MEMORIA PROYECTO FIN DE CARRERA:

DESARROLLO SISTEMA DE SIMULACIÓN Y CÁLCULO DE RESULTADOS PARA SONDA DE TRIANGULACIÓN LÁSER

RUBÉN GAZOL ESCRIBANO
ING. TÉC. INDUSTRIAL
ESP. MECÁNICA

1. INTRODUCCIÓN

El proyecto que se detalla en esta memoria ha sido realizado por Rubén Gazol Escribano bajo la supervisión del Carlos E. Cajal, director de proyecto, perteneciente al departamento de Ingeniería de Diseño y Fabricación, perteneciente al área de Procesos de Fabricación de la Universidad de Zaragoza.

Este proyecto forma parte de uno de mayor envergadura consistente en el diseño, desarrollo e implantación de un sistema de triangulación láser con el fin de verificar tolerancias y planitud en una línea de producción de intercambiadores de calor en el sector del automóvil. Este sistema permitirá verificar el 100% de la producción de manera rápida, eficaz y sencilla, permitiendo aceptar o desechar piezas fabricadas según el resultado dado por el sistema.

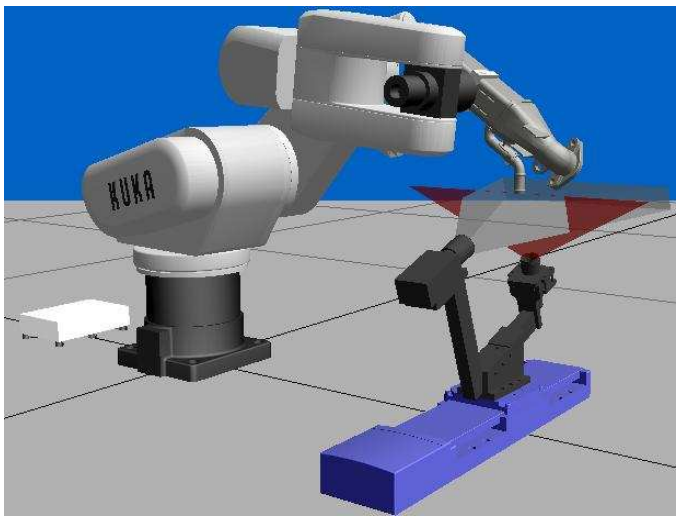

Ilustración 1. Representación del sistema del proyecto.

En este proyecto, se utiliza un láser, que será el encargado de realizar el barrido sobre las caras de la pieza que quieras que será el encargado de recoger todos los puntos que se requieran.

Se utilizará también un brazo robótico modelo Kuka que será el encargado de posicionar la pieza correspondiente en la posición que se requiera para realizar la recogida de datos. Este brazo deberá coger y dejar la pieza que se vaya a analizar.

La cámara y el láser corren por una cremallera, el mismo movimiento que los desplaza por la cremallera hace que el haz de rayos del láser realice el barrido y se recojan los datos necesarios.

1.1 OBJETIVOS

El objetivo principal del proyecto es simular el comportamiento del equipo, siempre referido la parte correspondiente a la toma de datos que realizará el mismo cuando esté en funcionamiento, además de detectar posibles problemas y/o potenciales mejoras.

Se trata de poner a prueba el simulador, realizar simulaciones con distintos parámetros de cámara y distintas piezas, en varias posiciones y comparar y analizar los datos obtenidos, programar el tratamiento de datos obtenidos de la mejor manera.

Para comparar de manera eficaz los datos obtenidos según las distintas simulaciones realizadas habrá que realizar varias aplicaciones, determinando cual será el mejor método sobre el que deban trabajar estas aplicaciones.

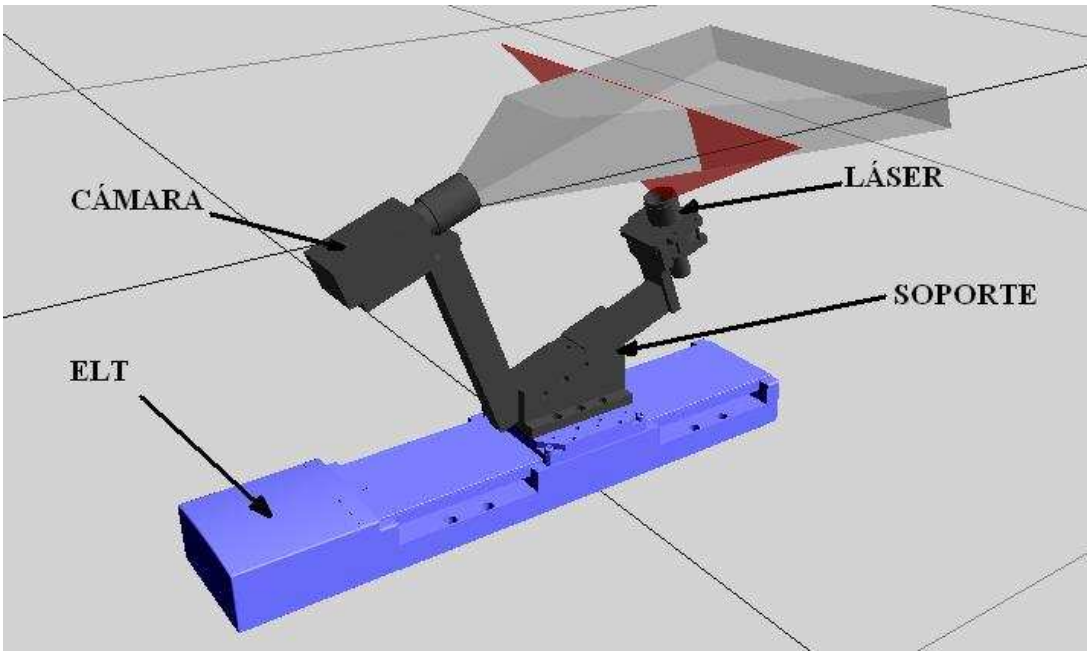

Ilustración 2. Módulo del sistema que realiza las operaciones y en cuyo funcionamiento se basa este proyecto.

Además, a los datos que se obtienen con estas simulaciones se le añaden ruido o vibraciones que simulan posibles "tropiezos" del sistema de cámara durante su movimiento por la cremallera que los dirige.

1.2 ALCANCE

Este sistema da lugar a satisfacer las exigencias en cuanto a calidad y tolerancias se refiere, ya que permite la verificación del 100% de la producción y con la pieza en las posiciones que se requiera.

Además el sistema estudiado se puede aplicar para cualquier pieza que se fabrique y requiera gran exactitud en planitud y dimensiones. El sistema de triangulación láser permite verificar longitudes, diámetros de agujero, posicionamiento de los agujeros y plenitud de una manera rápida y sencilla.

2 PLANIFICACIÓN

El proyecto se comenzó a mediados de Septiembre de 2009, con un primer contacto con el funcionamiento de cada uno de los aparatos que componen el conjunto de la máquina y así poder comprender el modo de funcionamiento del conjunto y como deben sincronizarse cada una de las partes con las otras.

Las tareas llevadas a cabo que se detallarán más adelante son las siguientes:

- Familiarizarse con el sistema cámara-láser y su funcionamiento mediante la lectura del resto de proyectos que forman parte del proyecto que engloba este.
- Aprendizaje del lenguaje de programación de MATLAB.
- Realizar la simulación sobre la pieza base del modelo XS2002.
- Crear un código, con el que introducir ruido a los puntos recogidos en las simulaciones.
- Buscar el método más adecuado de introducir el ruido creado anteriormente por líneas de puntos en lugar de aplicar un ruido distinto a cada punto.
- Limitar el ruido creado y añadido a los archivos creados durante la simulación a los valores requeridos.
- Realizar el calibrado de la cámara y comprobar que todas las creaciones anteriores funcionan correctamente.

- Añadir ruido a nubes de puntos correspondientes a piezas o partes de las mismas que requieran mayor atención o sean más interesantes.

- Cálculo del plano láser que realiza el barrido, necesario para realizar el cambio de coordenadas.

- Realización de la Memoria Proyecto Fin de Carrera.

3 DESCRIPCIÓN DEL TRABAJO REALIZADO

A continuación se citan y explican las tareas que se han llevado a cabo a lo largo del tiempo en que se ha desarrollado el completo de este proyecto, desde el aprendizaje y toma de contacto con los programas necesarios para el trabajo hasta la realización, por último, de la presente memoria.

3.1 APRENDIZAJE DE SOFTWARES

Para el correcto desarrollo del proyecto se han debido utilizar distintos software, los cuales se presentan y describen a continuación.

3.1.1 MATLAB R2007

MATLAB es un potente lenguaje diseñado para la computación técnica, puede se utilizado en computación matemática, modelado y simulación, análisis y procesamiento de datos, visualización y representación de gráficos, así como para el desarrollo de algoritmos.

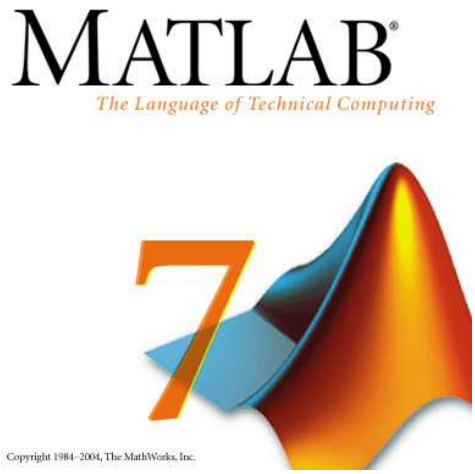

Ilustración 3. Portada Matlab.

MATLAB es ampliamente conocido y utilizado en universidades e institutos para el aprendizaje en cursos básicos y avanzados de matemáticas, ciencias y especialmente, como es el caso, en ingeniería. En la industria se utiliza habitualmente en investigación, desarrollo y diseño de prototipos. El programa estándar de MATLAB comprende una serie de herramientas que pueden ser utilizadas para resolver problemas comunes, pero además, incorpora otras librerías específicas llamadas toolboxes, que son colecciones de funciones especializadas y diseñadas para resolver problemas muy específicos.

Además, MATLAB también permite diseñar y editar interfaces de usuario, permitiendo introducir listas de selección, menús desplegables, botones de pulsación, botones de opción y deslizadores, así como diagramas y controles ActiveX. También puede crear interfaces gráficas de usuario por medio de programación usando las funciones de MATLAB.

