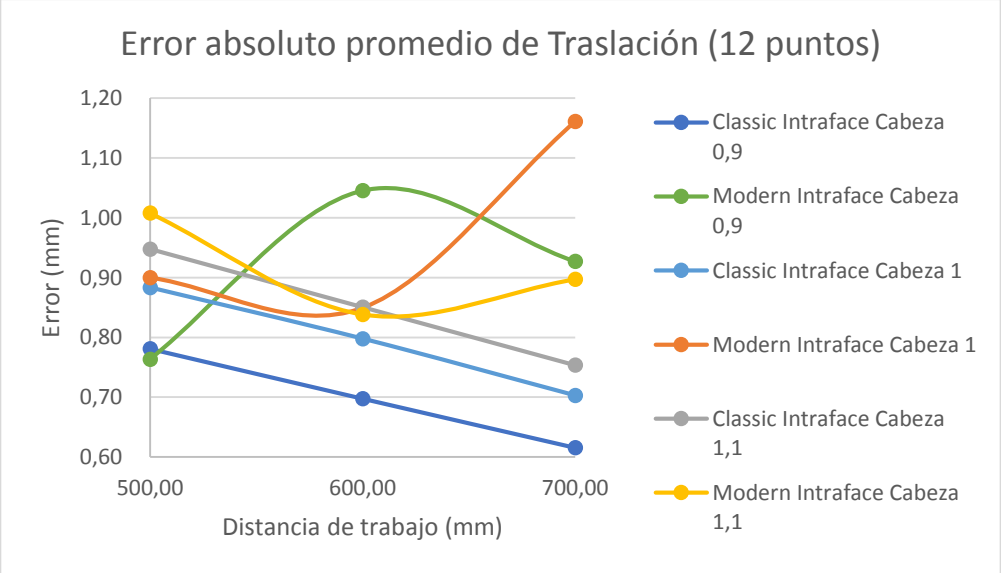


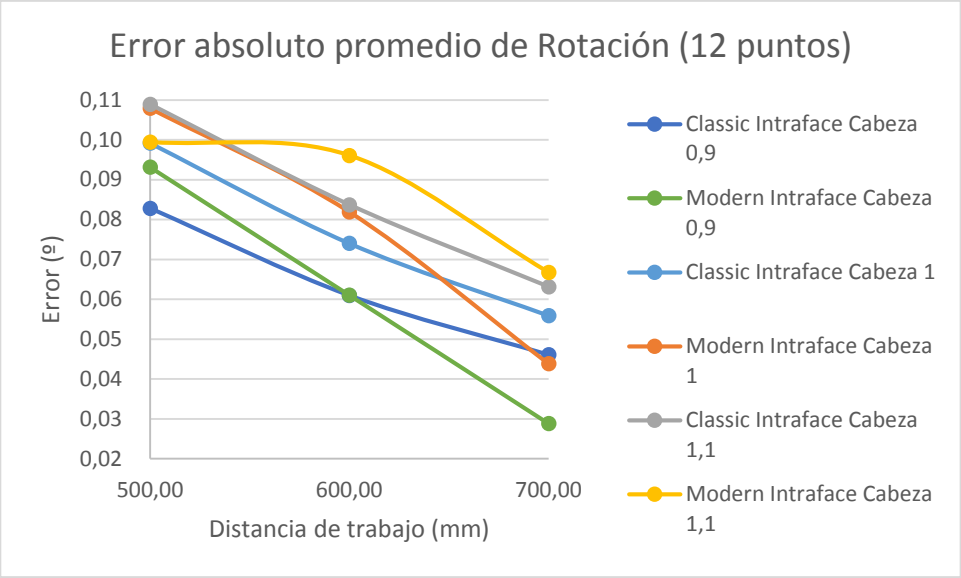
7.1.2 Distancia de trabajo y tamaño de la cabeza del usuario

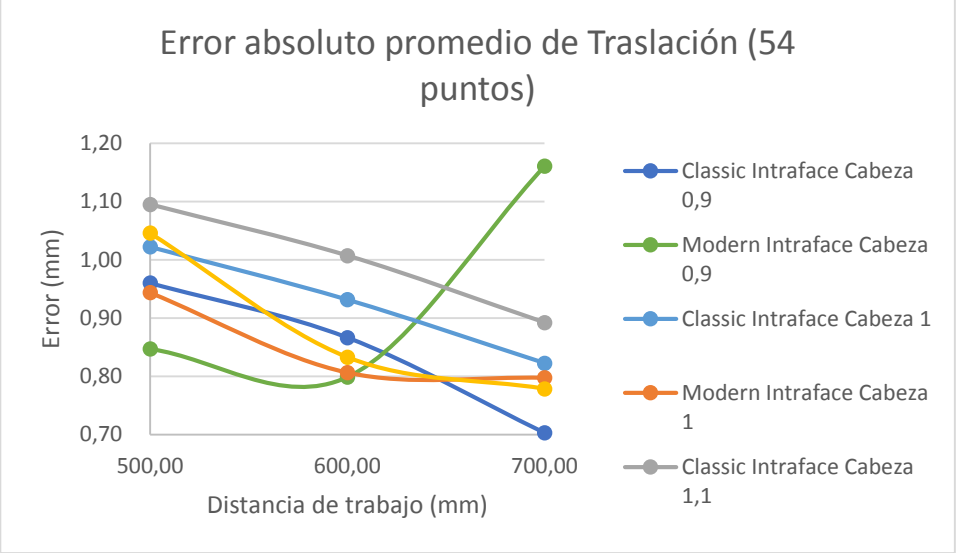
Se puede determinar que el error de estimación disminuye importantemente si la distancia de trabajo es mayor. Estamos hablando de mejoras entorno al 200% por lo que en la medida de lo posible para obtener los mejores resultados con Posit intentaremos evitar posicionarnos muy próximos a la cámara.

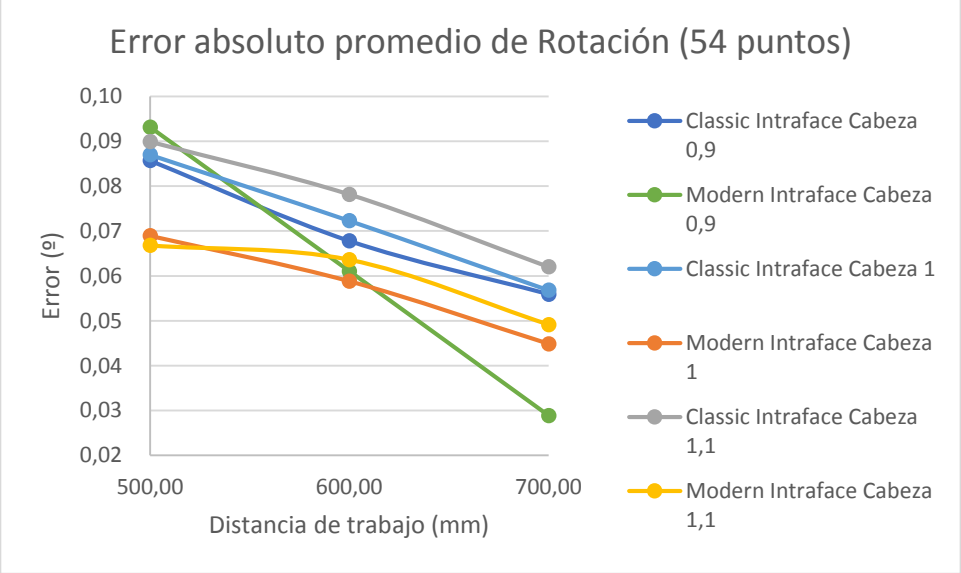
En cambio el tamaño de la cabeza del usuario tiene una importancia menor en este estudio ya que si bien es cierto que con cabezas de tamaños mayores el error es menor no se llega a los rangos de mejora obtenidos al variar la posición.

Respecto a la aplicación sobre 12 puntos o 54, vemos cómo en la utilización de 54 puntos tenemos mejores estimaciones de la rotación que cuando se usan 12. En traslación promedio no se aprecia este error pero realmente esto es debido a la distancia en el eje Z (lejanía respecto a la cámara) donde realmente los errores son mayores.




Error absoluto promedio de Rotación (12 puntos)
0,11
0,10
0,09
0,08
0,07
0,06
0,05
0,04
0,03
0,02
Error (º)
500,00
600,00
700,00
Distancia de trabajo (mm)
Classic Intraface Cabeza 0,9
Modern Intraface Cabeza 0,9
Classic Intraface Cabeza 1
Modern Intraface Cabeza 1
Classic Intraface Cabeza 1,1
Modern Intraface Cabeza 1,1


Error absoluto promedio de Traslación (54 puntos)
1,20
1,10
1,00
0,90
0,80
0,70
Error (mm)
500,00
600,00
700,00
Distancia de trabajo (mm)
Classic Intraface Cabeza 0,9
Modern Intraface Cabeza 0,9
Classic Intraface Cabeza 1
Modern Intraface Cabeza 1
Classic Intraface Cabeza 1,1


Error absoluto promedio de Rotación (54 puntos)
0,10
0,09
0,08
0,07
0,06
0,05
0,04
0,03
0,02
Error (º)
500,00
600,00
700,00
Distancia de trabajo (mm)
Classic Intraface Cabeza 0,9
Modern Intraface Cabeza 0,9
Classic Intraface Cabeza 1
Modern Intraface Cabeza 1
Classic Intraface Cabeza 1,1
Modern Intraface Cabeza 1,1

7.1.3 Desalineamiento entre usuario y modelo

Se ha probado a desalinear el modelo 3D en los diferentes grados de libertad que tiene de manera individual y también en conjunto. Se ha podido observar que si bien los errores de estimación van a incluir el desalineamiento asignado, el añadido a este no supera los 0.6º en promedio. Pero si se aprecia que afecta de forma no lineal a la estimación y por lo tanto será complicado de eliminar este desajuste en una simulación real.

