Universidad de Guanajuato
Departamento de Matemáticas
Métodos Numéricos
Dr. Joaquín Peña Acevedo

Reporte de proyecto final
Sistema de seguridad

Diego Aarón Moreno Galván
Licenciatura en Matemáticas
5to. Semestre

Índice

Introducción.

Motivación del proyecto.

En estos días la delincuencia ha aumentado mucho en las ciudades, más porque hay cada vez menos educación en México, y menos empleo. Por eso la necesidad de proteger tus bienes y más que nada en estas temporadas de vacaciones, en las cuales salimos de nuestras casas y necesitamos cuidar nuestro hogar. De aquí surge la idea realizar un sistema de seguridad el cual se encargue de cuidar nuestro hogar cuando no estemos.

El sistema de seguridad consiste en dos cámaras ubicadas en algún lugar de tu casa. Éstas detectarán si ocurre algún movimiento en la zona, ubicarán al ladrón en el espacio tridimensional, analizarán la trayectoria que el ladrón realiza y va a predecir la posición del ladrón para lanzar un dardo automáticamente y dormir al ladrón; después se podría llamar a la policía si se activa el lanzamiento del dardo.

Enfoque del proyecto.

Por el momento, solo nos daremos a la tarea de realizar los aspectos del proyecto que tenga que ver con los métodos númericos, pues la otra parte importante que no tiene que ver con la clase es detectar el movimiento a partir de imágenes y reconocer objetos.

Para esto, consideraremos que el ladrón es una masa puntual (podemos pensar que le apuntamos al centro de masa del ladrón que es como por el abdomen), generaremos un trayectoria tridimensional la cual representa el movimiento del ladrón respecto al tiempo (posición del ladrón en un instante), proyectaremos la trayectoria a imágenes 2D que son la imágenes que perciben las cámaras y a partir de éstas reconstruiremos la trayectoria, verificaremos el error que tenemos al reconstruir y con esto, vamos a predecir la posición que tendrá el ladrón extrapolando la trayectoria y así saber dónde disparar el dardo. Podemos probar con diferentes trayectorias y calcular el error que tenemos al predecir la posición que tiene el ladrón.

Inicialización del espacio de trabajo.

Supongamos que nuestro patio (o donde coloquemos nuestras cámaras) es rectangular de dimensiones **N**x**M** metros, las cámaras las colocaremos en dos esquinas consecutivas del lugar a **h** metros de altura apuntando las dos cámaras al centro del patio (o a algún lugar del patio en el cual tengamos la mejor visión de lo que pasa). Para el ejemplo que explicaremos en este reporte, colocamos las cámaras de manera que una esquina estuviera en el origen y la otra

cámara en y = 0 y x = N (las dos cámaras a altura de h metro). Podemos visualizar un ejemplo del espacio de trabajo en la figura 1.

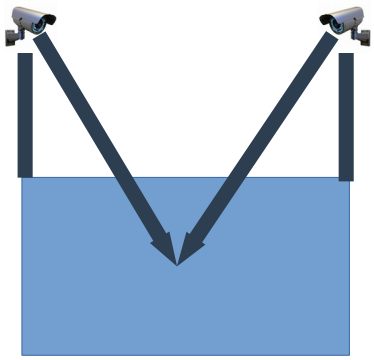


Figura 1.

Proyección a cámara.

Conociendo cómo trabaja una cámara.

Una cámara cuenta con un ojo o foco donde se va a recibir los rayos de luz de los objetos y un plano proyectivo en el cual se posicionará la imagen de lo que ve el foco. Ver figura 2. Recordemos que el plano proyectivo de la cámara es tal que está dividido en pixeles, cada pixel simula ser un punto en el plano; por esta razón, estamos limitados físicamente a proyectar exactamente la posición de cada punto en el espacio. Supongamos que tenemos una cámara que cuenta con un plano de pixeles de **NP**x**MP**, donde físicamente cada pixel es un cuadrito de lado $\boldsymbol{\varepsilon}$, ver figura 3; de esta manera podemos hacer una discretización del plano proyectivo dado por: $x_i = i\varepsilon$, para i = 0, 1, …, NP. Y en el caso vertical con: $y_i = i\varepsilon$, para i = 0, 1, …, MP.

Además, debemos tomar en cuenta la distancia **f** que hay entre nuestro ojo o foco de la cámara y el plano proyectivo; podemos observarlo también en la figura 3. Ésta será de gran utilidad para identificar la amplitud de visión que vamos a tener, pues entre más cerca esté el plano de proyección al foco tendremos más área de visión y entre más alejado esté el plano, tendremos menos área de visión; sin embargo, como ya antes se mencionó, estamos limitados físicamente a tener NPxMP pixeles con área de percepción de ε^2, así que la calidad (o definición) de la imagen que obtendremos será más baja si el plano está más cerca del foco pues muchos rayos de luz (cercanos entre sí) que percibe el ojo serán proyectados en el mismo pixel, lo cual disminuye la precisión (o calidad) en la imagen. Por el contrario, si el plano está más alejado del foco se obtendrá más definición en la imagen por el mismo principio.