Sus características principales son:

- Lenguaje de alto nivel para cálculo técnico.
- Entorno de desarrollo para la gestión de código, archivos y datos.
- Herramientas interactivas para exploración, diseño y resolución de problemas iterativos.
- Funciones matemáticas para álgebra lineal, estadística, análisis de Fourir, filtraje, optimización e integración numérica.
- Funciones gráficas bidimensionales y tridimensionales para visualización de datos.
- Herramientas para crear interfaces gráficas de usuario personalizadas.
- Funciones para integrar los algoritmos bastados en MATLAB con aplicaciones y lenguajes externos, tales como C/C++, FORTRAN, Java, COM y Microsoft Excel.

3.1.2 SOLID EDGE v18

Solide Edge es un programa de parametrizado de piezas en 3D basado en un software de sistema de diseño asistido por ordenador (CAD). Permite el modelado de piezas de distintos materiales, doblado de chapas, ensamblaje de conjuntos, soldadura, funciones de dibujo en plano para ingenieros.

Este es uno de los paquetes instados a enterrar el uso masivo del CAD 2D dando paso al CAD 3D.

Ilustración 4. Portada Solid Edge

Solid Edge es un potente software CAD 3D que permite a los fabricantes diseñar con inteligencia y reducir los plazos de comercialización, aumentar la calidad y disminuir los costes. Su tecnología incorpora las capacidades de gestión del diseño directamente en CAD, lo que aporta eficiencia a la intención del diseño en toda la empresa y aumenta la colaboración. Es capaz de trabajar sin ningún tipo de problema con proyectos realizados con diferentes programas de diseño mecánico de CAD.

Sus principales características son:

- Diseña con cotas en 3D.
- Reconoce y mantiene la intención del diseño con Reglas Activas.
- Edita operaciones sin regeneración
- Edita la geometría sin importar el orden de creación.
- Ensamblaje de distintas piezas para formar una única.
- Simulación del comportamiento de la máquina bajo Reglas Activas.
- Explosionado de conjuntos.

3.1.3 AUTODESK 3DS MAX 9

Autodesk 3ds Max es un programa de creación de gráficos y animación 3D. Dispone de una sólida capacidad de edición, una omnipresente arquitectura de plugins y una larga tradición en plataformas Microsoft Windows.

Este software tiene 2 versiones, una especializad con herramientas específicas para los desarrolladores de juegos, realizadores de efectos visuales y diseñadores gráficos, mientras que la otra tiene características especializadas para arquitectos, diseñadores, ingenieros y especialistas en visualización.

Ilustración 5. Portada 3D Studio Max

Sus principales características son:

- Amplio juego de herramientas de modelado 3D que permiten el modelado poligonal y diseño de formas libres en 3D.
- Posee una gran variedad de opciones para pintar texturas, mapearlas y asignarlas a capas, para dar el sombreado y la textura deseada.
- Crea animaciones en 3D de alta calidad.
- Posee herramientas de producción con alto rendimiento para crear efectos y dinámica.
- Es capaz de importar datos de numerosos orígenes, y de transferir información de 3ds Max entre archivos, aplicaciones, usuarios y ubicaciones.
- Recopila y permite el uso compartido de datos en escenas complejas, para que múltiples usuarios puedan colaborar en el flojo de trabajo.

3.1.4 GEOMAGIC STUDIO 10

Geomagic Studio transforma datos escaneados en 3D y mallas de polígonos de modelos 3d precisos para la ingeniería inversa, diseño del producto, prototipazo rápido y análisis.

Geomagic Studio es capaz de reconocer archivos exportados de los principales paquetes de CAD mecánico, proporcionando además capacidades de modelado paramétrico, así como funciones para la captura de geometría exacta, que le da la potencia y flexibilidad para elegir el método de modelado que mejor se adapta para su aplicación.

Ilustración 6. Portada Geomagic Studio

En este caso, será utilizado solo para representar la planitud de las simulaciones realizadas. Ya que es capaz de colorear cada uno de los puntos de la pieza de un color según la "cota" en la que esté situado, dando así una imagen en la que a primera vista se pueden sacar conclusiones más que aptas.

3.2 DOCUMENTACIÓN GENERADA

- REALIZACIÓN DE DISTINTAS SIMULACIONES CON PARÁMETROS VARIABLES DE LA PIEZA BASE DEL MODELO XS2002.

En un primer contacto, con el modo de trabajo y captación de datos en la simulación, se realizaron las simulaciones o barridos correspondientes a las bases de cada uno de las piezas modelo.

Dichas simulaciones se realizaron de dos maneras distintas, y bajo las siguientes condiciones:

- Paso 0,2 mm (avance de 20 mm/s y captura de 100 frames/s)
- Ángulo de láser respecto de la vertical constante a 15º.

- Ángulo de la cámara respecto de la vertical variable desde 85º a 15º en saltos de 2º.
- Representación en MATLAB cada 10º con detalle de los agujeros avellanados.

Este modo de realizar simulaciones, nos permite, una vez representado los puntos en MATLAB, comparar los datos recogidos por la simulación en cada posición de la cámara. A primera vista, se aprecia que una posición vertical de la cámara permite observar mejor el interior de los avellanados, la posición de cada uno de los agujeros o puntos de referencia que tenga la pieza y de esta manera poder comprobar que la situación de distintos elementos de la pieza están colocados de manera correcta y acorde a lo necesario para su utilización.

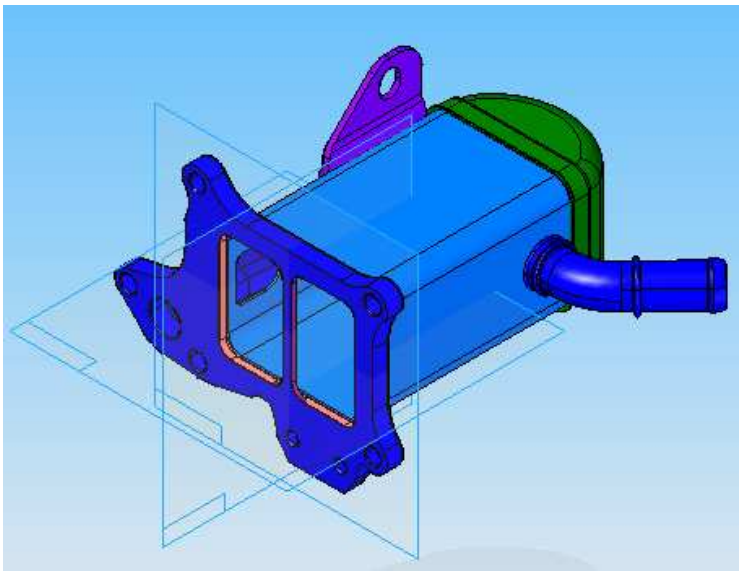

Ilustración 7. Imagen del modelo XS2002. En azul oscuro a la izquierda, la base sobre la que se han realizado las simulaciones.

Una posición más horizontal de la cámara (ángulo de 85º con la vertical) permitirá, más adelante, la representación y verificación de la plenitud de la pieza, permitiendo ver errores que hagan que el acoplamiento del intercambiador de calor a la base en la que esté fijado no sea el apropiado,

impidiendo el buen funcionamiento o rebajando el rendimiento de la máquina que lo haga funcionar.

Un segundo modo de realizar simulaciones permite ver con mayor o menor detalle los puntos importantes de la pieza. Este modo, se realizó bajo las siguientes condiciones.

- Ángulo de láser fijo a 20º respecto de la vertical.
- Ángulo de cámara fijo a 70ª respecto de la vertical.
- Simulaciones realizadas con paso variable desde 0,01 mm hasta 1 mm en saltos de 0,1 mm.

Este modo de realizar simulaciones permite realizar simulaciones con distinto detalle de la pieza.
El paso de la simulación viene marcado por los siguientes parámetros, cada uno de ellos requiere una velocidad de avance del carro y un número de recogida de datos por segundo, determinados por los denominados frames.
Un frame es la toma de datos de las coordenadas de cada uno de los puntos que forman parte en ese momento del haz de láser y de la pieza, formando así las líneas que se aprecian en cada una de las representaciones realizadas en MATLAB.

Sabiendo que:

$$Paso = \frac{Velocidad}{Frames} \quad \text{Siendo las unidades:} \quad mm = \frac{mm/s}{frames/s}$$

En la siguiente tabla aparecen los valores de la velocidad de avance y los frames/s utilizados para determinar cada uno de los distintos pasos:

Paso (mm)	Velocidad (mm/s)	Frames/s
0.01	20	2000
0.1	20	200
0.2	20	100
0.3	33	110
0.4	20	50
0.5	20	40
0.6	33	55
0.7	77	110
0.8	20	25
0.9	9	10
1	20	20

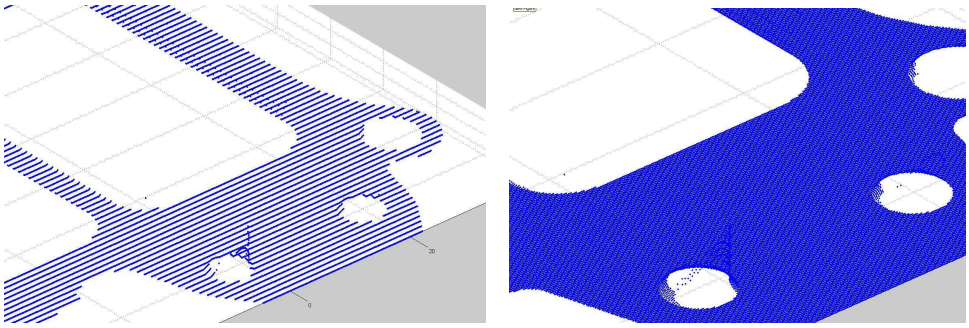

Ilustración 8. A la izquierda representación de la simulación realizada con paso de 1 mm y a la derecha la misma pieza con paso 0,2 mm.

- CÁLCULO DEL ÁNGULO DE APERTURA DE LA CÁMARA PARA LAS CONDICIONES EN QUE SE HA REALIZADO LA SIMULACIÓN

Para calcular el ángulo de apertura de la cámara utilizaremos la aplicación llamada “Aplicación cálculo longitud-píxel”.

El cálculo se ha realizado para una distancia de la cámara respecto de la pieza de 200 mm y una distancia focal de 12 mm.