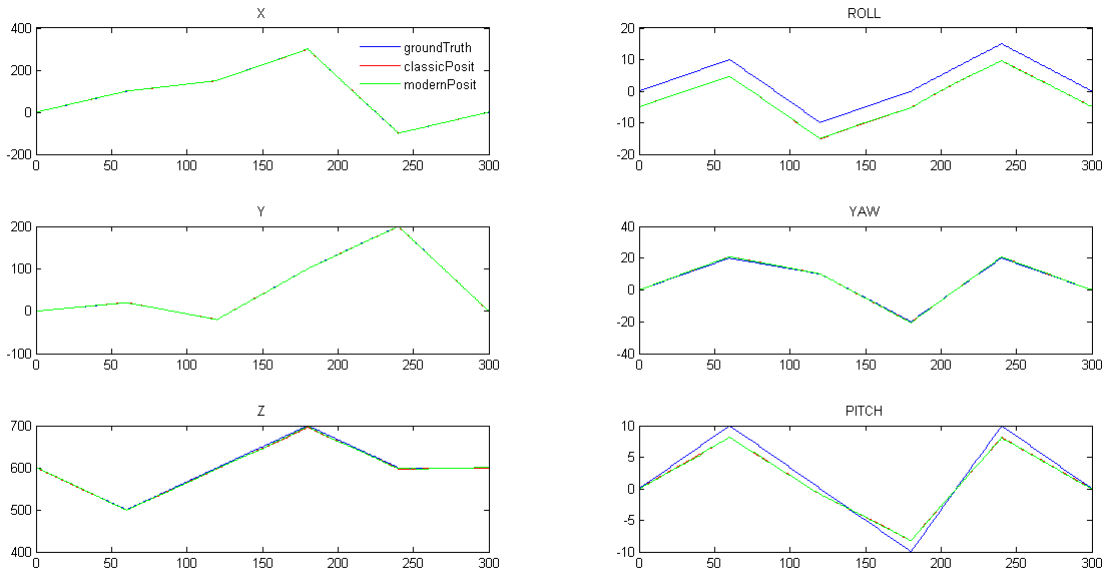


Figura 21. Desealineamiento del modelo respecto al usuario de 5º en ROLL

7.1.4 Estudio de Softposit

Las pruebas realizadas con SoftPosit quedan resumidas en la siguiente tabla:

Número de puntos	12	54
Tiempo promedio	2.18s	5.6s
Porcentaje de éxito	82%	83.3%
Error promedio	X: 788,53 Y: 552,31 Z: 3289,40 ROLL= 30,53 YAW= 11,23 PITCH= 53,91	X: 0,29 Y: 0,16 Z: 4,49 ROLL= 0,01 YAW= 0,12 PITCH= 0,02

En ambas pruebas el modelo global de puntos ha sido el que engloba los 58 puntos. Lo que podemos extraer de los resultados es que Softposit tiene un porcentaje de éxito en la búsqueda de la *pose* superior al 80% aunque realmente lo interesante es si tras encontrar la *pose* esta es la correcta y se puede estimar el error correctamente. Así se puede ver como en el uso de 12 puntos los errores son gigantescos y podemos arriesgarnos a asegurar que SoftPosit no encuentra su par correctamente. Por tanto el porcentaje de éxito al usar 12 puntos no puede decirse que sea del 82%.

En cambio, utilizando 54 puntos obtenemos unos resultados muy buenos y esperanzadores, en el mismo rango de error que el dado por ModernPosit. Eso sí con un porcentaje de éxito del 83.3% en vez del 100% obtenido con ModernPosit.

El otro hándicap observado se halla en el tiempo de computación que si bien vemos que depende del número de puntos en ninguno de los casos posibilitaría la aplicación en tiempo real.

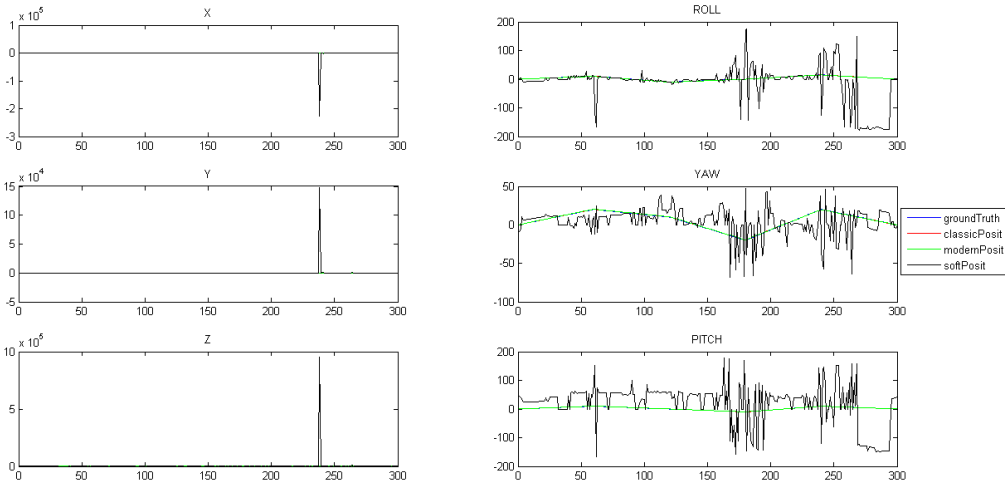


Figura 22. SoftPosit aplicado sobre 12 puntos

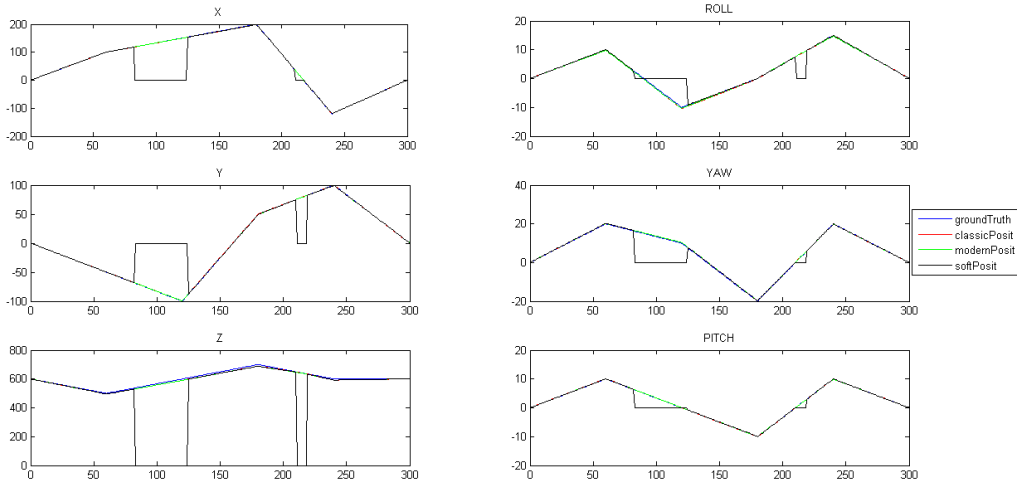


Figura 23 SoftPosit aplicado sobre 54puntos

En las gráficas podemos apreciar claramente que en el caso de 12 puntos no se consigue ningún valor correcto, mientras que en gráfica que ilustra el movimiento con 54 puntos a seguir vemos cómo va correctamente siguiendo la *pose* real.

Indicar además que los escalones que se ven realmente son los frames donde no se consigue calcular la posición y orientación, pero considero que si no se pintan (que es lo correcto) pasaría por alto el efecto y de esta forma se ven los momentos donde no es capaz de encontrar la pose.