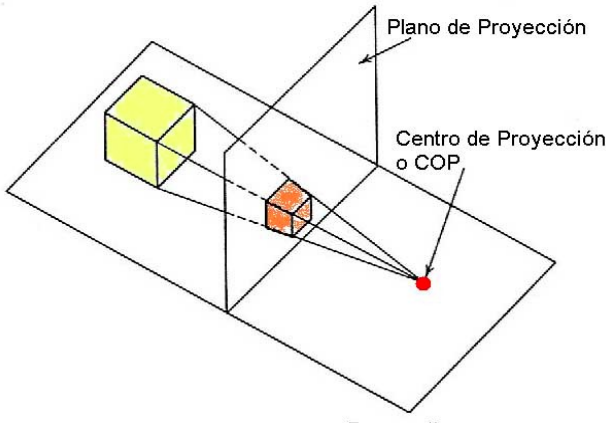


Figura 2.

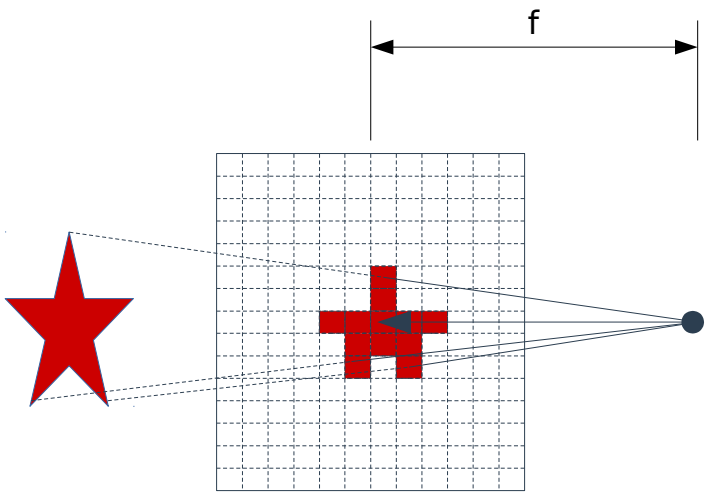


Figura 3.

Proyección de una trayectoria a la cámara.

Generaremos una trayectoria tridimensional suponiendo que es el camino que recorre el ladrón en nuestro patio. Para el ejemplo que explicaremos paso a paso, se usó la función definida por: $f(t) = (t, \cos(t)+2, \sin(2t)+1)$. Donde t es el tiempo que es tomado entre cada imagen de la cámara. Se explicará detalladamente con esta trayectoria y después se darán los resultados obtenidos con diferentes trayectorias. Usualmente, un ladrón solamente haría una trayectoria simple, como en línea recta, pero vamos a probar con cosas más complejas para ver la capacidad de precisión que podemos alcanzar. Las figuras 4.1, 4.2, 4.3 y 4.4 representan la trayectoria tridimensional $f(t)$ de distintas perspectivas.

Para proyectar la trayectoria a la cámara vamos a necesitar la ecuación del plano proyectivo de la cámara, pues la estrategia es intersectar la recta que pasa por el punto medido en el tiempo t, digamos el punto p, y el foco te nuestra cámara, con dicho plano. Si el vector **s** es paralelo a dicha recta, la ecuación de la recta es: $(x\text{-}p_1)/s_1 = (y\text{-}p_2)/s_2 = (z\text{-}p_3)/s_3$. Sabemos que las cámaras apuntan al centro del patio, así podemos obtener el vector $\boldsymbol{u}$ restando la posición de la cámara y el punto a donde apunta dicha cámara. Este vector $\boldsymbol{u}$, si lo normalizamos, es un vector normal al plano proyectivo. Adémas, sabemos que dicho plano está a distancia **f** del foco, entonces el punto **v** en el que intersecta el vector $\boldsymbol{u}$ que pasa por el foco de la cámara al plano está dado por la posición de la cámara - $c\boldsymbol{u}$, donde $c = \mathbf{f}/\boldsymbol{u}$. De esta manera, la ecuación del plano proyectivo la obtenemos con la ecuación:
$v_1(x - u_1) + v_2(y - u_2) + v_3(z - u_3) = 0$. Así, podemos encontrar el punto de intersección real resolviendo el sistema de ecuaciones siguiente para cada uno de los puntos medidos:

$$\begin{vmatrix} v_1 & v_2 & v_3 \\ 1/s_1 & -1/s_2 & 0 \\ 1/s_1 & 0 & -1/s_3 \end{vmatrix} \begin{vmatrix} x \\ y \\ z \end{vmatrix} = \begin{vmatrix} v_1u_1 + v_2u_2 + v_3u_3 \\ p_1/s_1 - p_2/s_2 \\ p_1/s_1 - p_3/s_3 \end{vmatrix}$$

Recordamos que los puntos de la cámara son percibidos por los pixeles, tenemos NPxMP de ellos y cada uno de ellos es de lado ε, entonces lo que podemos hacer es la discretización en el plano xy antes mencionada y transladamos cada punto obtenido al plano "artificial" que construimos, para después ver en qué intervalo de nuestra discretización cayó el punto y diremos que éste fue el pixel que lo percibió.

La estrategia que se usó para trasladar cada punto al plano xy, fue encontrando la intersección del plano proyectivo con la vertical donde se posiciona la cámara; después se calculó el ángulo entre ese vector y el vector del punto respecto al centro del plano proyectivo (ver figura 5) con la fórmula: $\alpha = \arccos(u.v/(||u||\ ||v||))$. Teniendo lo anterior, ponemos el vector de igual longitud que el punto respecto al centro del plano proyectivo y rotamos por el ángulo obtenido con la transformación $(x, y) = (x, y)[(\cos\alpha \ \ -\sin\alpha), (\sin\alpha \ \ \cos\alpha)]$.