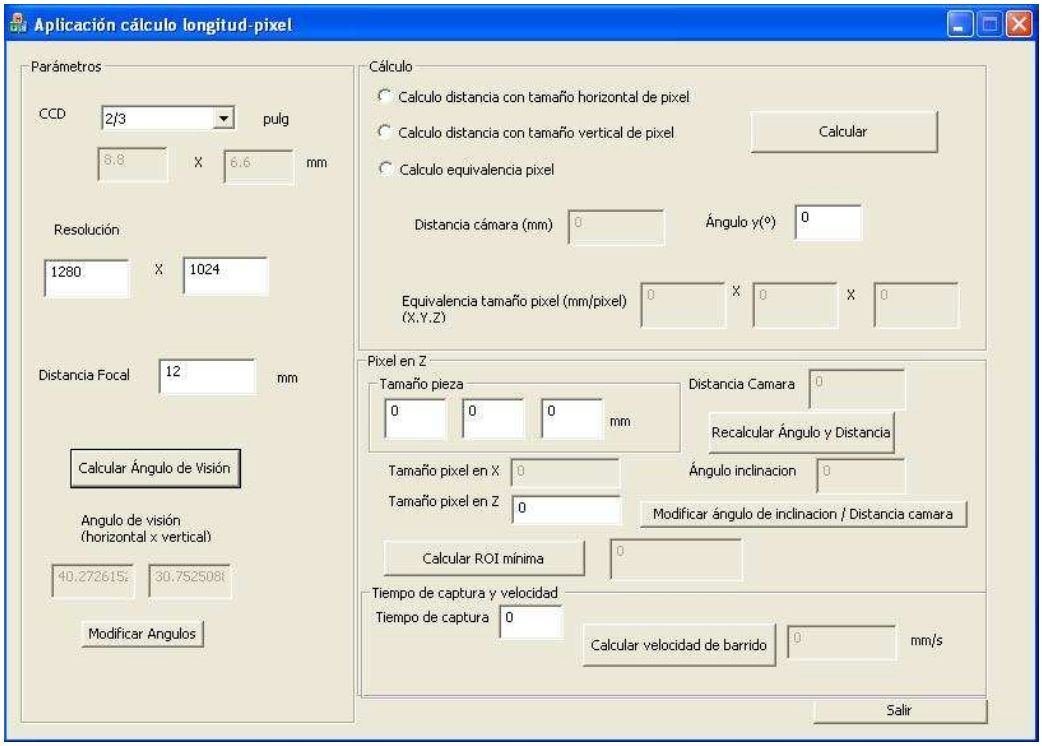

Ilustración 9. Apariencia de la aplicación Cálculo longitud-píxel.

El resultado es el que se muestra a continuación (ampliado de la imagen anterior).

Ilustración 10. Resultado del cálculo del ángulo de apertura de la cámara.

- INTRODUCCIÓN DE RUIDO EN LAS REPRESENTACIONES GRÁFICAS DE MATLAB DE LAS COORDENADAS (U,V) RECOGIDAS EN LA SIMULACIÓN.

El objetivo prioritario de esta tarea tenía como objetivo establecer el mejor método de introducir ruido en las representaciones de MATLAB. En primer lugar no importaba cual fuese el ruido ni las condiciones de este, sino introducirlo, y que se viese representado.

En este caso, introducir un ruido distinto a cada una de las coordenadas de cada uno de los puntos de la manera más sencilla era crear una matriz con tantos valores como coordenadas requería la simulación. Para esto, había que dar a la matriz 2 columnas y tantas filas como hubiese en la matriz original obtenida a través de la simulación.

El primer paso era hacer que el programa generase una matriz de valores, los cuales se sumarían al valor de cada una de las coordenadas de cada uno de los puntos. Esta matriz debía contener tantas columnas como el número de coordenadas a las que se les metiese ruido, y tantas filas como puntos se hubiesen registrado en el archivo .txt durante la simulación del barrido.

Cada punto registrado tiene dos tipos de coordenadas, de posición (coordenadas XYZ) y de pantalla (coordenadas UV). En este caso se realizó con las coordenadas de pantalla, por lo que la matriz debía poseer 2 columnas y tantas filas como el archivo cargado en el programa.

El código encargado en generar dicha matriz es:

```
Ruido=random ('Normal',A,B,F,C);
```

En la cual:

- La variable “Ruido” tomará el valor de la matriz generada.

- La distribución que van a seguir los valores, va a ser una distribución Normal (la llamada campana de Gauss), centrada en el 0, ya que partimos de la hipótesis de que todos los puntos deberían pertenecer a un mismo plano y además correctamente.
- El valor A recoge la media de la distribución normal, en todos nuestros casos este valor será 0 por lo explicado en el punto anterior.
- El valor B determina el valor de la varianza de la distribución normal, este valor determina las probabilidades de que los puntos se alejen más o menos de la media establecida. Es decir, a mayor varianza, mayor es la probabilidad de que los puntos se alejen de la media, y mayor será ruido añadido.
- El valor F de la función corresponde al número de filas que ha de tener la matriz generada aleatoriamente (en este caso, la matriz tendrá tantas filas como puntos se hayan recogido durante la simulación de la pieza).
- El valor C de la función determina el número de columnas por las que estará formada la matriz (en este caso el número de columnas será 2).

Una vez se tiene creada esta matriz, solo queda sumarla al archivo original y representarla. Esta tarea se realiza mediante el siguiente código (se trata de un ejemplo aplicado aleatoriamente con valores también aleatorios):

```
Plot(ruido(:,1)+archivo.txt(:,3),ruido(:,2)+archivo.txt(:,5),
'b.');
Axis equal;
Grid;
```

La función Plot cumple la función de representar los puntos que se impongan, en este caso, se trata de representar los puntos generados en archivo.txt (que será el nombre con que MATLAB cargue los archivos originales para tratarlos a lo largo del completo del proyecto).

El código hace que la columna 1 y 2 de la matriz ruido se sumen a las columnas 3 y 5 de la matriz archivo.txt respectivamente dando lugar a la aparición de ruido en cada punto de la pieza.

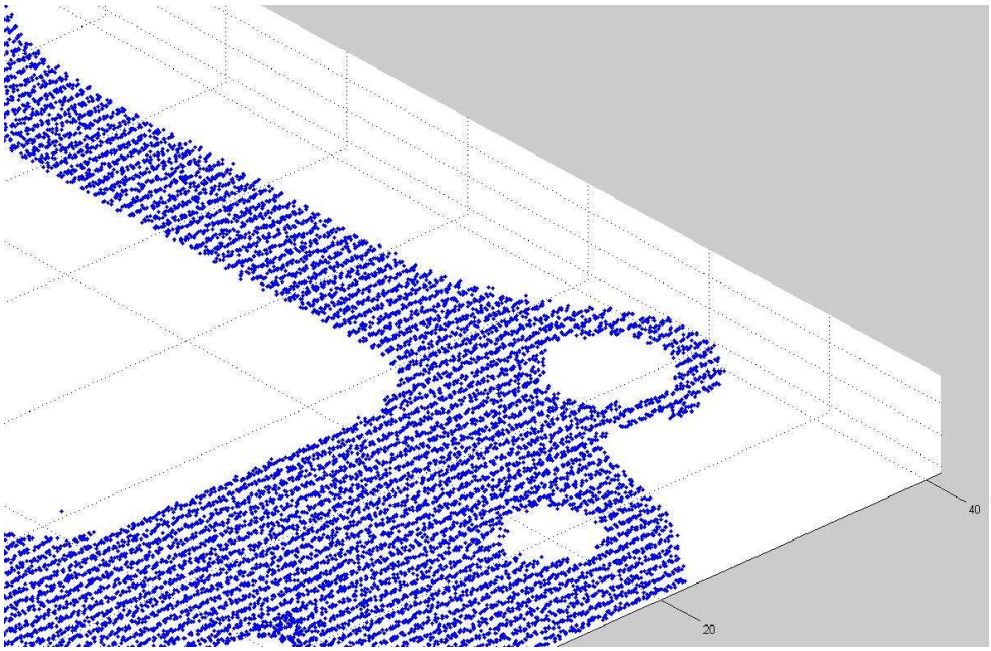

Ilustración 11. Representación de los datos recogidos en la simulación con el ruido añadido de la manera explicada anteriormente pero en los ejes XYZ.

La apariencia de desorden de los puntos se debe a que cada coordenada de cada punto tiene añadido un ruido distinto al de los puntos que la rodean. En este caso la varianza es pequeña, ya que como se aprecia en toda la imagen, los puntos mantienen la forma de la figura, salvo, a la izquierda de la imagen, en la que se aprecia que uno de los puntos ha "saltado" de su posición mucho más lejos, pues se debe a que ha recibido, en al menos una de sus coordenadas, un ruido fuera del rango marcado y mucho mayor que la del resto.

Como se explica a pie de foto, la imagen representa el añadido de ruido en las coordenadas XYZ de la pieza, el código que lo realiza es el siguiente:

```
ruido=random('Normal',0,0.1,42176,3);
plot3(paso1(:,1)+ruido(:,1),paso1(:,2)+ruido(:,2),paso1(:,3)
+ruido(:,3),'b.');
axis equal;
grid;
```

A lo largo del proyecto, veremos en numerosos puntos funciones del lenguaje que utiliza MATLAB para la programación, ya que visionar el código da lugar a un mejor entendimiento y facilita el entendimiento de las tareas que lleva a cabo y el como lo hace.

- GENERACIÓN DE UN CÓDIGO EN MATLAB DE FORMA QUE EL RUIDO INTRODUCIDO SEA IGUAL PARA TODA UNA LÍNEA DE PUNTOS.

El ruido creado anteriormente no se corresponde con la realidad, ya que, de ser así los datos recogidos por el sistema, los datos serían muy poco fiables. No se podría encontrar una relación entre puntos, ni determinar si realmente hay un defecto en la fabricación de la pieza, bien sea por planitud como por colocación de los distintos elementos que componen la pieza completa.

Para acercarlo más a la realidad, el ruido se va añadir de manera que cada una de las filas de puntos recogidas durante la simulación, recordemos que el sistema recoge las coordenadas de los puntos correspondientes a la línea por la que en ese momento “barre” el láser, tengan un ruido distinto del resto de líneas en cada eje de coordenadas, pero en la que todos los puntos de la línea tengan el mismo ruido en el mismo eje.

Puesto que las coordenadas XYZ muestran la posición en mm y las coordenadas de pantalla UV muestran la posición de los puntos en píxeles,

cada una de las coordenadas recibirá un ruido bajo unas condiciones particulares.

El método de añadir ruido a ambos sistemas de coordenadas van a ser muy similares. Se generará una matriz como la mostrada en el punto anterior, con las mismas dimensiones que la matriz del archivo.txt original que se vaya a tratar. En esta ocasión, por ser más compleja la manera de añadir ruido, se hará necesario el añadir una variable más, que se llamará resultado, y que recogerá el valor de la matriz resultante de la suma de los valores de la matriz ruido y de la matriz original.