7.2 Bloque 2:

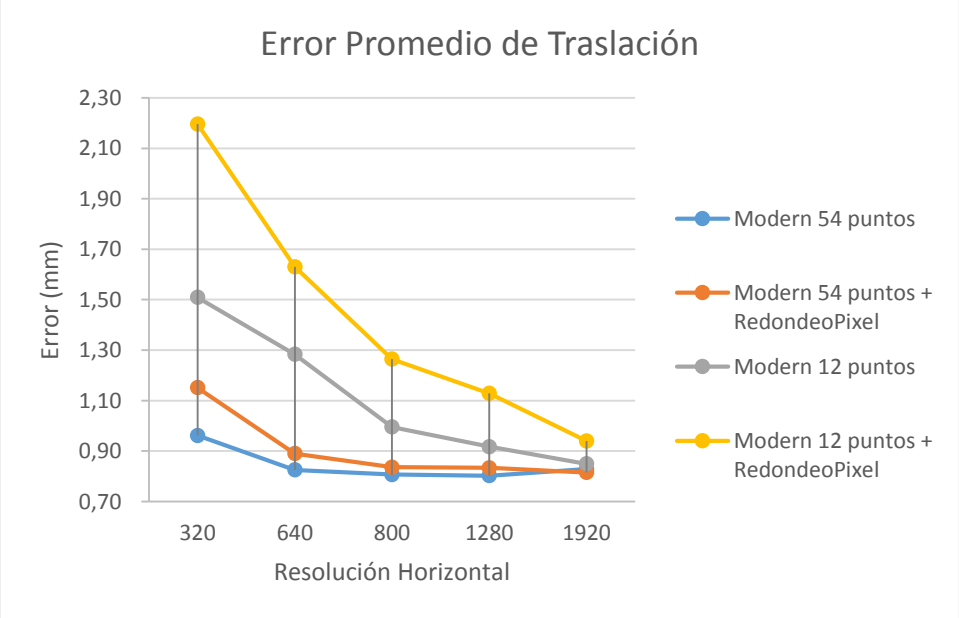
7.2.1 Resolución de imagen

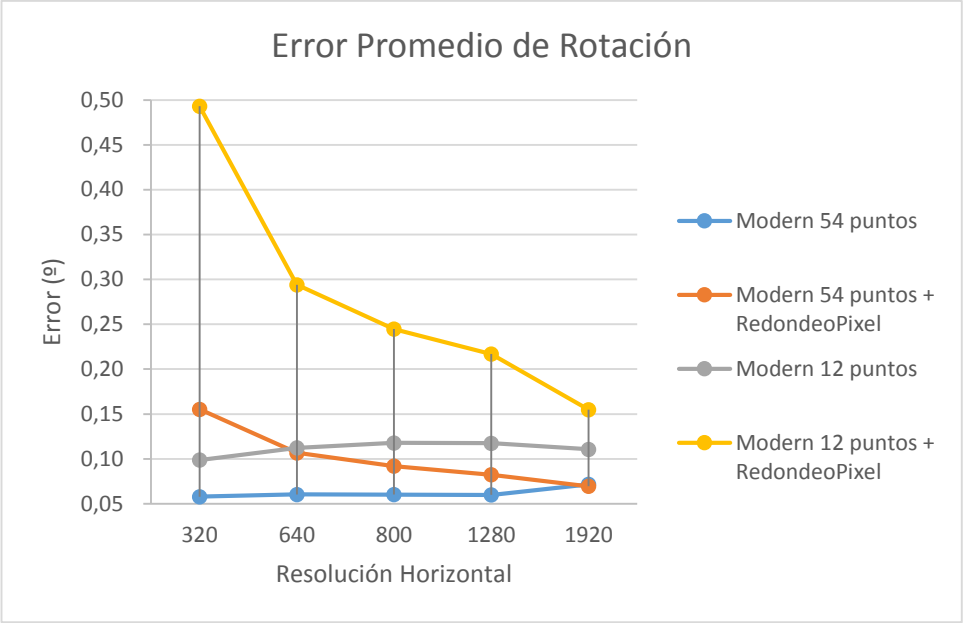
Se han estudiado resoluciones estándar que se usan actualmente y, dentro de ello y debido a que algunos métodos de tracking utilizan pixeles enteros para determinar el punto marcado, se ha procedido a realizar la prueba con medidas en pixeles decimales y enteros.

Se aprecia que cuando utilizamos pixeles decimales sin redondeo la estimación es prácticamente idéntica. Sin embargo cuando se procede a redondear a pixel entero se aprecia perfectamente que la estimación mejora a la vez que aumentamos la resolución.

Un punto clave se obtiene en el cruce entre la estimación sin redondeo para 12 puntos y con redondeo en los 54. Donde apreciamos en la gráfica de rotación que a partir de una resolución de 800x600 obtenemos mejores resultados redondeando los pixeles donde se proyectan los 54 puntos que en la proyección de los 12 puntos sin redondear.

Por tanto, lo más adecuado es utilizar la mayor resolución permitida por la cámara, y en su defecto una resolución superior a 800x600 pixeles, donde se ha conseguido reducir el error de estimación a la mitad respecto a la resolución mínima estudiada de 320x240.

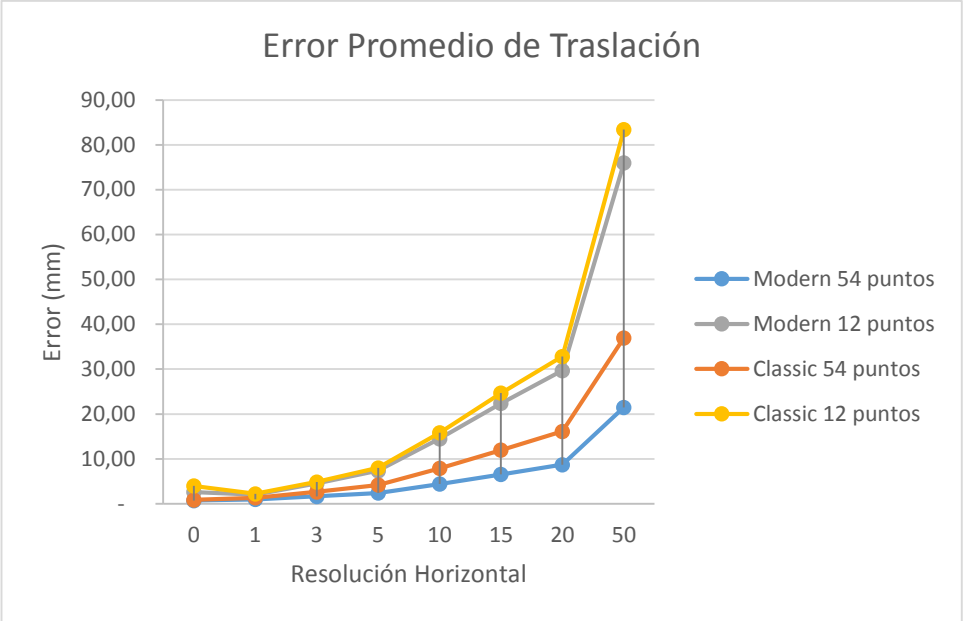


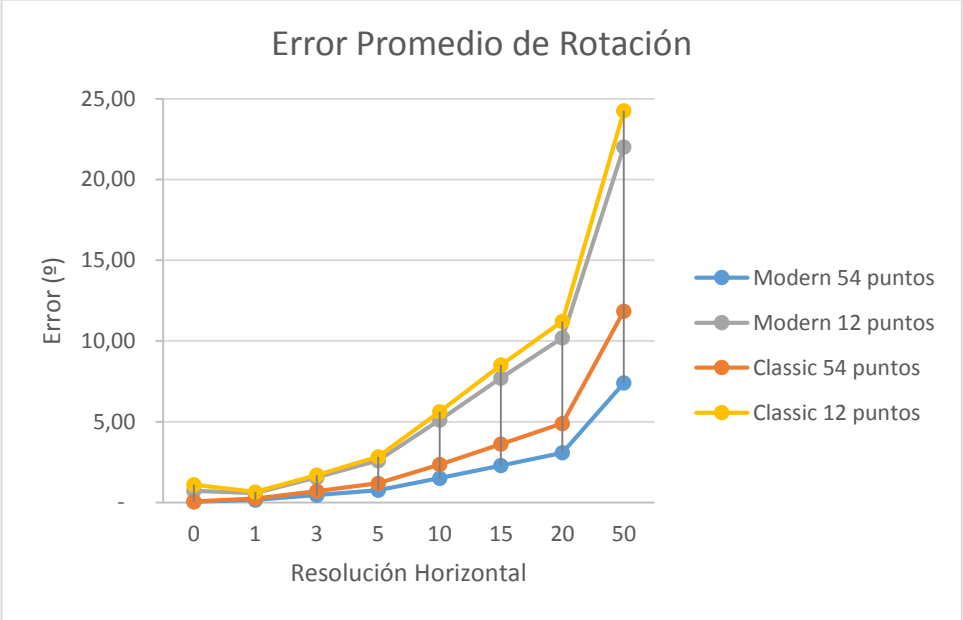


7.2.2 Ruido aleatorio en el plano imagen

Como se esperaba, la aplicación de ruido empeora los resultados. En nuestras pruebas el ruido esta medido en pixeles y se observa una clara dependencia lineal: a mayor nivel de ruido (desviación estándar) mayor error en la estimación. No obstante, nos permite determinar que hasta utilizando una desviación estándar de 5 pixeles estamos en rangos de estimaciones realistas para métodos actuales de estimación de la posición de la cabeza mediante procesado de imágenes 2D.

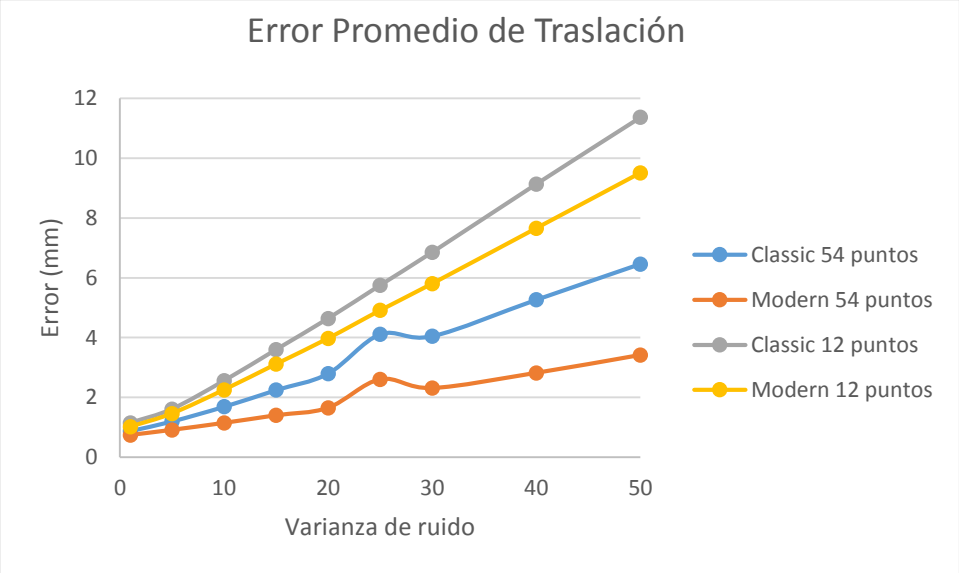
Analizando si es más correcto utilizar 12 o 54 puntos, vemos como siguiendo 54 puntos obtenemos un sistema mucho más robusto. Estos datos son los esperados.