Teniendo esto, solo queda verificar en que pixel esta nuestro punto y ya con eso obtenemos la imagen de nuestra trayectoria. Para el ejemplo que realizamos con $f(t)$, observamos las imagenes de las cámaras obtenidas en las figuras 6.1 y 6.2. Dichas cámaras puestas en un patio de dimensiones 8x8, la cámara 1 puesta en la esquina (0,0,3) y la cámara 2 en (8,0,3), ambas apuntando al centro del patio.

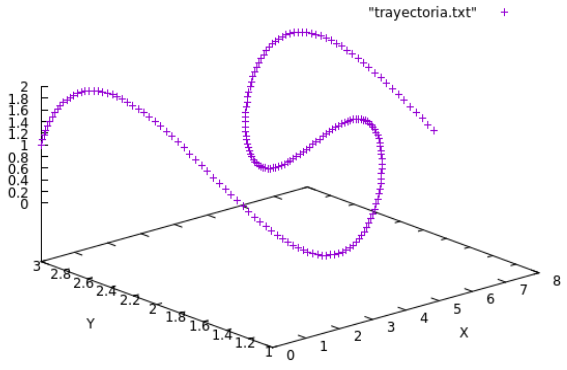


Figura 4.1

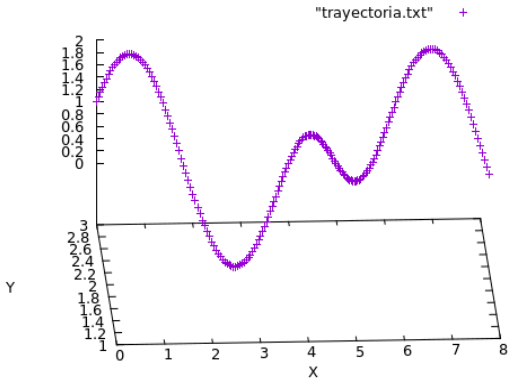


Figura 4.2

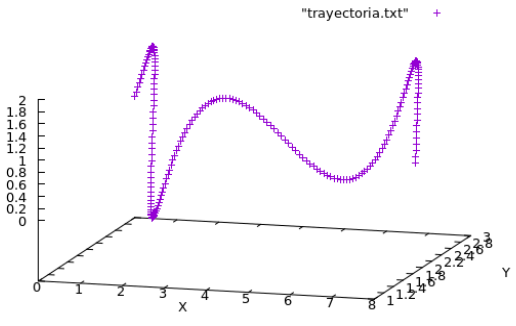


Figura 4.3

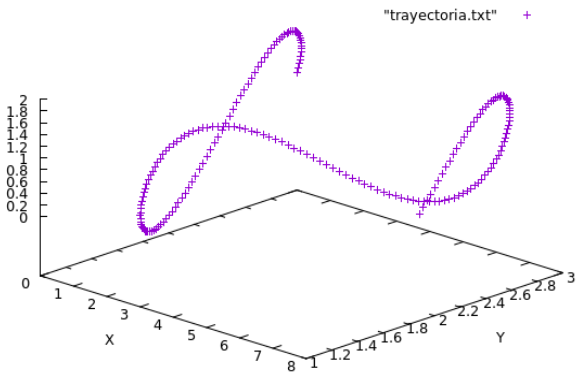


Figura 4.4

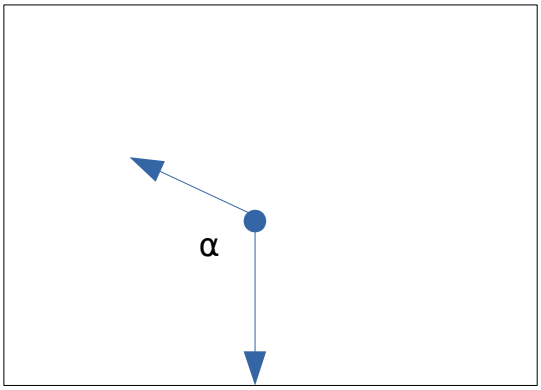


Figura 5.

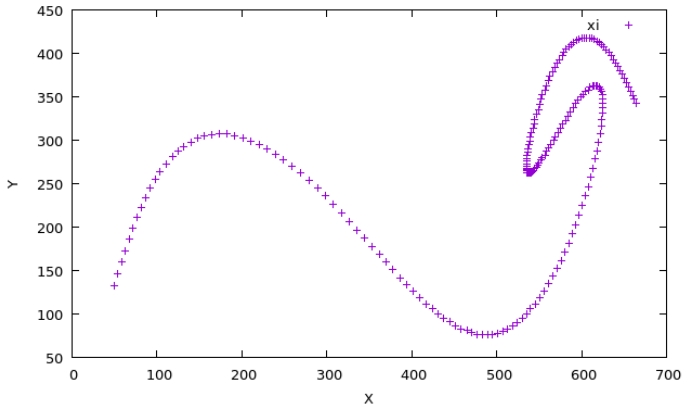


Figura 6.1

Recordando que las imágenes obtenidas con las cámaras están en escala de pixeles. Para este ejemplo, la cámara cuenta con 800 pixeles de ancho y 600 pixeles de alto.