Coordenadas de posición XYZ.

Las coordenadas XYZ recogen la posición del punto en el espacio, y lo registra en mm. Para añadir ruido a estas coordenadas establecemos las siguientes condiciones:

- El ruido se añadirá a cada línea de puntos.
- En las coordenadas de pantalla solo se añadirá ruido en la coordinada V, y este ruido deberá estar comprendido entre +3 y -3 píxeles.

El código generado para añadir el ruido es el siguiente:

```
[filas,columnas]=size(archivo);
resultado=archivo;

ruido=random('Normal',0,0.060975609756097560975609756097561,f
ilas,columnas);

contador=1;
posicion=2;
anterior=archivo(1,1);
```

```
posterior=archivo(posicion,1);
for i=1:(filas-2)

   if ((posterior+10)<anterior);
      contador=(contador+1);
   end

   resultado(i,1)=(archivo(i,1)+ruido(contador,1));
   resultado(i,2)=(archivo(i,2)+ruido(contador,2));
   resultado(i,3)=(archivo(i,3)+ruido(contador,3));
   anterior=posterior;
   posicion=(posicion+1);
   posterior=archivo((posicion),1);
end

plot3(resultado(:,1),resultado(:,2),resultado(:,3),'b.');
axis equal;
grid;
```

El código generado trabaja de la siguiente manera (por líneas de código):

```
[filas,columnas]=size(archivo);
```

Se asignan a las variables ‘filas’ y ‘columnas’ los valores de las dimensiones del matriz origen (dimensiones de la matriz que contiene el documento cargado para el trabajo llamado ‘archivo’). Estos valores serán los que más adelante determinen las dimensiones que ha de tener la matriz generada ‘ruido’.

```
resultado=archivo;
```

Además, en este caso, al trabajar por líneas de puntos con el archivo, es necesario generar una nueva variable ‘resultado’ que contenga, y vaya guardando conforme avanza el código, los resultados de la suma de los valores de la matriz ‘ruido’ y los valores de la matriz original ‘archivo’.

```
ruido=random('Normal',0,0.060975609756097560975609756097561,f
ilas,columnas);
```

De manera análoga a la mencionada en el punto anterior, se genera la matriz ‘ruido’ con valores aleatorios recogidos mediante distribución normal, con el valor medio de 0 (por las mismas razones explicadas en los puntos anteriores), el valor de varianza determinado por las condiciones establecidas para el ruido (las cuales serán calculadas en el próximo punto del proyecto) y las variables ‘filas’ y ‘columnas’ que darán dimensiones a la matriz generada.

```
contador=1;
posicion=2;
anterior=archivo(1,1);
posterior=archivo(posicion,1);
```

Para lograr identificar los puntos que forman parte de la misma línea y cual es el punto inicial de la siguiente línea es necesario crear distintas variables.

Las variables ‘contador’ y ‘posicion’ tienen la simple función de determinar la fila sobre la que se está trabajando en la matriz y la fila con la que la anterior se comparará más adelante.

Las variables ‘anterior’ y ‘posterior’ tienen la función de recoger el valor la coordenada X de los puntos. La variable ‘anterior’ recoge el valor de la coordenada X de un punto, y la variable ‘posterior’ contiene el valor de la coordenada X del punto siguiente al de la variable ‘anterior’, es decir, trabaja en la fila siguiente de la matriz.

Las líneas de código escritas más arriba son las que inicializan dichas variables, son las líneas de código en las que dichas variables recogen su primer valor de la simulación.

```
for i=1:(filas-2)

   if ((posterior+10)<anterior);
      contador=(contador+1);
   end

   resultado(i,1)=(archivo(i,1)+ruido(contador,1));
   resultado(i,2)=(archivo(i,2)+ruido(contador,2));
   resultado(i,3)=(archivo(i,3)+ruido(contador,3));
   anterior=posterior;
   posicion=(posicion+1);
   posterior=archivo((posicion),1);
end
```

Estas líneas de código forman el cuerpo del programa, son las líneas que realizan el trabajo. Y lo hacen de la siguiente manera:

El código recorre la matriz original contenida en la variable ‘archivo’ desde la primera fila hasta la última, haciendo que cada variable recoja el valor correspondiente a la fila sobre la que se esté trabajando.

Puesto que el sistema cámara-láser avanza, realizando su barrido, en dirección al eje Y, las coordenadas del eje Y de cada uno de los puntos de una misma línea será ha de tener el mismo valor, siendo así, una coordenada a seguir. Comparar estas líneas tenía el inconveniente de de los puntos pertenecientes a la parte más profunda del avellanado de los agujeros existentes, ya que variaban de coordenada Y respecto del resto de puntos que formaban la línea a la que pertenecen.

La solución a este problema se resolvió comparando las coordenadas X de los puntos. La recogida de las coordenadas de puntos, se realiza de un extremo de la línea al otro, más concretamente del lado negativo del eje X hacia los valores positivos de este mismo eje. Por lo que la variación en la coordenada X de los puntos era más o menos constante en puntos pertenecientes a la misma línea mientras que era mucho más abultada entre el último punto que conforma una línea de puntos y el primer punto de la siguiente línea.

Las variables ‘anterior’ y ‘posterior’ son las encargadas de almacenar cada uno de los valores de las coordenadas X de dos filas contiguas en la matriz original. Siempre que el valor de la variable ‘posterior’+10 sea menor que el valor contenido en la variable ‘anterior’ existirá un cambio de línea. En todas las líneas anteriores a la que se está trabajando se les añadirá el mismo ruido en cada una de las coordenadas de todos los puntos.

```
21.0439 ; 0.399948 ; -6.10352e-005 ; 792.535 ; 512 ;
21.1439 ; 0.399948 ; -6.10352e-005 ; 793.26 ; 512 ;
21.2439 ; 0.399948 ; -3.05176e-005 ; 793.985 ; 512 ;
21.3439 ; 0.399933 ; 0 ; 794.71 ; 512 ;
21.4439 ; 0.399933 ; 0 ; 795.435 ; 512 ;
21.5439 ; 0.399933 ; 0 ; 796.16 ; 512 ;
21.6439 ; 0.399933 ; 0 ; 796.884 ; 512 ;
-4.35617 ; 0.999931 ; 0 ; 608.425 ; 512 ;
-4.25617 ; 0.999931 ; 0 ; 609.149 ; 512 ;
-4.15617 ; 0.999931 ; 0 ; 609.874 ; 512 ;
-4.05617 ; 0.999931 ; 0 ; 610.599 ; 512 ;
-3.95617 ; 0.999931 ; 0 ; 611.324 ; 512 ;
-3.85617 ; 0.999939 ; -3.05176e-005 ; 612.049 ; 512 ;
-3.75617 ; 0.999916 ; 3.05176e-005 ; 612.774 ; 512 ;
-3.65617 ; 0.999916 ; 3.05176e-005 ; 613.498 ; 512 ;
-3.55617 ; 0.999916 ; 3.05176e-005 ; 614.223 ; 512 ;
-3.45617 ; 0.999916 ; 3.05176e-005 ; 614.948 ; 512 ;
```

Ilustración 12. Imagen en la que se aprecia un cambio de línea en un barrido realizado con un paso de 0,6 mnn.,

Como se aprecia en la imagen, tal cual aparece en el archivo generado por la simulación, en la primera columna, que se corresponde con la coordenada X de los puntos, hay un salto en el valor, que indica un cambio

de línea. También se aprecia un salto en el valor de las coordenadas Y (segunda columna) cuya valor es de 0,6 puesto que corresponde al archivo generado por una simulación realizada con un paso de 0,6 mm.

Pues bien, lo que el cuerpo del programa se encarga de realizar es, que una vez llegado a este punto en el que las variables tienen los siguientes valores:

‘anterior’ = 21.6439

‘posterior’ = -4.35617

Puesto que se cumple que: ‘posterior’ + 10 < ’anterior’, a los valores que se encuentran hasta la fila del valor ‘anterior’ se les asigna un mismo ruido (esta operación se realiza fila a fila, en el momento que se detecta un cambio de fila, los valores de ‘ruido’ que se asignan varían).

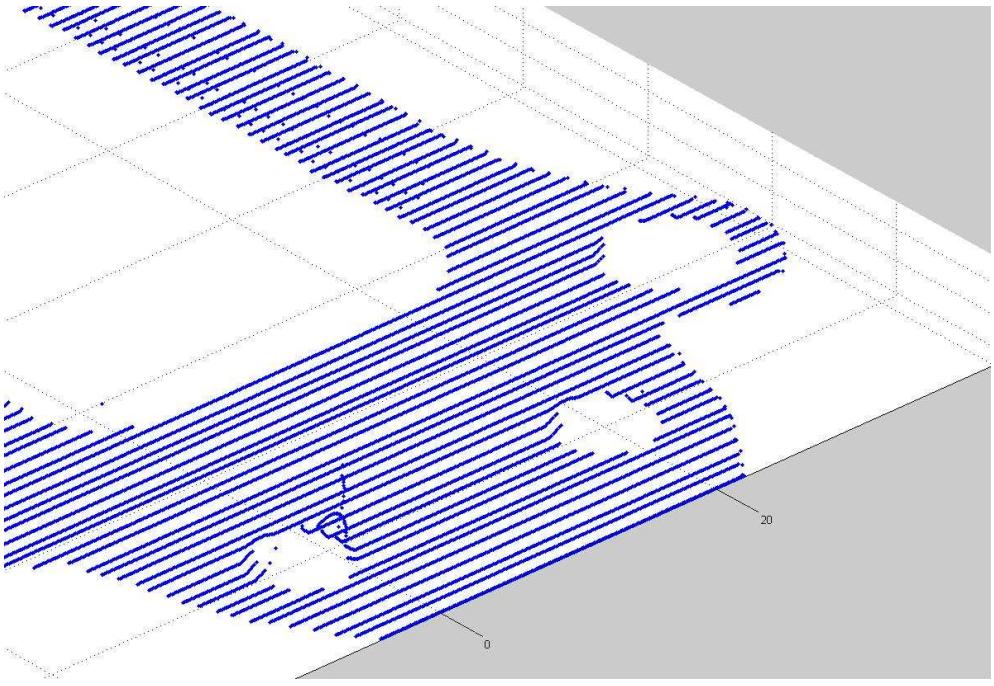

Ilustración 13. Representación del ruido añadido a cada línea de puntos.