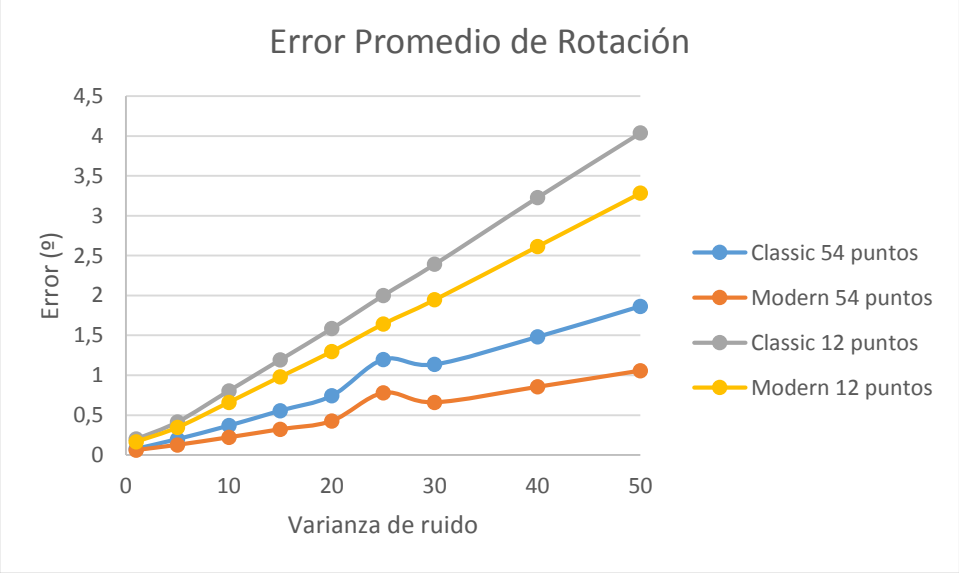

Error Promedio de Traslación
90,00
80,00
70,00
60,00
50,00
40,00
30,00
20,00
10,00
-
Error (mm)
0
1
3
5
10
15
20
50
Resolución Horizontal
Modern 54 puntos
Modern 12 puntos
Classic 54 puntos
Classic 12 puntos


Error Promedio de Rotación
25,00
20,00
15,00
10,00
5,00
-
Error (º)
0
1
3
5
10
15
20
50
Resolución Horizontal
Modern 54 puntos
Modern 12 puntos
Classic 54 puntos
Classic 12 puntos

7.2.3 Ruido aleatorio dependiente de la posición y orientación del usuario respecto a la cámara

Podemos apreciar como los resultados entre la aplicación sobre 12 o 54 puntos se han aproximado entre sí. Esto es debido a que en la elección de 12 puntos se han tomado puntos frontales de la cara, por lo que la aplicación de este ruido no va a distorsionar en un grado elevado los puntos ideales. Sin embargo eligiendo 54 puntos varios de ellos van a estar oblicuos respecto a la cámara, por tanto con ruido más elevado y se obtiene un error de estimación mayor. De todas formas, éste no consigue ser tan elevado como para superar el error propio obtenido al simular con 12 puntos.





-

7.2.4 Estudio de oclusiones

Los resultados obtenidos tras la realización de esta prueba nos van a permitir afirmar que en caso de oclusión de puntos es preferible la supresión de dichos puntos a la aproximación visible de ellos.

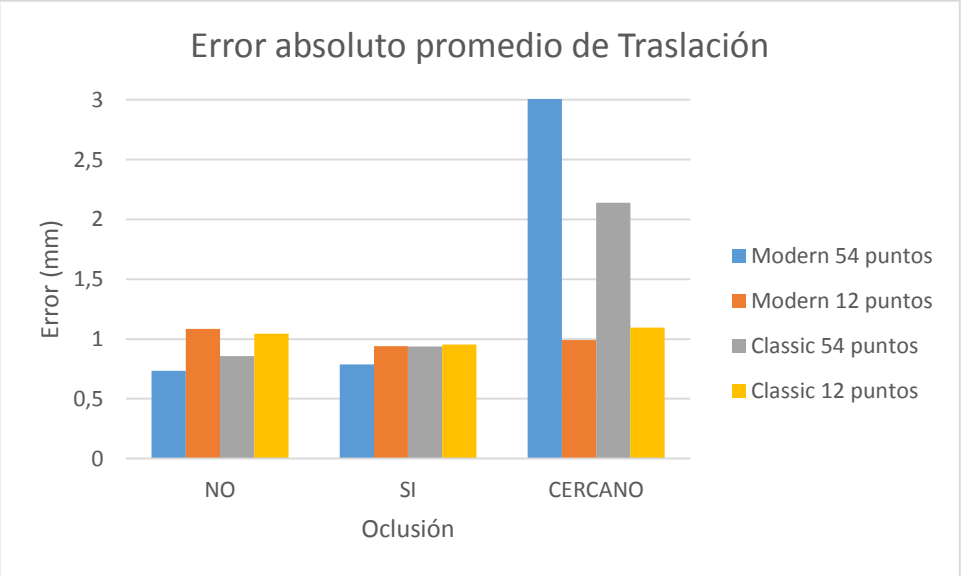
Podría destacarse que en la simulación aproximando los puntos al más cercano aplicado sobre los 54 puntos, se observan unos errores de estimación incluso del 200% respecto a si se ocluyen los puntos. Esto es debido a que los puntos no precisos (como es el caso) afectan muy negativamente a POSIT y al incluir los puntos de la mandíbula que están muy fácilmente ocluidos se empeoran los resultados.

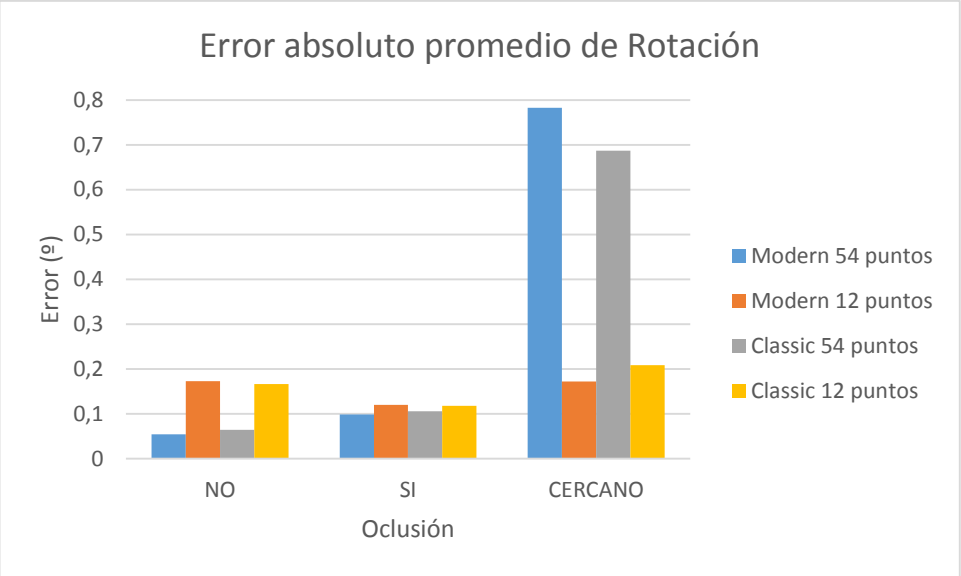
En las gráficas el rótulo del eje X se corresponde con:

NO: No hay oclusión de puntos

SI: Si se ocluyen los puntos

Cercano: se aproximan los puntos ocluidos al más cercano de los visibles



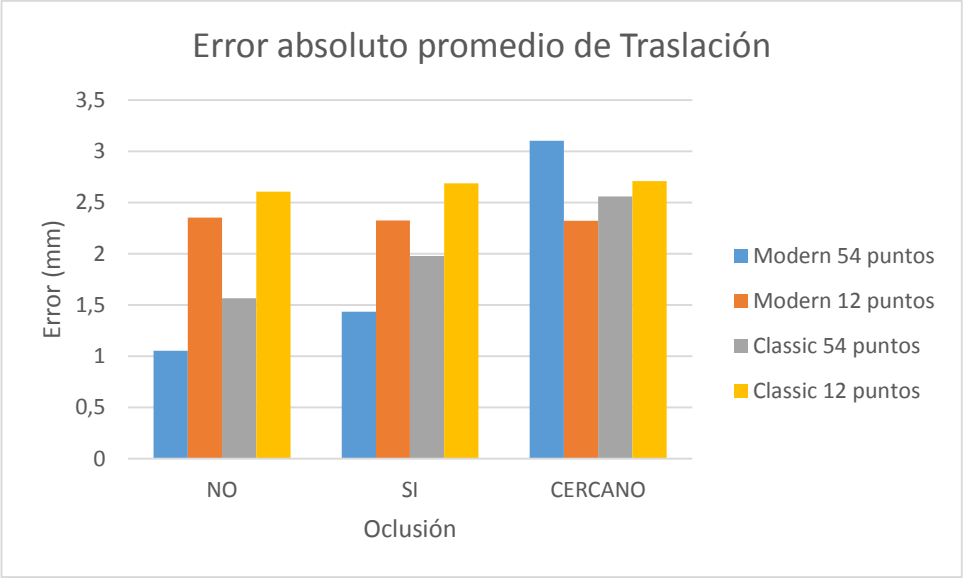


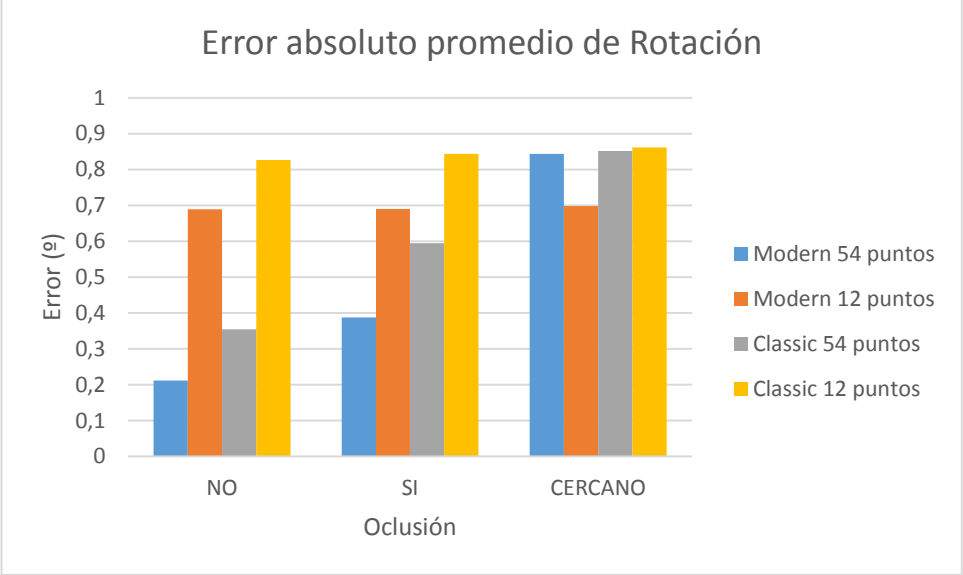
7.2.5 Combinación de ruido dependiente de la posición y orientación y oclusiones

Será de utilidad si realizando un tracking se puede determinar que un punto pueda no ser correcto, en dichos casos mejorará la estimación de su posición y orientación si ese punto es eliminado para el cálculo de Posit. Se muestran los resultados de aplicar una desviación estándar máxima de 10 pixeles.