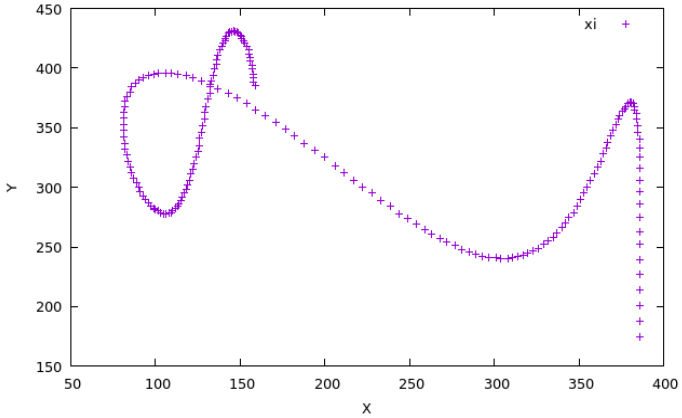


Figura 6.2

Reconstrucción de trayectoria.

Escalación de imágenes a plano proyectivo.

Como sabemos, la imagen es de NPxMP pixeles de lado ε; hacemos la discretización $x_i = i\varepsilon$, para i = 0, 1, …, NP. Y en el caso vertical con: $y_i = i\varepsilon$, para i = 0, 1, …, MP. Entonces, si el punto de la trayectoria esta en el pixel (n,m), lo vamos a aproximar al punto $(n\varepsilon, m\varepsilon) - (\varepsilon/2, \varepsilon/2)$. Así, obtenemos la escalación de las imágenes en el plano (ver figura 7.1 y 7.2) y procedemos a trasladarlo a la posición de las cámaras para realizar la reconstrucción.

Figura 7.1

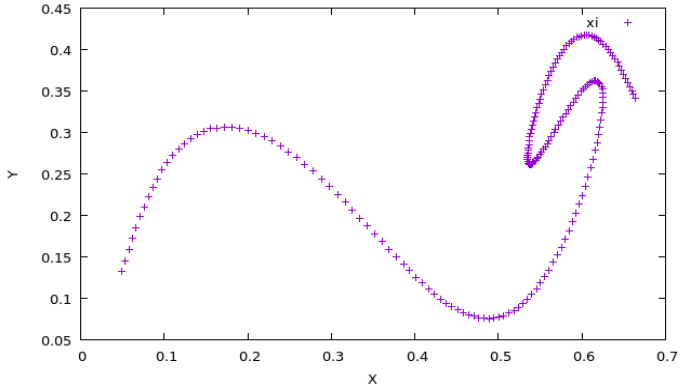
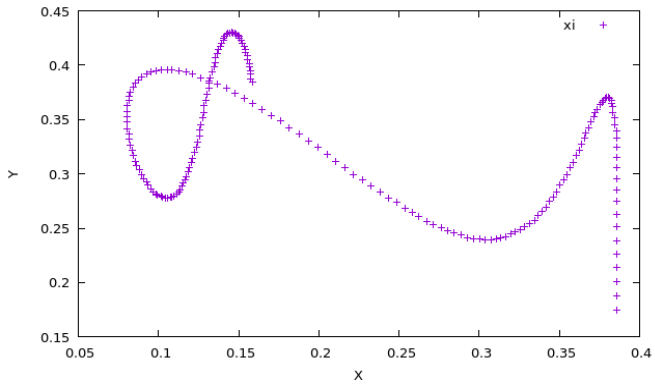


Figura 7.2



Traslación del plano.

Para cada punto de la proyeccion del plano, lo que vamos a hacer es rotarlo en dirección de donde apunta la cámara y después rotarlo verticalmente para que coincida con el plano proyectivo real de la cámara.

Para rotar cada punto, nos fijamos en la dirección en que apunta a la cámara y rotamos el punto respecto al centro del plano, esto lo hacemos con la fórmula de ángulo entre vectores ya antes mencionada entre el eje y de la cámara y el vector de a donde apunta la cámara. Después lo rotamos verticalmente en el ángulo de 90^{o}-α, donde α es el ángulo entre el plano xy y el vector que va del foco de la cámara a donde apunta. Así, obtenemos el plano de los puntos de la trayectoria ya puesto en el lugar del plano proyectivo de cada cámara. Podemos observar el plano de la cámara 1 en la figura 8.1, 8.2 y 8.3 , y el plano proyectivo de la cámara 2 en la figura 8.4, 8.5 y 8.6.