Se aprecia los puntos forman parte de una línea de puntos pero en la que cada línea dista de la anterior y la posterior una distancia distinta.

Esta forma de representación simula la posibilidad de que mientras el carro se desplaza y el sistema recoge los datos correspondientes, sufra pequeñas paradas y/o arrancadas. Estos enganchones, al correr por una cremallera son muy posibles.

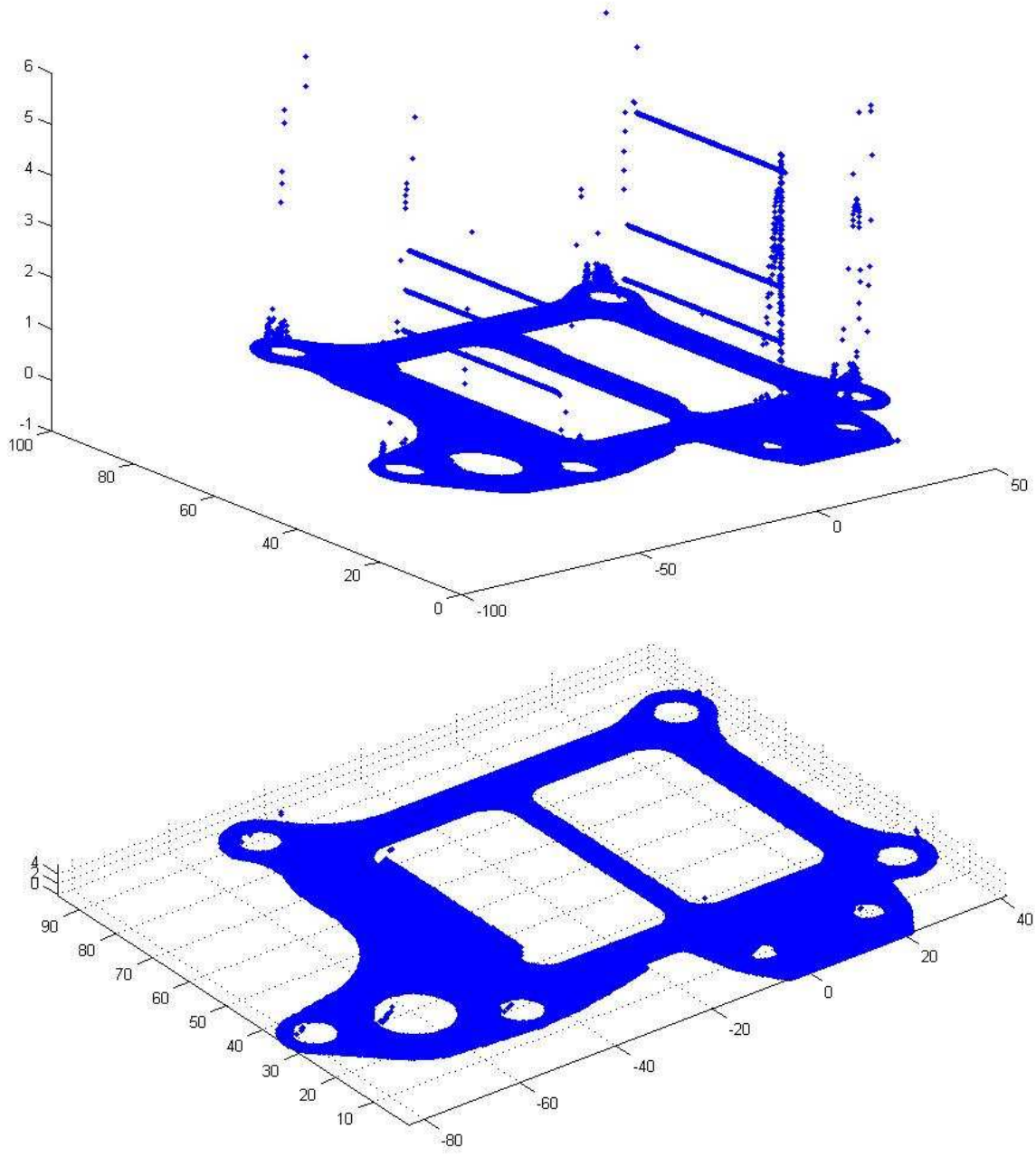

Ilustración 14. Comparativa de los 2 ruidos añadidos (Arriba ruido en UV y abajo en XYZ)

- CÁLCULO DE LA VARIANZA DE LA DISTRIBUCIÓN UTILIZADA CORRESPONDIENTE PARA CADA UNO DE LAS COORDENADAS

Como se ha explicado anteriormente, la varianza utilizada para la distribución normal será la que determine los valores de ruido que aparecerán y el porcentaje de puntos que recibirán un ruido más elevado de lo normal o de lo esperado.

A mayor varianza, mayor es la probabilidad que tengan los valores generados de ser mayores de los límites que se habían marcado. Cabe decir, que la probabilidad la podemos marcar nosotros, según queramos más o menos ruido.

Coordenadas de posición XYZ.

El ruido generado aleatoriamente ha de estar comprendido entre los valores 0,2 mm y -0,2 mm. Como se ha comentado anteriormente, los valores creados han de seguir una distribución normal cuya media es 0 y su desviación típica desconocida.

Sabiendo que la curva de una distribución normal es asintótica al eje de abcisas, todos los valores comprendidos entre $+\infty$ y $-\infty$ son teóricamente posibles, ningún valor tiene probabilidad 0. Debemos hallar el valor de la desviación típica que le otorgue a los valores mayores de 0,2 y menores de -0,2 una probabilidad prácticamente nula. En este caso calcularemos la varianza para que la probabilidad de que los valores aleatorios que aparezcan tengan una probabilidad de encontrarse en el rango establecido del 99,9%.

Imponiendo el condicionante de la probabilidad del 99,9% comenzamos los cálculos, siendo lo siguiente lo que queremos calcular:

P (K1 ≤ Z ≤ K2)

Para una distribución normal, las cuentas son las siguientes:

Sabiendo que: P(X ≤ x) = P (Z ≤ z) =

$$P(-0,2 \le X \le 0,2) = P(X \le 0,2) - P(X \le -0,2) = (\frac{-0,2-0}{\sigma} \le Z \le \frac{0,2-0}{\sigma}) = 0,999$$

$$P(X \le -0,2) = P(X \ge 0,2) = 1 - P(X \le -0,2)$$

$$P(Z \le \frac{0,2}{\sigma} - 1 + P(Z \le \frac{0,2}{\sigma}) = 0,999$$

$$\frac{1,999}{2} = P(Z \le \frac{0,2}{\sigma}) = 0,9995$$

El valor de Z se toma de las tablas normalizadas de la distribución normal. Hay que buscar el valor que tenga una probabilidad del 0,9995.

El valor recogido de la tabla es 3,27.

$$P(3,27 \le \frac{0,2}{\sigma}) = 0,995 \Rightarrow 3,27 = \frac{0,2}{\sigma} \Rightarrow \sigma = 0,06116207951$$

El valor de la varianza para las coordenadas XYZ debe ser **0,06116207951**

Coordenadas de pantalla.

En este caso, los valores que componen la matriz ruido deben estar contenidos entre -3 y 3 píxeles. Como anteriormente estos valores deben seguir una distribución normal cuya media es 0, por los mismos motivos explicados anteriormente. Los valores que no se encuentren en el rango anterior deben tener una probabilidad prácticamente nula de aparecer en la matriz. Al igual que en el caso anterior se establece una probabilidad de aparición de valores contenidos en el rango del 99,9%.

Calculando de manera análoga a la anterior y sabiendo que el valor Z es 3,27 podemos calcular la varianza para este caso:

$$P(3{,}27 \leq \frac{3}{\sigma}) = 0{,}995 \Rightarrow 3{,}27 = \frac{3}{\sigma} \Rightarrow \sigma = 0{,}91743119266$$

En este caso, el valor de la varianza es **0,91743119266.**

Como se puede apreciar, comparando los valores obtenidos, la varianza en el segundo caso es mucho mayor que en el primero, puesto que el rango en el segundo es mucho más amplio.

A continuación, una muestra en imágenes de las diferencias que supone la varianza a la hora de realizar las representaciones, la primera con varianza 0,01 ; la segunda con varianza 0,1 y la última con varianza 0,9. En la primera el ruido introducido es casi nulo, en la segunda se muestra una representación de un ruido acercado a la realidad, y en la última un ruido exagerado.