Se aprecia respecto a la prueba anterior que al añadir ruido se aproximan los errores estimados de las tres opciones analizadas, esto es debido a que el aproximar a un punto cercano es añadir un ruido 2D, y como las otras opciones también incluyen ruido sus estimaciones empeoran.

Aun así, el mejor se obtendrá sin ocluir ninguno de los 54 puntos y estimando con *modernPosit,* y siendo más realista ocluyendo dichos puntos.


Error absoluto promedio de Traslación
3,5
3
2,5
2
1,5
1
0,5
0
Error (mm)
NO
SI
CERCANO
Oclusión
Modern 54 puntos
Modern 12 puntos
Classic 54 puntos
Classic 12 puntos

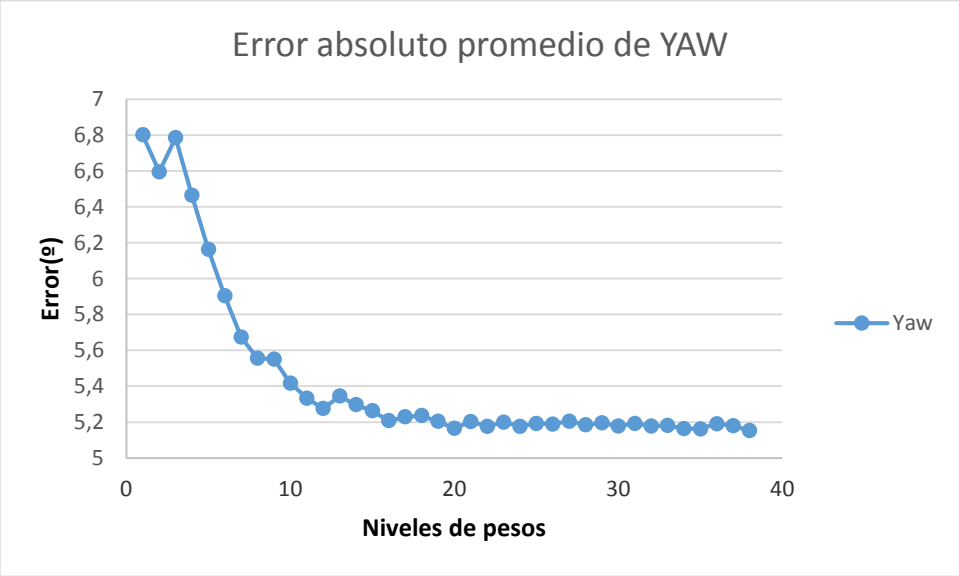

Error absoluto promedio de Rotación
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
Error (º)
NO
SI
CERCANO
Oclusión
Modern 54 puntos
Modern 12 puntos
Classic 54 puntos
Classic 12 puntos

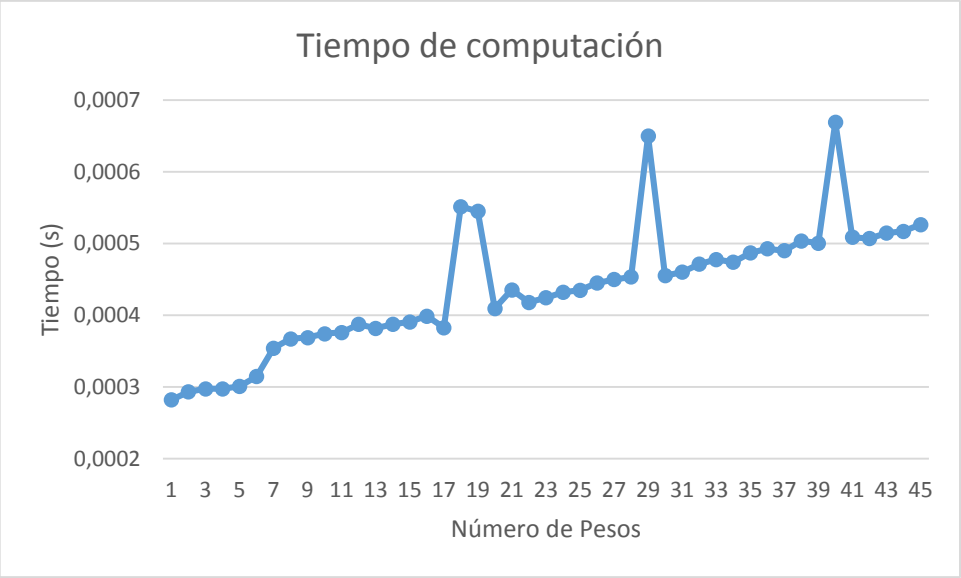
7.3 Bloque 3:

7.3.1 Aplicación de pesos a Posit

Se ha podido apreciar como la repetición de los puntos más correctos de la proyección mejora considerablemente los resultados obtenidos. Es muy importante determinar que el tiempo de computación no es un problema en este estudio ya que aunque aumenta no es crítico, mientras que los resultados se estabilizan al aplicar pesos guardando una relación 1:10 aproximadamente. Debido a que tiempo de computación no aumenta ni siquiera un orden de magnitud podemos pensar en utilizar una relación 1:20 y evitar posibles errores no tenidos en cuenta en esta simulación.

El estudio se ha realizado sobre las simulaciones a las cuales se les ha aplicado ruido o un error diferente en cada punto y podemos resumirlo con una gráfica representativa. Se ha tomado uno de los resultados con error muy elevado en la estimación de la rotación ya que en su 1º calculo teníamos 6.8º de error en Yaw y aplicando pesos obtenemos con 20 niveles de pesos un error de 5.2º un 24% menos de error mientras que el tiempo de computación no aumenta ni 1 milisegundo.

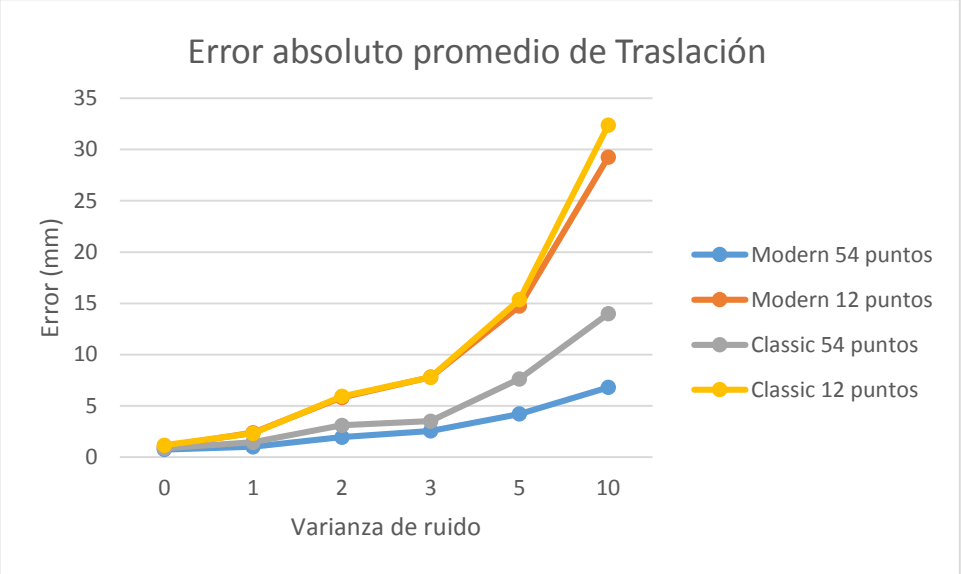


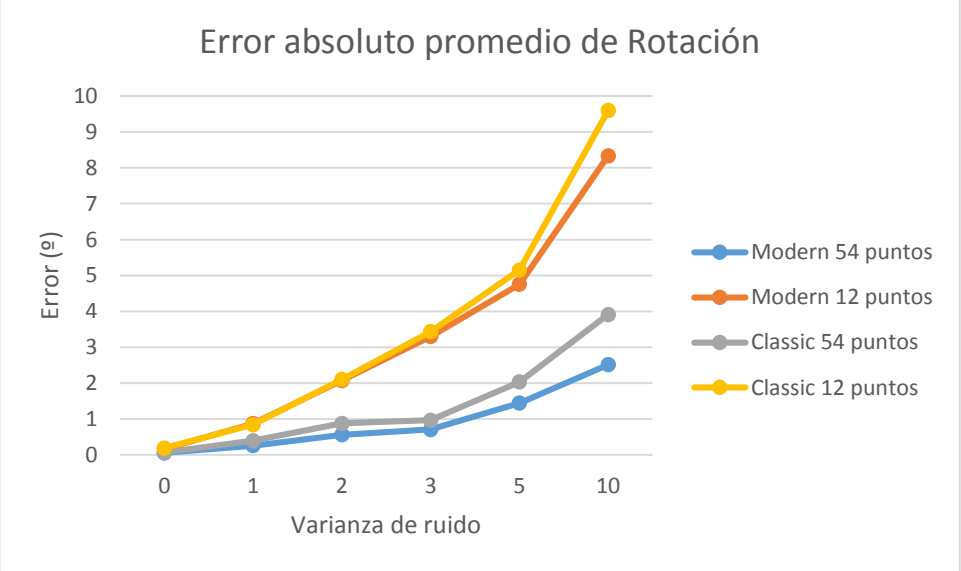

Tiempo de computación
0,0007
0,0006
0,0005
0,0004
0,0003
0,0002
Tiempo (s)
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45
Número de Pesos

7.4 Bloque 4

7.4.1 Ruido aleatorio en el modelo 3D

En esta prueba podemos determinar que utilizando 12 puntos el error en la estimación de error se vuelve inadmisible con diferencias muy pequeñas (desviación estándar 3 pixeles) entre el modelo 3D y el usuario, mientras que usando 54 puntos se puede tener un error admisible hasta variaciones distribución normal de ruido con desviación estándar de 10 mm.