Figura 8.1

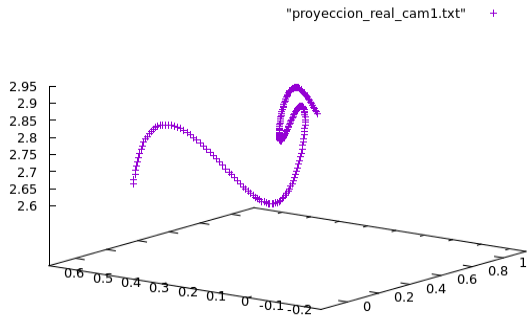


Figura 8.2

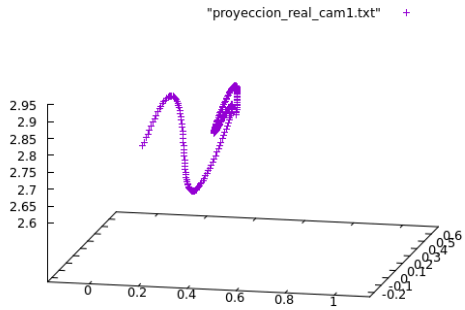


Figura 8.3

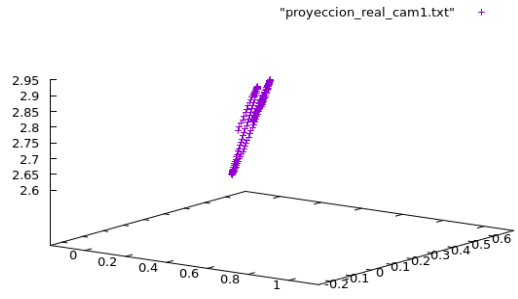


Figura 8.4

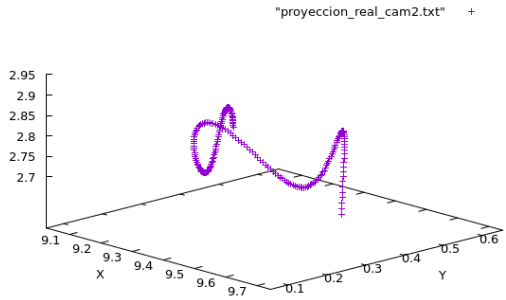


Figura 8.5

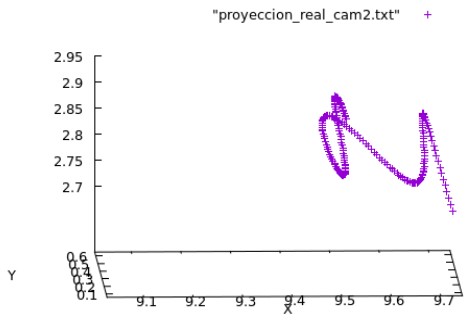
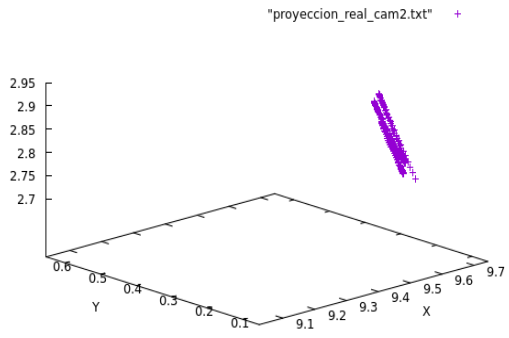


Figura 8.6



Reconstrucción por rectas.

Si nuestros cálculos fueran exactos, para reconstruir la trayectoria bastaría con extender las rectas que pasan por el foco de las cámaras y un punto del plano proyectivo correspondiente en el tiempo en ambas cámaras, al extenderlas nos fijaríamos en su intersección y este punto sería el punto tridimensional de la trayectoria. Sin embargo, sí tenemos pequeños errores al percibir las cosas, pues si recordamos, los pixeles solo percibían todos los rayos de luz en el área que abarcaran, entonces si dos puntos o más de la trayectoria se proyectaban en el mismo pixel, perderíamos información pues solo detectaríamos que los dos están en el mismo lugar cuando no es así. Además, también perdemos información sobre la posición exacta del punto en la trayectoria pues nosotros, al escalar la imagen al plano, lo estamos aproximando por el centro de cada pixel. Por lo tanto, sí tenemos muchísimos errores aunque sean pequeños.

Debido a estos errores, al extender las rectas, muchas veces no se intersectan las rectas, entonces debemos fijarnos en la mínima diferencia entre las rectas y éste sería nuestro punto.

Reconstrucción por planos y rectas.

Con el objetivo de encontrar la diferencia mínima entre las dos rectas, lo que haremos en vez de extender las dos rectas es que extenderemos una recta desde una cámara y desde la otra cámara extenderemos el plano vertical completo que contiene a la otra recta. De esta manera aseguramos que va a tener solución al mirar su intersección. El procedimiento para encontrar la distancia mínima de las dos rectas es hacer lo anterior para una cámara, encontramos su intersección, después hacerlo para la otra cámara y de esta manera obtendremos los dos puntos más cercanos en las distintas rectas (podemos ver los resultados obtenidos por el plano sobre la primera cámara en las figuras 9.1, 9.2 y 9.3, y los resultados con el plano en la 2da. cámara en las figuras 9.4, 9.5 y 9.6. A los puntos encontrados los vamos a aproximar por el punto de en medio, el cual es el promedio de los dos puntos. (ver figura 9.7).

Realizamos este proceso para todos los puntos y así reconstruimos la trayectoria. Calculamos el error que tenemos al reconstruir la trayectoria con $E=||p_i - f(x_i)||$ donde p_i es el punto reconstruido en el tiempo i medido. Podemos observar la trayectoria reconstruida en la figura 9.8. A continuación, una tabla con los errores de reconstrucción para diferentes trayectorias. El error 1 es el de reconstrucción con el plano en la 1ra. cámara, el Error 2 es el de reconstrucción por el otro lado, y el Error de reconstrucción es el promedio generado.

Función $f(t)$	Error 1	Error 2	Error reconstrucción
(t, cos(t)+2, sin(2t)+1)	0.167	0.151	0.146
(t, 2t+2, sin(t)+1)	0.199	0.177	0.185
(sin(t)+4, 2t+2, sin(t)+1)	0.188	0.163	0.173

De la tabla podemos observar que en un caso, el error del promedio mejora la aproximación, sin embargo, en dos de los casos fue mejor aproximación hacerlo por el plano en la 2da. cámara.