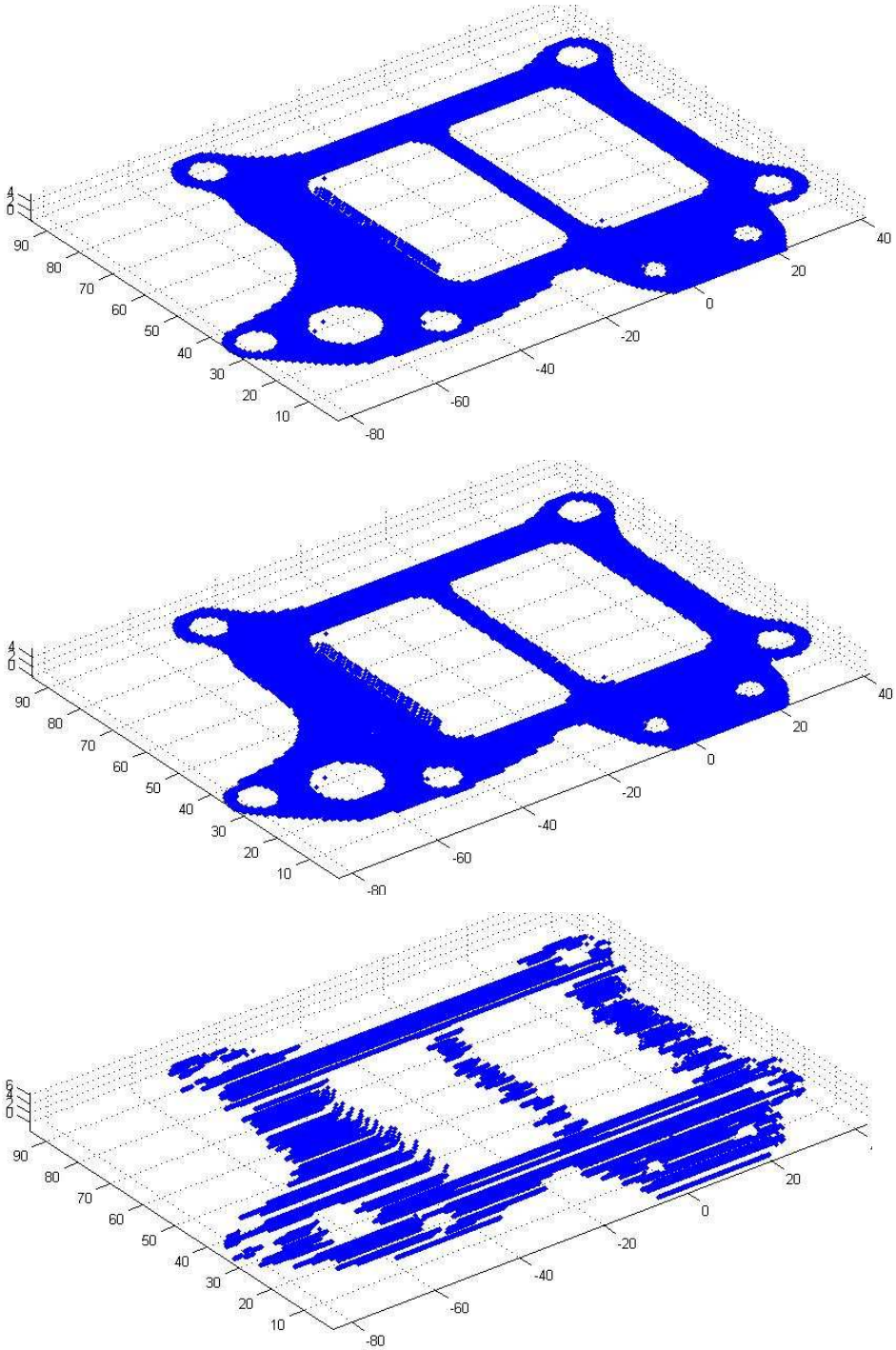

Ilustración 15. Desde arriba, representación con ruido nulo, la segunda con ruido próximo al real y la tercera con ruido exagerado.

- CALIBRACIÓN DE LAS CÁMARAS.

Para comprobar el funcionamiento de las cámaras y ver si este es correcto hay que realizar una calibración de la cámara. Esta función la realizaremos con una interfaz creada mediante un código de MATLAB.

Esta aplicación realiza el calibrado de la cámara mediante la toma de datos, realizando una simulación similar a la que se realiza la aplicación DirectX que recoge las coordenadas mediante barridos.

La aplicación que realiza el calibrado de las cámaras se llama Metrovisionlab.

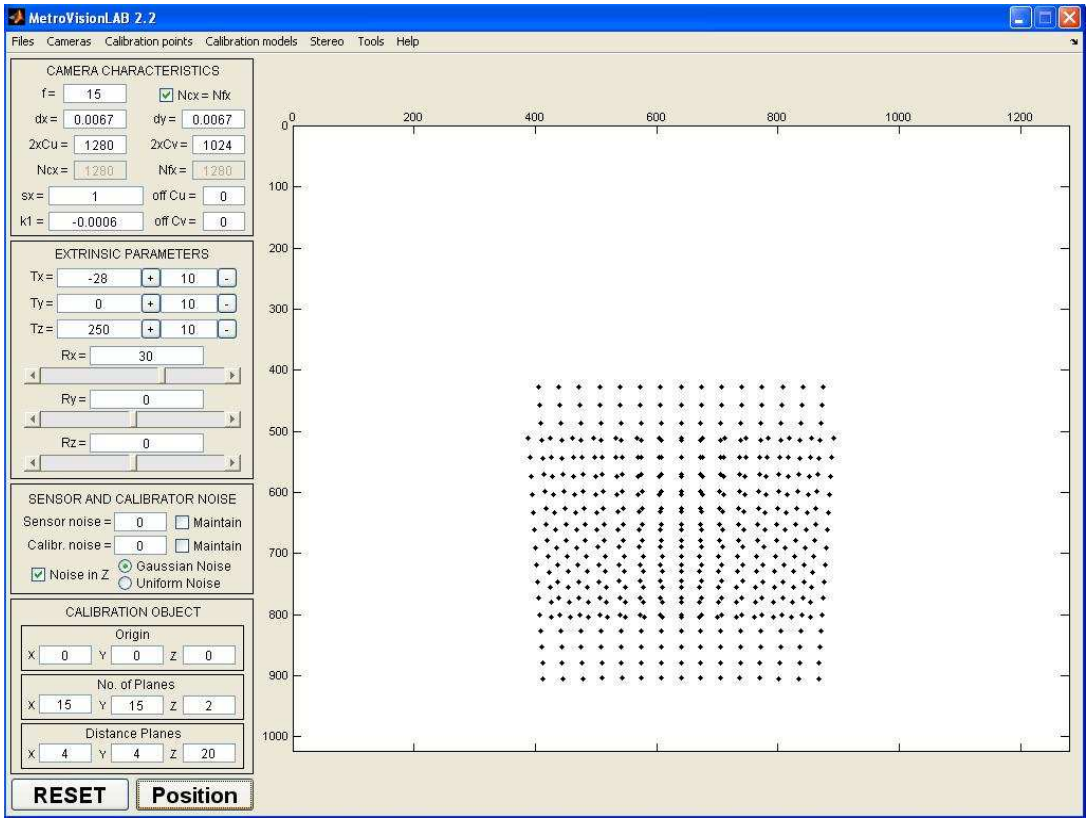

Ilustración 16. Esta es la imagen que aparece cuando se carga la aplicación Metrovisionlab en MATLAB.

Para realizar la calibración de la cámara, la aplicación crea puntos, en la disposición que se desea. Estos serán los puntos de los que la cámara y durante la simulación

Para realizar la calibración de la cámara se crean el programa genera puntos repartidas en 2 planos con distinta coordenada Z. Cada plano de puntos estará formado por 225 formando 15 filas y 15 columnas.

Los parámetros de calibración se corresponden a los utilizados para algunas de las simulaciones anteriores, y tales como aparecen en la Ilustración 7. Los parámetros introducidos dan lugar a la siguiente situación.

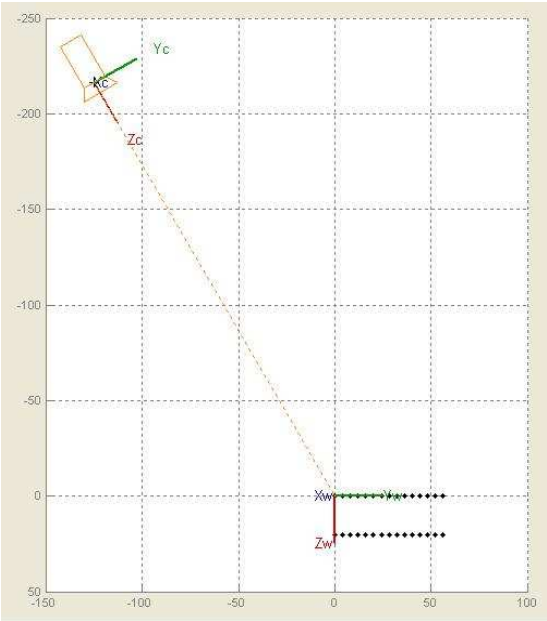

Ilustración 17. Situación en la que tiene lugar la calibración de la cámara.

- GENERACIÓN DE UN CÓDIGO PARA EL CAMBIO DE COORDENADAS DE UV A XYZ.

Para comprobar si la calibración y el código del mismo son correctos, procedemos a realizar una simple prueba que trata de cambiar las coordenadas UV a coordenadas XYZ.

Para esta tarea hubo que adaptar un código de cambio de coordenadas existente a la forma de trabajar de MATLAB y que llevase a cabo correctamente la función que requiere.

El código es el siguiente:

```
[filas,columnas]=size(archivo);
resultado=archivo;
contador=1;
posicion=2;
anterior=archivo(1,1);
posterior=archivo(posicion,1);
paso=0;

 for i=1:filas;
       a11=MCam(1,1)-(archivo(i,4)*MCam(3,1));
       a12=MCam(1,2)-(archivo(i,4)*MCam(3,2));
       a13=MCam(1,3)-(archivo(i,4)*MCam(3,3));
       b1=(archivo(i,4) * MCam(3,4))-MCam(1,4);

       a21=MCam(2,1)-(archivo(i,5)*MCam(3,1));
       a22=MCam(2,2)-(archivo(i,5)*MCam(3,2));
       a23=MCam(2,3)-(archivo(i,5)*MCam(3,3));
       b2=(archivo(i,5) * MCam(3,4))-MCam(2,4);

                a31=cA;
                a32=cB;
                a33=cC;
```

```
                b3=cD;

determinante=((a11*a22*a33)+(a21*a32*a13)+(a12*a23*a31)-
(a31*a22*a13)-(a32*a23*a11)-(a21*a12*a33));

       Xw=((b1*a22*a33)+(b2*a32*a13)+(a12*a23*b3)-
(b3*a22*a13)-(a32*a23*b1)-(b2*a12*a33))/determinante;
       Yw=((a11*b2*a33)+(a21*b3*a13)+(b1*a23*a31)-
(a31*b2*a13)-(b3*a23*a11)-(a21*b1*a33))/determinante;
       Zw=((a11*a22*b3)+(a21*a32*b1)+(a12*b2*a31)-
(a31*a22*b1)-(a32*b2*a11)-(a21*a12*b3))/determinante;
                resultado(i,1)=Xw;
                resultado(i,2)=Yw;
                resultado(i,3)=Zw;
 end;

 for i=1:(filas-2)
     if (posterior<anterior);
         contador=contador+1;
         paso=(paso+0.2);
     end;
     resultado(i,2)= (resultado(posicion,2)+paso);
     posicion=posicion+1;
     anterior=posterior;
     posterior=resultado(posicion,2);
 end;
plot3(resultado(:,1),resultado(:,2),resultado(:,3),'b.');
```

En el código, la llamada al archivo de extensión .txt llamado ‘archivo’ recoge las coordenadas de todos los puntos con los que se realizan las calibraciones (recordar que estos puntos pueden ser más o menos según nos interese). Aparece un nuevo archivo, archivo MCam. Es un archivo de extensión .txt que contiene los datos de la cámara con la que se realizan las simulaciones.

Las variables cA, cB, cC y cD recogen parámetros del plano en el que están contenidos los puntos analizados y el punto de partida de la cámara (un plano queda determinado por 3 puntos.

Un par de ejemplos con los puntos de calibración, en primer lugar con el láser totalmente vertical y las primeras líneas con Z = 0 de los puntos de calibración.