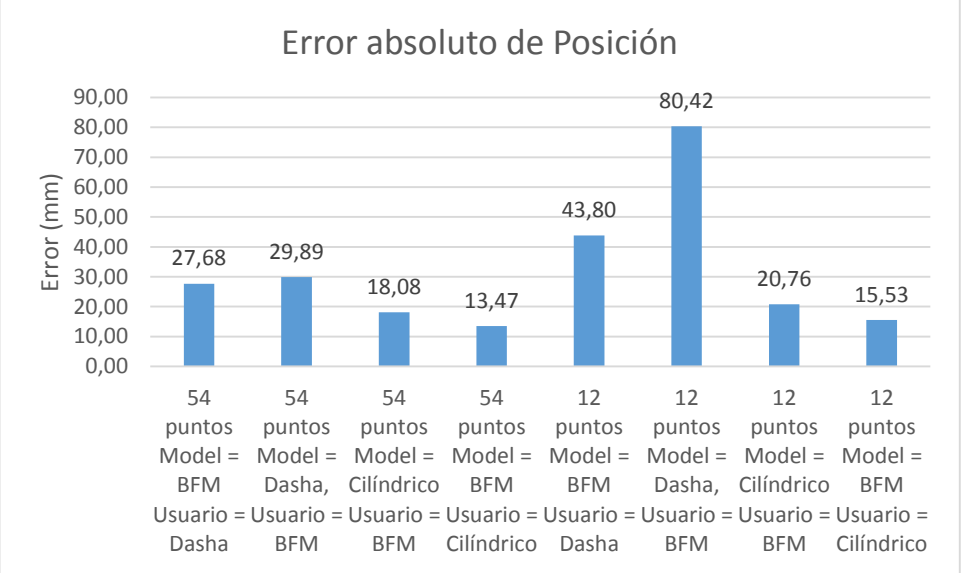


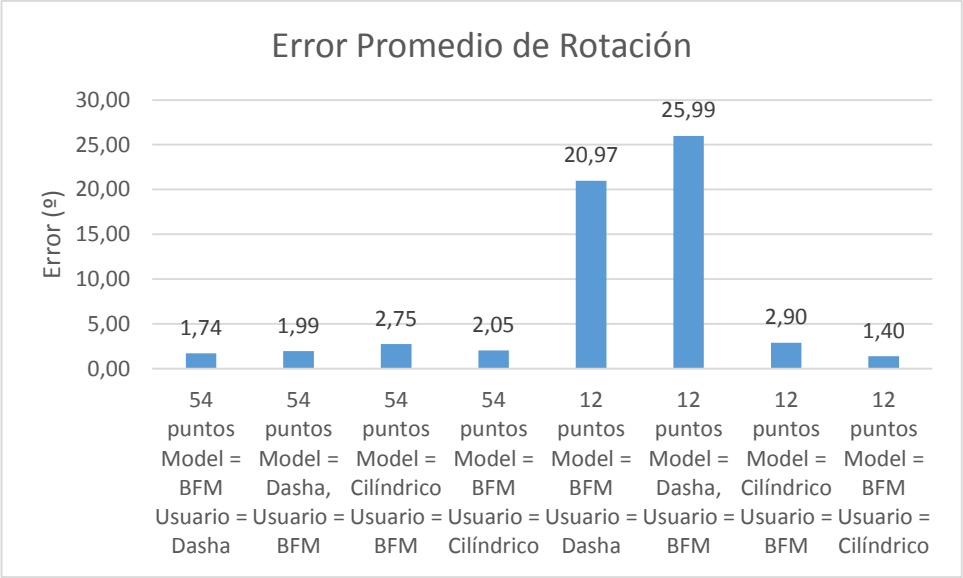
7.4.2 Combinación de diferentes modelos

Respecto a traslación, podemos apreciar unos errores muy elevados, esto es debido a la diferencia de elección del origen de coordenadas del modelo. Estos errores podrían ser corregidos con un registro entre los modelos.

Respecto a rotación, los resultados son muy aceptables utilizando 54 puntos y es un resultado muy interesante que el error estimado de rotación entre modelos distintos esté alrededor de los 2 grados. En el análisis de la utilización de 12 puntos en modelos completamente diferentes se determina que no es admisible ponerlo en práctica.

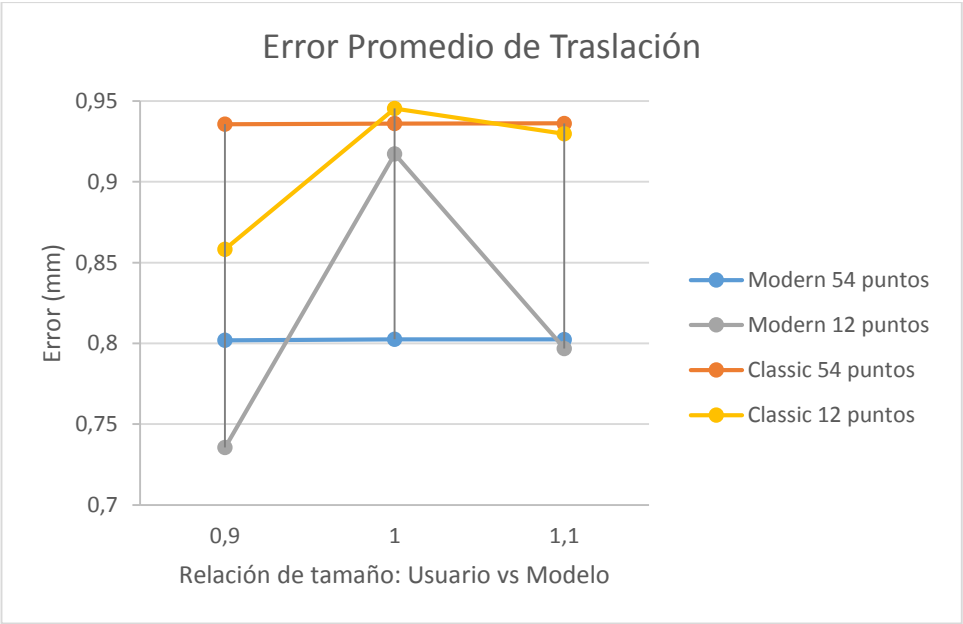
La utilización del modelo cilíndrico nos va a permitir la simplificación computacional de determinadas pruebas, ya que proyectando el modelo en el cilindro se obtienen unos resultados muy buenos, tanto en la utilización de 12 como de 54 puntos.

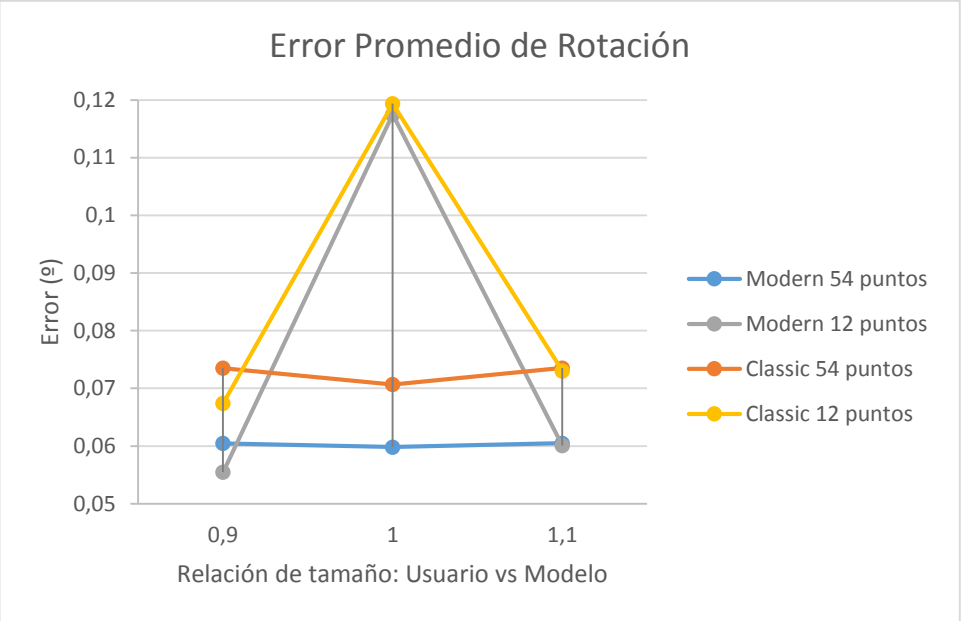




7.4.3 Relación de tamaño entre usuario y modelo

Tras el estudio realizado resulta muy llamativo que utilizando 12 puntos, sobre todo en rotación obtengamos peores resultados cuando el modelo es idéntico al usuario (Relación 1:1). Esto es algo que no se esperaba y que posiblemente sólo ocurra en casos ideales. Vemos que en consonancia con las pruebas anteriores los mejores resultados son ofrecidos por el *ModernPosit.* La relación de tamaños está dada de la forma Tamaño Usuario : Tamaño Modelo