Figura 9.1

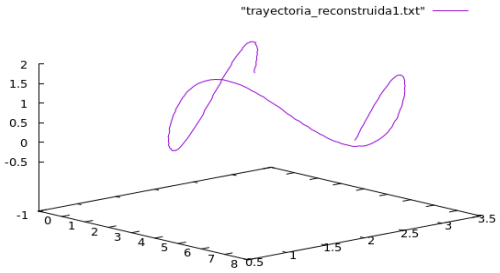


Figura 9.2

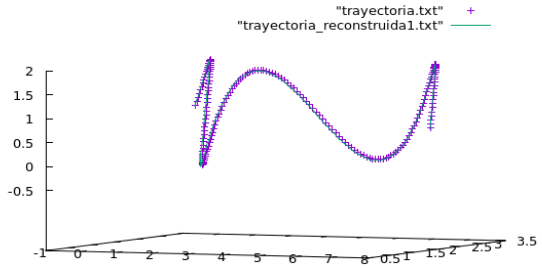


Figura 9.3

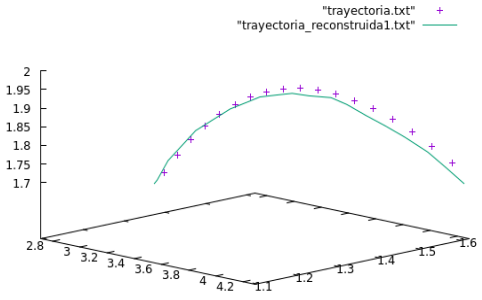


Figura 9.4

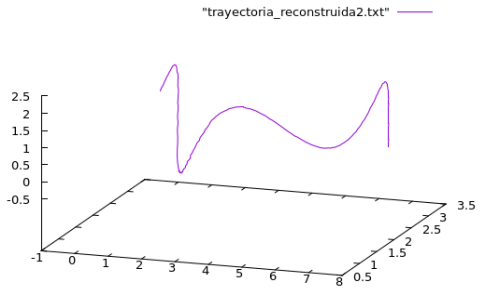


Figura 9.5

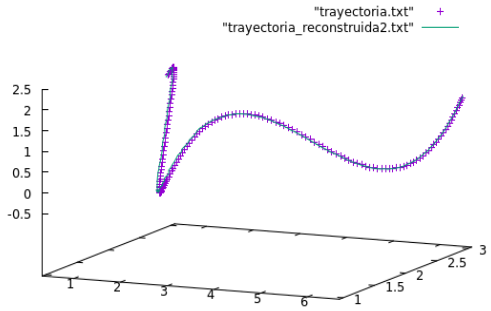


Figura 9.6

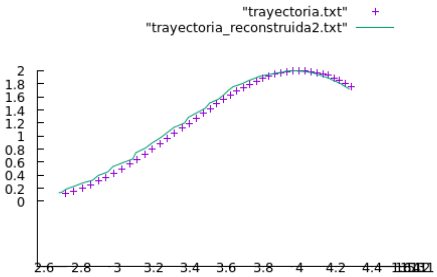


Figura 9.7

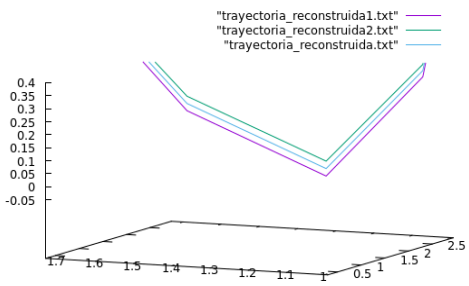
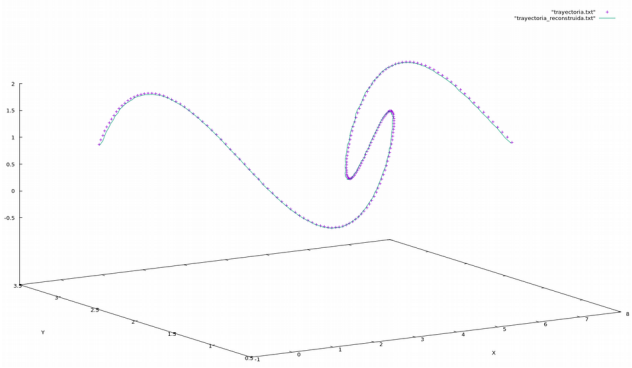


Figura 9.8



Extrapolación de la trayectoria.

Métodos de extrapolación con polinomios.

Ya hemos reconstruido la trayectoria tridimensional del ladrón, ahora lo que sigue es tratar de predecir si posición en un instante después para poder hacer los cálculos y lanzar el dardo que lo va a dormir. Para esto, nuestra primera opción es predecir la posición del ladrón por medio de un polinomio interpolador el cual pasa por los puntos medidos anteriormente y vamos a seguir como si este polinomio continuara; en otras palabras, supongamos que la trayectoria medida se puede ver como un polinomio, entonces para predecir el lugar donde estará en el siguiente instante, solo tendríamos que evaluar el polinomio en dicho tiempo.

Para nuestro caso, esto resulta demasiado ineficiente e inexacto, pues las mediciones del tiempo las estamos haciendo muy cerca y además tenemos demasiados puntos, así que encontrar el polinomio interpolador causa muchos problemas porque tanto para el polinomio de Lagrange como el de Newton se divide entre la diferencia de cada punto, y si esta es muy pequeña (que es nuestro caso) el valor de la función se dispara. Para estos casos se probó con el polinomio de Lagrange y de Newton, los cuales fueron muy malos aproximando, incluso en valores cercanos al último punto (daban error de aproximación mayores a 100). Así que optamos por tratar de predecir la posición con los métodos explicados a continuación.

Extrapolación lineal.