Al realizar el calibrado de la cámara y sacar los datos de los puntos de calibración utilizados, haciendo una prueba con las 2 filas (superior e inferior) que tienen su coordenada Y en 0, para saber si funcionar correctamente el código que cambia las coordenadas U y V de los puntos a coordenadas XYZ,

- CÁLCULO DE LOS PLANOS QUE CONTIENEN LOS PUNTOS

Para realizar el cambio de coordinadas UV a coordenadas XYZ es necesario saber el plano que forman los puntos que se analizan y el punto en el que la cámara está situada.

Con los parámetros que se muestran a continuación, con los que se han realizado las simulaciones, en la aplicación MetroVisionLAB 2.2 se pueden obtener los valores de la matriz MCam a introducir en el código creado en Matlab para realizar el cambio de coordenadas UV a XYZ.

2201.0171	306.1239	562.1653	159258.2723
-19.1419	2171.9698	-663.3	127415.7315
-0.035113	0.4793	0.87695	248.8267

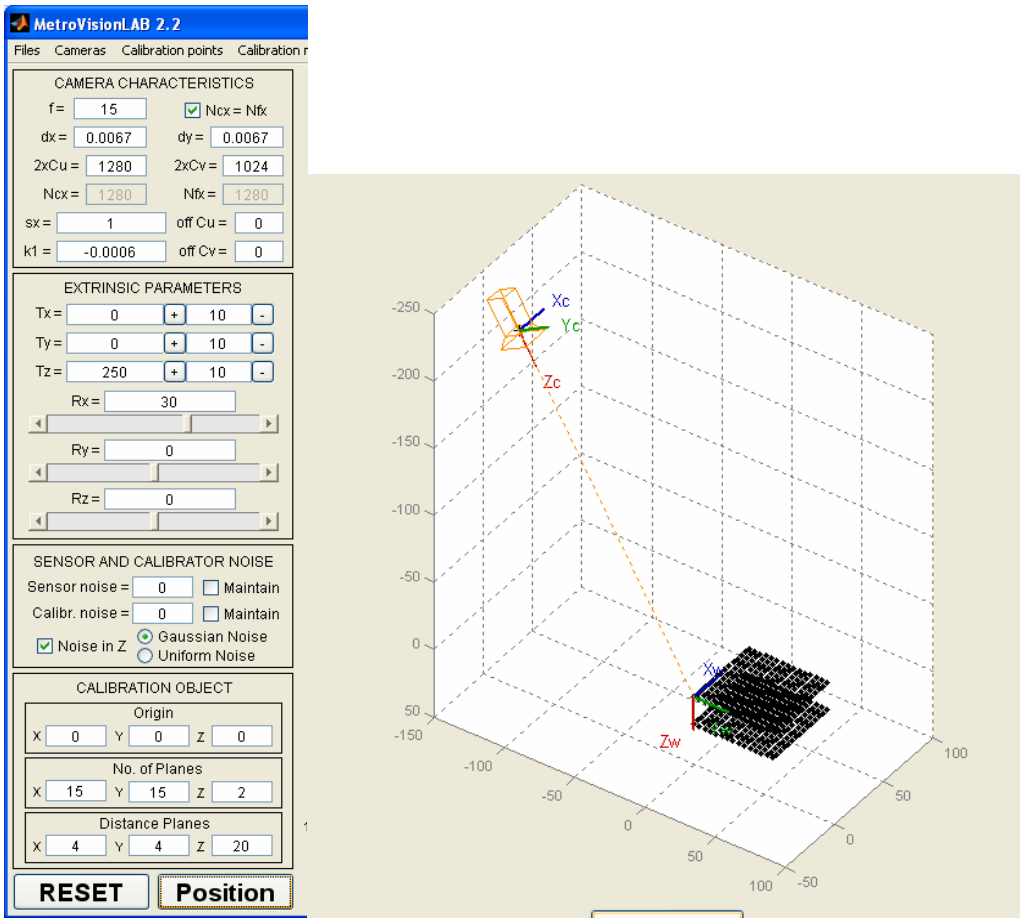

Ilustración 18. Imagen de los parámetros utilizados para el calibrado de la cámara (izda.) y la situación a la que dan lugar dichos parámetros (derecha).

Con la cámara formando un ángulo de 70º con la horizontal y a 250 mm. De distancia y el láser en un ángulo de 20º con la horizontal y a 250 mm de distancia.

Para calcular los parámetros cA, cB, cC y cD del plano he cogido los 3 puntos que se muestran en la siguiente imagen, siendo el punto número 1 la posición del láser que se puede determinar con los datos nombrados anteriormente (250 mm de distancia y 20º con la horizontal). Las coordenadas de los puntos 2 y 3 se pueden extraer de la aplicación Metrovisionlab, que al realizar el calibrado, los puntos que genera para esto, son puntos que la aplicación da a conocer.

En este caso, he cogido los puntos extremos del plano superior de puntos, cuyas coordenadas Y y Z son 0. Las coordenadas son:

P1 (0 , 234.923 , -85.505)

P2 (0 , 0 , 0)

P3 (60 , 0 , 0)

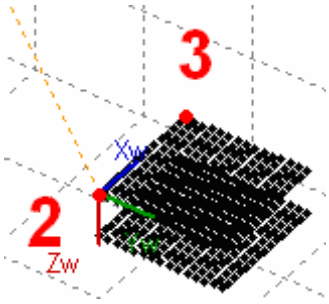

Ilustración 19. Representación de los puntos elegidos para el cálculo del plano.

Conocidos estos 3 puntos se puede realizar el cálculo del plano y determinar los valores de los parámetros cA, cB, cC y cD. Tras los cálculos, el plano resultante es:

$$234{,}923x - 5.130{,}3y - 14.095{,}389z = 0$$

Los parámetros a introducir en el código de Matlab, que realice el cambio de coordenadas, en archivos con datos recogidos de simulaciones con las condiciones establecidas más arriba serán las siguientes:

cA = 234,923

cB = 5.130,3

cC = 14.095,389

cD = 0

En la representación que se muestra a continuación se puede ver una línea de puntos, perteneciente a la base de la pieza XS2002, con la que se han hecho las primeras pruebas de cada una de las tareas. Se aprecia que la forma lineal la mantiene como en la original, la inclinación que posee dicha línea sobre uno de los ejes se debe a que las coordenadas correspondientes tienen una pequeña desviación respecto de las originales a la hora de realizar los cálculos. Estas diferencias que aparecen son debidas a la acumulación de errores durante los cálculos por los redondeos que se llevan a cabo durante las simulaciones.

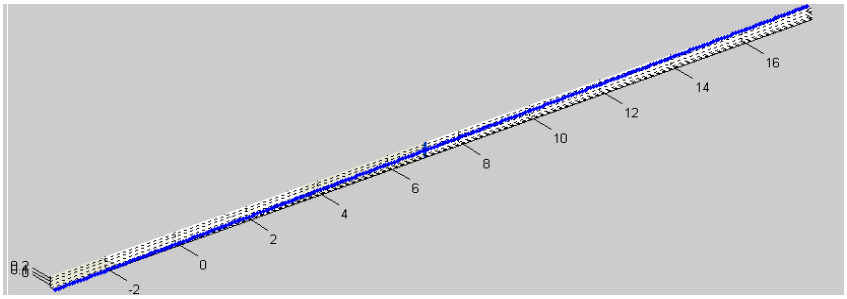

Ilustración 20. Imagen de una de las líneas de puntos con coordenadas cambiadas.

4 DESVIACIONES Y DIFICULTADES RESPECTO A LA PLANIFICACIÓN

A lo largo del periodo en el que se ha trabajado en el proyecto, han aparecido numerosas dificultades y problemas que han retrasado y dificultado las tareas que he llevado a cabo.

4.1 PROBLEMAS CON LAS PIEZAS.

Las piezas fueron durante el comienzo un problema, ya que, las piezas de que se disponían estaban creadas por partes y ensambladas en archivos de extensión .par.

Los archivos de extensión .par, con los que trabajamos, son conjuntos de piezas ensambladas mediante relaciones de coincidencia, en la que la primera pieza cargada, y según la posición en que se sitúa, es la que determina la posición del punto de origen de la pieza.

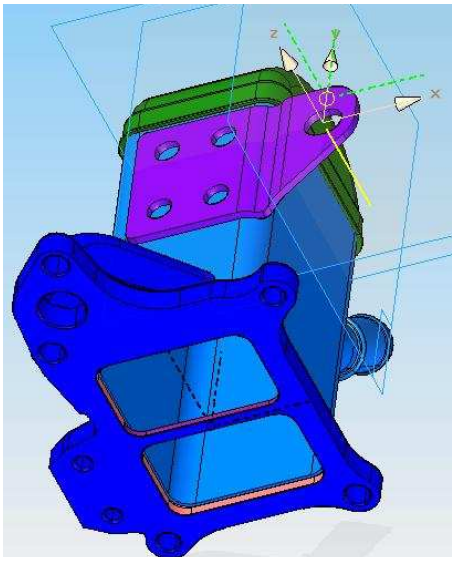

Ilustración 21. Pieza XS2002 con el origen situado en la pata.

A la hora de realizar las simulaciones para los barridos, la pieza se sitúa según esté posicionado el origen de coordenadas y el sentido que lleven estas direcciones. Para cada barrido distinto (cada posición de la pieza requería un barrido distinto, en la simulación) había que cambiar la posición del origen de coordenadas de la pieza y posicionarla en un punto correspondiente a la cara de la pieza que se quiere analizar.

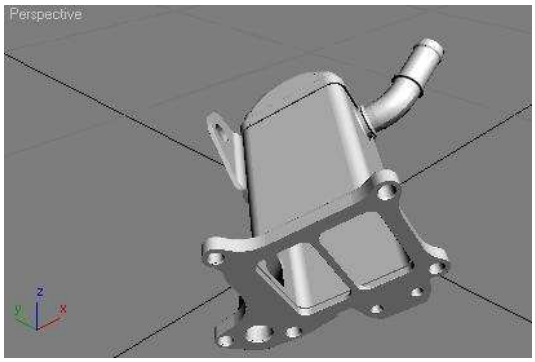

Ilustración 22. Pieza situada con un origen de coordenadas aleatorio.

Este cambio de posición del eje de coordenadas no se puede realizar con el programa de las piezas de origen (Solid Edge), por lo que se requería el manejo dejo del programa Autodesk Studio 3DS Max 9.

Mediante la exportación de las piezas desde Solid Edge a Studio 3DS Max se solucionó el problema, ya que, 3DS Max trata las piezas como un único sólido, por lo que el centro de coordenadas de la pieza se puede cambiar y posicionar en un punto que nos interese.