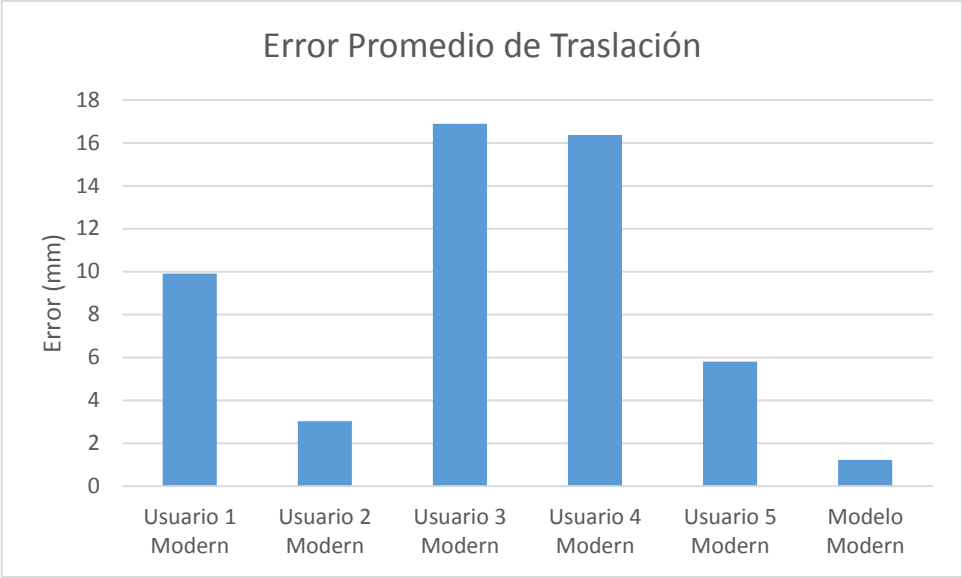


7.4.4 Diferente usuario respecto al mismo modelo

Tras elegir 5 usuarios aleatorios de los disponibles en la base de datos del modelo BFM podemos analizar el error que se obtiene al referenciarlos respecto al modelo BFM 3D.

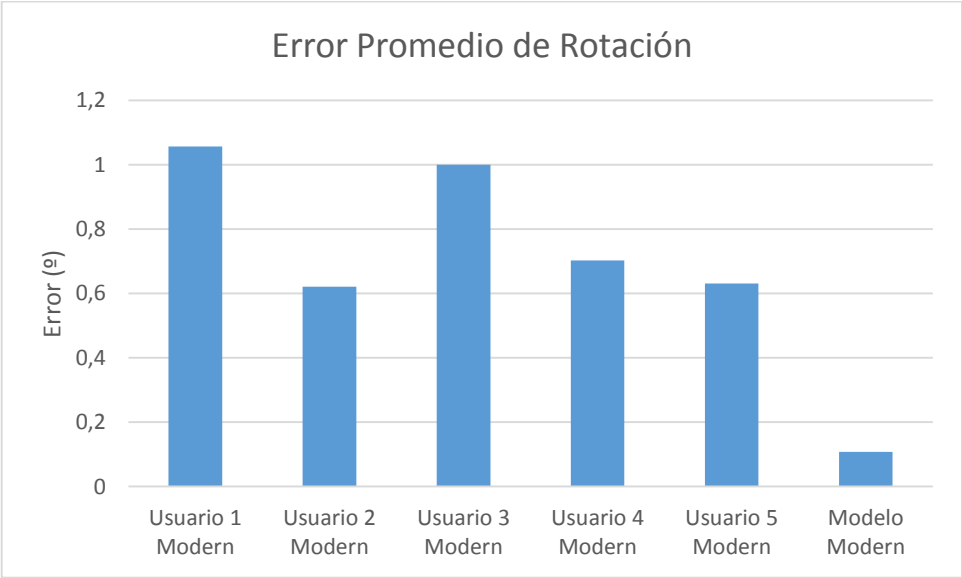
El error de estimación del usuario respecto de su mismo modelo está en 1.22 mm y 0.11º. Por lo que esto será la estimación más precisa de modernPosit.

Respecto a los demás usuario vemos errores muy dispares en la estimación de la traslación, estos son debidos sobre todo a la componente en el eje Z y principalmente causados por la posición del origen de coordenadas.



En la estimación de la rotación los errores son mucho más alentadores ya que vemos que oscilan entre 0.6º y 1º de error en promedio.

Particularmente destacaría de esta prueba que la estimación de la traslación al venir determinada por el origen de coordenadas no es precisa, aunque si podría mejorarse haciendo un ajuste de ellos. Sin embargo los resultados en orientación sí que son adecuados en los rangos de error que estamos observando a lo largo de las pruebas.



8 Conclusiones

Para la aplicación real de POSIT, el algoritmo classicPosit no nos ofrece ninguna ventaja por lo cual quedaría sustituido por modernPosit si conocemos la correspondencia entre los puntos y el modelo.

En caso de que no se conozca la correspondencia de los puntos con el modelo el algoritmo a utilizar será SoftPosit pero sólo si el número de puntos es similar al número total de puntos del modelo.

El uso de SoftPosit en aplicaciones en tiempo real no es posible. Si será de utilidad para obtener la correspondencia entre los puntos del usuario y los del modelo en el primer frame. A partir de ese momento se tendrá que utilizar modernPosit.

En caso de poder determinar el número de puntos característicos será conveniente elegir como mínimo 24 puntos.

La utilización de los 4 puntos marcados en el tabique nasal mejora idealmente en un 10% la estimación de la *pose.*

La distancia de trabajo afecta negativamente cuando más cerca de la cámara se encuentre el usuario. En las simulaciones realizadas podemos afirmar que trabajando a 50 cm de la cámara como posición inicial se obtienen estimaciones de ruido adecuadas.

El desalineamiento del modelo 3D añade una componente no fija de error a las estimaciones de POSIT. Por tanto no puede ser ajustado correctamente aunque sea identificado.

Respecto a la resolución mínima requerida si el seguimiento de puntos se realiza a través de pixeles enteros deberemos utilizar 800x600 pixeles.

El ruido en los puntos es el principal problema de POSIT, si no se consiguen proyecciones correctas el incremento de error en la estimación es superior a si se consiguen eliminar dichos puntos de la proyección.

La aplicación de pesos a POSIT ofrece mejoras incluso del 24% sin un incremento visible del tiempo computacional.

Las diferencias de la proyección 2D o del modelo 3D afectan en el mismo orden de magnitud a la estimación de POSIT.

Si se estima la *pose* de diferentes usuarios respecto al mismo modelo 3D se pueden obtener resultados de rotación en torno a 1º si estos están perfectamente emparejados en rotación. Respecto a la traslación debería realizarse con algún método que elimine el error fijo dado por la posición del origen de coordenadas.

9 Bibliografía

[1] Daniel F. DeMenthon and Larry S. Davis, Model-Based Object Pose in 25 Lines of Code

[2]Daniel F. DeMenthon and Larry S. Davis, SoftPOSIT: Simultaneous Pose and Correspondence Determination, 2003

[3] IntraFace project, Human Sensing Laboratory – Robotics Institute, CMU and University of Pittsburgh

[4] Sagi Katz, Ayellet Tal and Ronen Basri, Direct Visibility of Points Sets. Julio 2007

[5] Barber, C. B., D.P. Dobkin, and H.T. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," ACM Transactions on Mathematical Software, Vol. 22, No. 4, Dec. 1996, p. 469–483.

[6] David Roncal, Técnicas de Seguimiento de Puntos Faciales y su Efecto en la Estimación de la Posición 3D de la Cabeza, 2014

[7] José Javier Bengoechea, Comparativa de Algoritmos de Visión Monocular Para la Estimación de la Posición de la Cabeza, 2014

ANEXO 1: Manual de Usuario

El simulador ha sido diseñado para que el usuario sea el encargado de modificar un script con las opciones que desea reproducir. En dicho script lo que se hace es preparar una estructura de nombre *Simulation* la cual agrupará todos los detalles requeridos.

Los datos referidos a la cámara irán en otra estructura diferente de nombre *Camera_parameters.*

Enuncio a continuación todas las posibles opciones a configurar por el futuro usuario.

Parámetros de la cámara

***Camera_parameters.foc**=[Distancia Focal]'*

***Camera_parameters.c**=[Centro de proyección]'*

***Camera_parameters.k**=[Distorsión radial y tangencial]*

***Camera_parameters.alpha**= Relación pixel*

***Camera_parameters.resolucion_Camera**=[Resolución de Calibración]*

***Camera_parameters.resolucion_video**=[Resolución de Video]*

***Camera_parameters.Nframesseg** = Fotogramas por segundo*

Los datos siguientes se corresponden con la calibración de la cámara que deseemos usar en el simulador.

Distancia focal: Podrá ser un único valor o un vector de tamaño 1x2.

Centro de proyección: Podrá ser un único valor o un vector de tamaño 1x2.

Distorsión radial y tangencial: Será un vector de tamaño 1x5.

Relación pixel: Tendrá valor 0 si el pixel es cuadrado y distinto en caso contrario.

Resolución de Calibración: Será [Resolución horizontal, Resolución vertical].

Resolución de video: Mismo formato que la resolución de calibración [Resolución horizontal, Resolución vertical].

Fotogramas por segundo: Valor único con el número de fotogramas por segundo del video.

Determinar la posición y rotación de cada frame.