La extrapolación lineal, lo que hará es predecir el siguiente punto a partir de una recta formada por los dos últimos puntos medidos. La fórmula de aproximación lineal está dada por: $f(t_{i+1}) = f(t_{i-1}) + (f(t_i) - f(t_{i-1}))(t_{i+1} - t_{i-1})/(t_i - t_{i-1})$. Podemos fácilmente darnos cuenta de que este método solo es bueno en tiempos cercanos al último punto medido, pues la función cambia de valores mientras nosotros estamos siguiendo en linea recta. Los errores de predicción de este método están dados en la siguiente tabla y podemos observar la predicción en las figuras 10.1, 10.2 y 10.3 de la primera aproximación.

Tiempo de predicción	Error $\|\|f(t_{i+1}) - p_{i+1}\|\|$
0.5 s	0.315
1 s	0.262
3 s	4.39
5 s	7.67

Observamos que lo antes mencionado se cumplió para este ejemplo pues para tiempos pequeños de predicción tenemos un error pequeño, empero cuando queremos predecir la posición con más tiempo, la recta se aleja de la solución real.

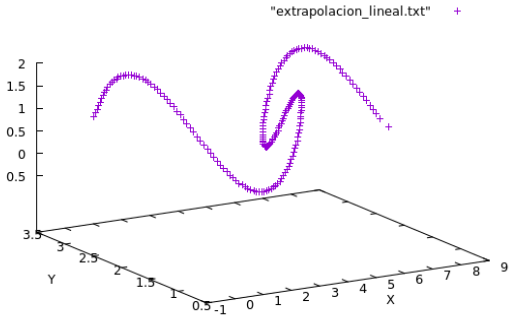


Figura 10.1

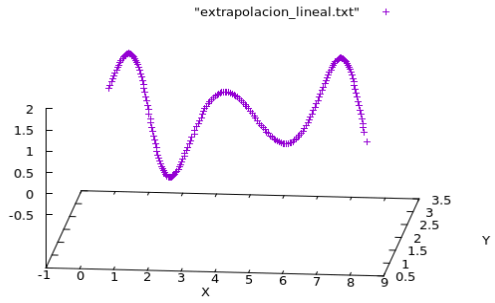


Figura 10.2

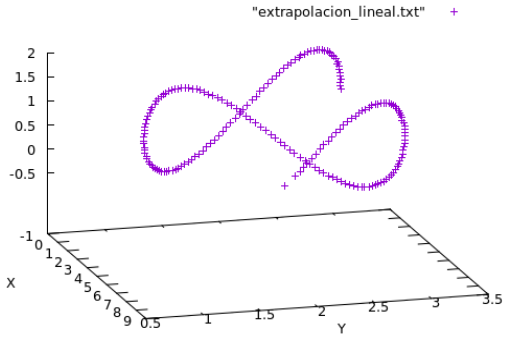


Figura 10.3

Extrapolación con spline.

Anteriormente notamos que la aproximación lineal no era muy buena cuando nos alejabamos mucho en el tiempo de predicción, así que una estrategia muy buena sería predecir el siguiente punto por medio de extrapolación con un spline cúbico, es decir, dados los puntos de la trayectoria, construimos splines cúbicos para interpolar la función y para extrapolar el punto deseado vamos a evaluar el último spline encontrado y evaluamos el tiempo deseado en dicho spline (aunque el punto esté fuera, consideramos que está dentro del último intervalo y lo evaluamos con este spline).

Los errores de predicción de este método están dados en la siguiente tabla y podemos observar la predicción en las figuras 11.1, 11.2 y 11.3 de la primera aproximación.

Tiempo de predicción	Error $\|\|f(t_{i+1}) - p_{i+1}\|\|$
0.5 s	0.047
1 s	0.4
3 s	1.89
5 s	47.3

Lo cual, mejora las cosas pero solo en los puntos medianamente cercanos pues como es un polinomio cúbico cuando se aleja puede expolatar peor que como si fuera lineal.

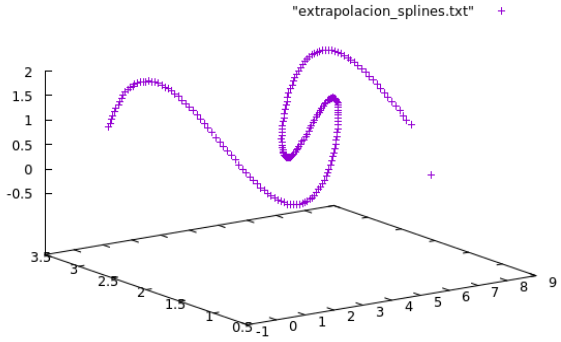


Figura 11.1

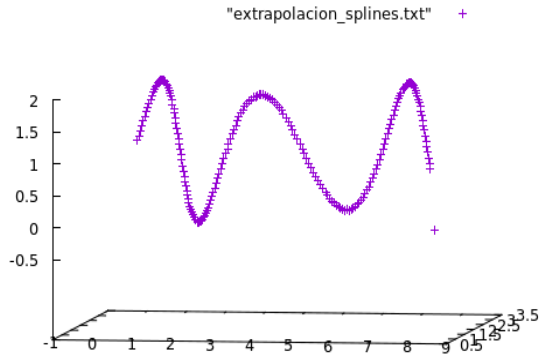


Figura 11.2

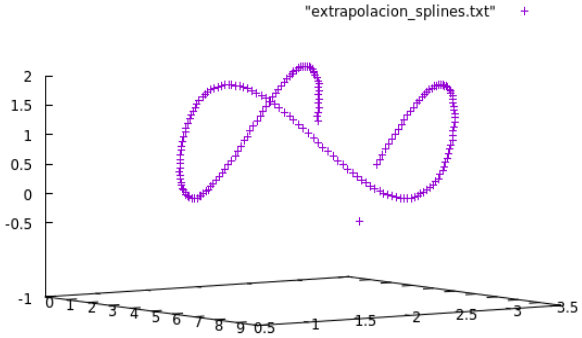


Figura 11.3

Otros resultados.