4.2 EL COMPORTAMIENTO DE MATLAB NO ERA EL ESPERADO

En varios de los códigos generados mediante Matlab, la simulación se realizaba correctamente realizando las pruebas con una de las piezas modelo.

Al realizar simulaciones con otras piezas o esa misma pieza en una posición distinta a la primera, la simulación no se realizaba correctamente. Tras varios días buscando la fuente del problema se llegó a determinar que el problema es generado por el código de la aplicación DirectX que realiza la recogida de los puntos de la figura.

La pieza, que se carga en formato .X en la aplicación, la introduce en un prisma cuyas dimensiones son las máximas correspondientes a la figura.

4.3 PROBLEMAS AL REALIZAR BARRIDOS

La aplicación DirectX recoge las piezas en formato .X, estas piezas, la aplicación las centra en un prisma cuyas dimensiones máximas son las máximas de la pieza.

El barrido se realiza en la cara en la que se sitúa el origen de coordenadas, que coincide con una de las caras que conforman el prisma. Luego, todas aquellas partes de la pieza que no pertenezcan, o no estén situados muy próximos, a puntos de la superficie del prisma, no son recogidas por el barrido.

Este problema supuso tener que realizar simulaciones de ciertas posiciones de las piezas eliminando partes de esta pieza. Este problema se planteaba principalmente a la hora de realizar el barrido en las partes a las que llamamos “patas”. En todas las piezas se solucionó dicho problema eliminando la base de la pieza.

Con la solución de este problema, se descubrió un error. Uno de los tubos de las piezas, no estaba alineado con el agujero que debe alojar dicho tubo en el cuerpo del intercambiador de calor, las coordenadas de los puntos del agujero no correspondían

con las coordenadas del extremo del tubo correspondiente a la pieza, como ocurría en el resto de las piezas.

4.4 TIEMPO NECESARIO PARA REALIZACIÓN DE SIMULACIONES

Un problema que se planteó durante la realización de las simulaciones es el tiempo que estas duraban. Realizando simulaciones en las que la toma de datos no era muy detallada, el tiempo oscilaba entre los 10 y los 15 minutos, pero conforme el paso con el que se realizaban dichas simulaciones iba disminuyendo, recordar que las primeras simulaciones se realizaron con pasos desde 1 mm hasta 0,01 mm, el tiempo que costaba realizar la simulación aumentaba. La última simulación, de paso 0,01 mm que se realizó en el ordenador propio tardó aproximadamente cinco horas y media en completarse.

El coste de tiempo de las simulaciones está directamente influido por la capacidad del equipo informático que las soporta. Por lo que para realizar simulaciones muy detalladas se hace necesario llevarlas a cabo con equipos potentes.

Como se nombra en uno de los puntos posteriores, la solución a este problema entra dentro de las acciones a realizar en el futuro, para el perfeccionamiento del sistema.

5 DEDICACIÓN

La dedicación al proyecto ha sido discontinua desde el principio, bien por algún que otro problema de salud, que cortó el comienzo del proyecto durante algo más de un mes, o bien por hechos, como encontrar trabajo, que impidieron durante varios meses dar continuidad al trabajo en el proyecto.

El trabajo más intenso y fructífero fue el llevado a cabo durante los meses de Octubre hasta comienzos de Noviembre, el mes de Enero y hasta mediados de Febrero, cuando coincidió el comienzo en el trabajo con el comienzo de un curso para personas apuntadas al paro y se volvió a interrumpir el trabajo en el proyecto. Durante los últimos meses, el trabajo ha sido intermitente.

6 HABILIDADES Y CONOCIMIENTOS ADQUIRIDOS

La realización del proyecto aquí detallado me ha permitido adquirir nuevos conocimientos y perfeccionar o mejorar otros que ya estaban ahí.

En primer lugar, me ha otorgado la oportunidad de no olvidar los sistemas de programación, tan laboriosos y a veces tan complicados, siendo además en este caso, motivo para el aprendizaje de un nuevo programa para mi como es Matlab y que tan útil puede resultar su uso incluso en la vida cotidiana. Este tema me gustaría seguir trabajándolo y estudiándolo con el fin de llegar a manejarlo con cierta soltura y poder recurrir a él cuando sea necesario.

Por otro lado, he aprendido muchas cosas del software Solid Edge de diseño CAD parametrizado, del cual conocía la forma de trabajar, porque la había estudiado de forma autodidacta. También he de decir, que durante la realización de este

proyecto, realicé un curso con otro programa de características similares a este, cuya forma de trabajar es similar. Considero que este tipo de programas de CAD parametrizado son importantes tenerlos en cuenta siendo ingeniero, ya que la versatilidad que ofrecen frente a cambios o problemas surgidos durante el diseño de la pieza que se requiera es muy superior a la que requieren las aplicaciones de dibujo habituales hasta ahora, en las que cambiar una sola dimensión de alguna pieza podía acarrear horas de trabajo. Además de que estos programas permiten realizar muchas más operaciones, la mayoría de ellas las realiza automáticamente suponiendo un ahorro de tiempo enorme.

7 CONCLUSIONES Y LÍNEAS FUTURAS

7.1 CONCLUSIONES DEL TRABAJO

Las simulaciones llevadas a cabo durante el proyecto y su análisis son correctos en su mayoría, salvo en alguna situación en la que el ruido añadido en alguno de los puntos ha dado la casualidad de tener un valor exterior al rango establecido en el código de ruido generado.

Viendo el funcionamiento del sistema, y los buenos resultados que han arrojado las simulaciones tanto con ruido como sin él, el sistema de triangulación láser trabajando en su máxima capacidad, y con servidores más potentes que permitan aprovechar el máximo de la capacidad de trabajo del sistema, puede suponer un ahorro de tiempo en el control de calidad al final de la línea de fabricación muy notable.

Además del ahorro de tiempo que supone, el realizar un examen de control al 100% de la producción dará lugar a una calidad excelente de los productos ya montados y finalizados en los que se vayan a disponer, siendo aumentada la fiabilidad de dichos productos.

Una mejor calidad de producto, que de menos problemas, también es un indicativo de diferenciación respecto de la competencia, y este sistema puede dotar al producto de la seguridad de estar en buen estado.

7.2 CONCLUSIONES PERSONALES

A nivel personal, la realización del proyecto me ha dado una visión más cercana del trabajo que supone y el esfuerzo que es necesario invertir en hacer funcionar un proyecto tan importante y de la envergadura del proyecto madre de este, siendo que hasta ahora, no había tenido una visión tan cercana a un proyecto.

Como se nombra anteriormente, el proyecto me ha otorgado conocimientos que hasta ahora me eran muy distantes o no conocía. Me ha acercado al mundo de la robótica, el cual desconocía completamente y me ha permitido recordar el funcionamiento de los lenguajes de programación.

A nivel personal, estoy satisfecho con las tareas realizadas sobre el proyecto, el cual me parece más que interesante y el cual tiene un largo recorrido por delante hasta su perfeccionamiento

7.3 LÍNEAS FUTURAS

Uno de los puntos que pueden resultar más interesantes de cara al futuro tiene que ver con el tratamiento de los datos generados. Actualmente para cada operación es necesario cargar, siempre en nuevas ventanas, los archivos que se generan anteriormente. Además, esto obliga a grabar todos los

archivos intermedios que se generan, ya que, han de estar disponible para cada una de las tareas que se quieran realizar.

Por otro lado, en lo que concierne al tratamiento de datos, a buen seguro existen lenguajes de programación que realizan las tareas para las que se les programe de manera más rápida y con mayor exactitud.

Otro aspecto importante a mejorar es el concerniente a la realización de simulaciones de barrido. Como se ha comentado en puntos anteriores, la aplicación DirectX realiza las simulaciones, encerrando la pieza en un prisma cuyas dimensiones son las máximas de la pieza sobre la que se está simulando. Cuando se realiza la simulación, la aplicación recorre una de las caras del citado prisma, siendo siempre la cara que contiene el origen de coordenadas, lo que impide realizar simulaciones sobre partes de las piezas que no se encuentren en dichas caras o muy próximas a ellas.

En lo referente a la realización de las simulaciones se aprecia que según la posición de las piezas sobre las que se realizan las simulaciones, estas realizan la simulación gran parte del tiempo fuera de la pieza, de manera que, la pérdida de tiempo es considerable, como dato, en alguna de las simulaciones, como el barrido sobre la boca del tubo de las piezas, más del 60% del tiempo que cuesta en realizarse la simulación, se realiza sobre vacío, es decir, que no recoge datos acerca de nada.

Otro punto a tener muy en cuenta tiene que ver con el posicionamiento, de las piezas con las que se va a trabajar durante las simulaciones. Actualmente se debe tener un archivo diferente para cada posición, para cada una de las piezas, siendo estos archivos exportados desde programas externos. En nuestro caso, hay que exportar un mismo archivo hasta 2 veces para poder realizar la simulación de manera correcta.

Teniendo la posibilidad de introducir la posición de la pieza en el mismo simulador se ahorraría mucho tiempo y muchos archivos, cada archivo que se exporta a otro es de extensión distinta al anterior.

Realizando una aplicación con un interfaz en el que el usuario pueda realizar todas estas propuestas de una manera sencilla y rápida, el ahorro en tiempo, en liberación de disco y en efectividad sería mucho mayor. Poder elegir el archivo a tratar, elegir la posición de la pieza sobre la que simular, y poder elegir que archivos guardar o visionar dará mucha más versatilidad al proyecto, haciéndolo más fácil y útil.

8 AGRADECIMIENTOS

El primer agradecimiento va dirigido a Don Carlos E. Cajal, director del proyecto, y que me dio la oportunidad de trabajar con él y tutelarme durante la tarea de realizar este proyecto. Además de introducirme en el mundo de la fabricación y de la calidad a lo largo de la carrera, en la cual ha sido un profesor ejemplar.

El segundo agradecimiento es para mis padres y familiares y por supuesto mi novia, por darme ánimos cuando las cosas no iban del todo bien y ponerme las pilas cuando era yo quien hacía que las cosas no fuesen tan bien como deberían.

Y el tercero, a todos los compañeros con los que he tenido el gusto de trabajar y estudiar, con los que he hecho una más que buena amistad. Gracias a aquellos que siempre han intentado elevar los ánimos bajos. Gracias a los que han demostrado que la vida puede dar mil vueltas de tuerca y plantearte y situarte en las situaciones menos esperadas en muy poco tiempo.