Simulation.translation *= {{Traslación eje X}, {Traslación eje Y}, {Traslación eje Z}}*

Simulation.translation_check *={{Frame checkpoint X},{ Frame checkpoint Y},{ Frame checkpoint Z}}*

Simulation.rotation *= {{Rotación ROLL}, { Rotación YAW}, { Rotación PITCH}}*

Simulation.rotation_check *={{Frame checkpoint ROLL },{ Frame checkpoint YAW },{ Frame checkpoint PITCH }}*

Simulation.aceleration *= [Aceleración traslación ; Aceleración Angular]*

Traslación eje X, Y, Z: podrá ser un valor o varios de ellos en forma de vector 1xN.

Frame checkpoint X, Y, Z: tendrá el mismo número de elementos, ya que indica el valor de la componente determinada y el frame.

Rotación ROLL, YAW, PITCH: la funcionalidad es igual al caso de la translación

Frame checkpoint ROLL, YAW, PITCH: la funcionalidad es igual al caso de la translación

Simulation.aceleration. Este determinará la aceleración del movimiento de la cabeza. Se podrá asignar una aceleración de traslación y otra de rotación ambas de la forma [x y z] ó [roll yaw pitch]. Si se desea que se haga de forma lineal se deberá inicializar con todos los datos a 0.

Elección del modelo

Simulation.Nframes *= Fotogramas del vídeo*

Simulation.model3D *= Modelo Completo: ‘*dasha’, ‘suizos’ o ‘cilindro’

Simulation.world *= Worldpoints: ‘*dasha’, ‘suizos’ o ‘cilindro’

Simulation.ptos *= Usuario: ‘*dasha’, ‘suizos’ o ‘cilindro’

Simulation.puntos *= Número de puntos: 12, 54 ó 58*

Simulation.cilindro *= [Eje X, Eje Z, altura Eje Y]*

Fotogramas del video: número de fotogramas que tendrá el video, deberá ser un número mayor que el último *checkpoint* asignado en la composición del video.

Modelo Completo: 'dasha', 'suizos' o 'cilindro'. Consiste en el modelo total a cargar, será utilizado principalmente para determinar la oclusión de los puntos.

Worldpoints: 'dasha', 'suizos' o 'cilindro'. Son los puntos prefijados de cada modelo 3D.

Usuario: 'dasha' 'suizos' o 'cilindro'. Corresponden con los puntos 3D de los cuales se obtendrá la proyección en el plano imagen 2D.

Número de puntos: 12, 54 ó 58. Se han prefijado unos conjuntos de 12 y 54 puntos. Si se desea elegir otros puntos se cargará el conjunto completo de 58 puntos y se podrá elegir los puntos determinados a través de un vector con sus índices.

Cilindro: [Eje X, Eje Z, altura Eje Y]. Se deberá dar un único valor para cada dato que queda completamente explicado por su nombre para obtener el cilindro con base elíptica.

Aplicación de modificadores

Simulation.tracker *= 'recupera' / 'No recupera'*

Simulation.oclusion *= 'Si' / 'No' / 'Cercano'*

Simulation.redondeoPixel *='Si' /' No'*

Simulation.ruidoRotation *= 'Si' / 'No'*

Simulation.ruidoRotation_sigma_max *= [Ruido Rotación]*

Simulation.sigma2D *= [Ruido2D]*

Simulation.sigma3D *= [Ruido3D]*

Simulation.puntos_perdidos *= {{Índice del punto},{ Índice del punto }}*

Simulation.puntos_perdidos_Frame *= [Número de frame];*

Simulation.engancha_puntos *= {{Índice del punto},{ Índice del punto }}*

Simulation.engancha_puntos_Frame *= [Número de frame];*

Tracker: Determina si tras perder visibilidad recupera el punto al recuperarla.

Oclusión: Controla si hay oclusión de puntos (‘Si’), si estamos ante un caso ideal donde no se ocluye ningún punto (‘No’), o si se aproxima el punto al más cercano de los visibles (‘Cercano’)

RedondeoPixel: Determina si se desea redondear la proyección 2D a pixeles enteros.

RuidoRotation: Aplicará ruido determinado por la posición y rotación del modelo.

RuidoRotation_sigma_max: Será un vector 1x2 con los valores de la desviación estándar de ruido que se desea aplicar en el plano imagen. Si no deseamos ruido de este tipo: [0 0] ó *RuidoRotation = ‘No’.*

Ruido2D: Será un vector 1x2 con los valores de la desviación estándar de ruido que se desea aplicar en el plano imagen. Si no deseamos ruido [0 0].

Ruido3D: Será un vector 1x3 con los valores de la desviación estándar de ruido que se desea aplicar en el modelo 3D. Si no deseamos ruido [0 0 0].

Puntos_perdidos: Se indicará el índice de los puntos que se desean perder, disponemos de varias celdas para determinar junto a *Puntos_perdidos_frame* el fotograma donde se van a perder.
Engancha_puntos: mismo funcionamiento que puntos_perdidos. Servirá para que un punto proyectado mantenga su valor fijo en los frames posteriores del video.

Conjunto de pruebas unificadas

Se ha optado por incluir dos bucles que engloben todo el proceso en donde se realiza la separación por cada usuario y por cada video a procesar.

Para ello se deberá controlar los valores del número total de usuarios (n_*user*) y de vídeos (n_*video*). Una vez conocido este detalle se comprenderá que el índice *i* determina el usuario y el índice *j* el número de video.

Considero que queda bastante bien marcado por medio de dos bucles condicionales IF, donde se debe modificar el código del script para determinar que usuario y en que video se están incluyendo todos los parámetro explicados previamente.

ANEXO 2: FAQ

- Vamos a generar una serie de caras distintas sobre la base PCA del modelo de los suizos, que van a ser nuestros distintos sujetos.

Los modelos son archivos .mat .Tengo 3 funciones donde hay que cargarlos:

→**model_choice.m** : archivo .mat de la forma Nx3. Se carga el modelo completo

→**worldpoints_choice.m** : archivo .mat de la forma Nx3. Se cargan los puntos que deseamos que sean los worldpoints para cuando los requiere POSIT.

→**points_choice.m** : archivo .mat de la forma Nx3. Se cargan los puntos que "trackeamos".

La forma de cargar todo esto es muy intuitiva, solo hay que copiar las líneas anteriores referentes a la carga de otros modelos y modificar el nombre del archivo que estamos cargando. Importante es decir que los worldpoints y los points deben coincidir en número.

Una vez las funciones estén modificadas, le asignamos un nombre a los datos que queremos cargar y la determinación de que modelo usar lo hacemos en las siguientes líneas donde cargamos el modelo de los suizos y los points son 54x3.

```
Simulation.model3D = 'suizos'; %Entrada de model_choice.m
Simulation.world = 'suizos'; %Entrada de worldpoints_choice.m
Simulation.ptos = 'suizos'; %Entrada de points_choice.m
Simulation.puntos = 54; % %Entrada de points_choice.m
```

Esta última variable quizá no la necesites depende de si incluyes el case de las funciones points_choice y worlpoints_choice o no.

- Con cada una de estas caras generaremos 12 videos distintos, como los de la base de datos (3 traslaciones, 3 rotaciones, y 6 libres). Es probable que utilicemos las mismas secuencias de rotaciones y traslaciones para cada usuario, aunque esto está aún por determinar. Así que estaría bien que nos contaras cómo utilizar estas funciones para generar secuencias de N frames controlando los rangos de rotación y traslación.

En la función principal tendremos el **Simulation.translation y Simulation.rotation** con sus pares de **Simulation.translation_check y Simulation.rotation_check**. En los primeros indicamos el valor que debe tener en el frame "emparejado". Ya en la función **checkpoints.m** calcula el valor en cada frame en forma uniformemente lineal. Importante es guardar correspondencia entre los los pares e indicar correctamente el valor en el frame determinado. No hay límite de checkpoints.

Por ejemplo abajo doy 4 variaciones en X (valor 0 en el frame 0,valor 190 en el frame 75….), con Y y Z constantes. Además no hay ninguna rotación en el video.

```
Simulation.translation = {{0 190 -190 0},{0},{600}}; %{{X},{Y},{Z}}
Simulation.translation_check={{0 75 225 300},{0},{0}}; %Frame
Simulation.rotation = {{0},{0},{0}}; %{{ROLL},{YAW},{PITCH}}
Simulation.rotation_check={{0},{0},{0}}; %Frame
```

En Simulation.Nframes determinamos el número de frame que tendrá el video.

- Como ves, es todo muy similar a lo que ya has hecho tú anteriormente. Una de las principales diferencias es que queremos que las secuencias comiencen con una rotación distinta de 0. Este offset inicial será pequeño, y en promedio además será también distinto de 0 para todos los usuarios. Imagino que esto lo podremos hacer sin ningún problema tras la explicación de cómo llevar a cabo el punto anterior.

Si en el punto anterior le da igual empezar en 0 o no. Lo del cálculo de que sea distinto en promedio para los demás usuarios no sé si lo calcularas antes. Sino deberías crear una función que te lo de y de manera fácil concatenar la estructura. Si va ser un cálculo sólo es más fácil hacerlo previamente y meter el valor a mano.

- Con lo anterior, necesitamos obtener las nubes de puntos 2D correspondientes a cada frame, para cada video de cada usuario, y tener claro el formato de estos datos para después procesar. Todo esto sin ruido, sin eliminar oclusiones ni nada, como en una situación de tracking perfecto.

Para simular un tracking perfecto:

```
Simulation.oclusion= 'No'; % No hay oclusiones
Simulation.redondeoPixel='No'; %No hay redondeo de las proyecciones a Pixel entero
Simulation.ruidoRotation = 'no'; %No hay ruido determinado por la rotación
Simulation.noise = 'sin noise'; % No hay ruido 3D
Simulation.tracker= 'recupera'; % Si no hay oclusiones ni pérdidas de puntos no te afecta pero con todo en ‘recupera’.
```

Los puntos de la proyección 2D de cada frame los obtienes en la variable **video.projection2D54** donde tendrás 2xN (donde N es el número de puntos) para cada frame que almacena la estructura.