Veamos en las figuras 12.1 y 12.2 la misma trayectoria del ejemplo pero con menos puntos de medición. La primera figura es la trayectoria reconstruida y la siguiente figura es la extrapolación en 1 segundo. Todo esto para ver que tan bien se trabaja si tenemos menos datos.

Función f(t)	Error 1	Error 2	Error reconstrucción	Error lineal	Error spline
(t, cos(t)+2, sin(2t)+1) con menos puntos	0.167	0.151	0.146	0.021	0.047
(t, 2t+2, sin(t)+1)	0.199	0.177	0.185	0.085	0.22
(sin(t)+4, 2t+2, sin(t)+1)	0.188	0.163	0.173	8.73	6.27

A continuación podemos también observar la trayectoria de las otras funciones en las figuras 12.3 y 12.4.

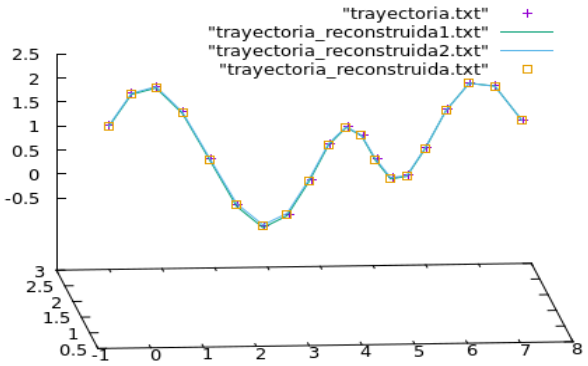


Figura 12.1

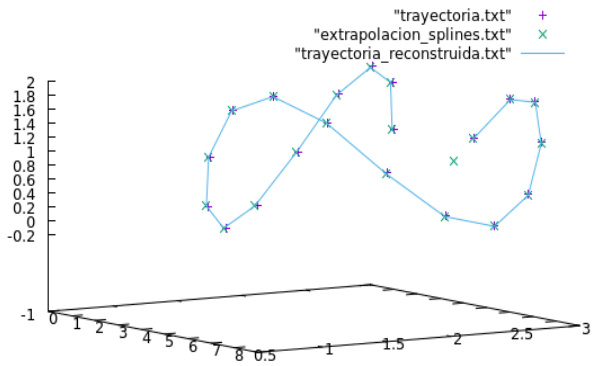


Figura 12.2

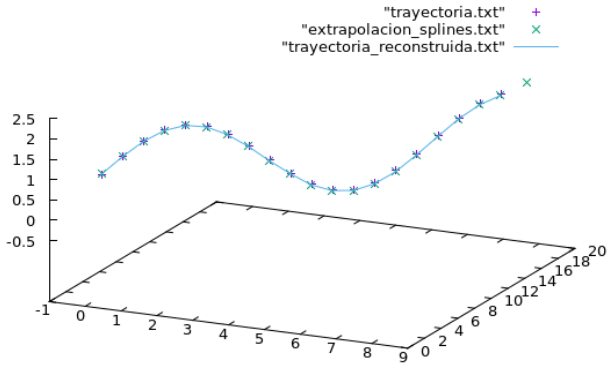


Figura 12.3

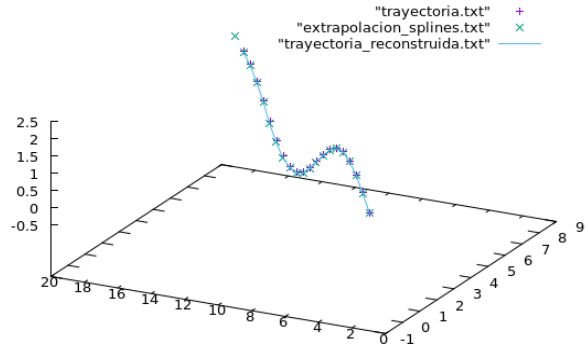


Figura 12.4

Conclusiones.

El objeto del proyecto se cumplió, pues lo que se buscaba principalmente es a partir de imágenes bidimensionales poder construir una trayectoria tridimensional, lo cual se logró satisfactoriamente pues no hubo mucho error en esos casos.

Como estamos limitados físicamente por la resolución de la cámara, resulta imposible restaurar exactamente el punto tridimensional pues perdemos información al proyectar desde los pixeles y lo que nos queda, es tratar de reducir el error de reconstrucción.

Respecto al objetivo de la predicción, los resultados fueron buenos, porque supongamos que sí lanzaremos el dardo, entonces éste se lanza y llega a su destino en 1 segundo más o menos (si es un patio no inmenso), y a parte tenemos un margen de error bastante amplio pues el ladrón no sería un punto, si no una masa con mucho más volúmen. Como observamos, si queremos predecir muy por delante del presente, los errores son fatales en los experimentos.

Futuro del proyecto.

Se puede seguir trabajando en mejorar el tipo de predicción, por ejemplo, se podría introducir un análisis de movimiento para resolver el problema de extrapolación con una ecuación diferencial. También se podría extender a utilizar machine learning para que cada movimiento que haga el ladrón, sea un tipo de entrenamiento para la red y así pueda aprender el algoritmo para predecir la posición del ladrón.