UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

IMPLEMENTACION DE ODOMETRÍA VISUAL UTILIZANDO UNA CAMARA ESTEREOSCÓPICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

ANDRÉS PEÑALOZA GONZÁLEZ

PROFESOR GUÍA:
MARTIN DAVID ADAMS

MIEMBROS DE LA COMISIÓN:
CLAUDIO PEREZ FLORES
MARCOS EDUARDO ORCHARD CONCHA

SANTIAGO DE CHILE
2015

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE
INGENIERO CIVIL ELECTRICISTA
POR: ANDRÉS PEÑALOZA GONZÁLEZ
FECHA: 09/10/2015
PROFESOR GUÍA: MARTIN DAVID ADAMS

IMPLEMENTACION DE ODOMETRÍA VISUAL UTILIZANDO UNA CAMARA ESTEREOSCOPICA

En ciertas aplicaciones de robótica es importante la utilización de un odómetro para poder estimar la posición de un robot en movimiento. Esto permite que el actor tenga una noción de la ubicación en el entorno por donde se mueve. En aplicaciones como vehículos autónomos es especialmente importante, pues es crítico conocer la posición del vehículo con respecto a su mapa interno para evitar colisiones. Usualmente los odómetros más utilizados son las ruedas y el GPS. Sin embargo estos no siempre están disponibles, debido a adversidades del ambiente. Es por estos motivos que se emplea odometría visual.

La odometría visual es el proceso de estimación del movimiento de un vehículo o agente utilizando las imágenes que éste obtiene de sus cámaras. Ella ha sido utilizada en la industria minera con los camiones de carga, y, últimamente en drones aéreos que podrían ser ocupados para el transporte de paquetes. También se ha utilizado para estimar la posición de los robots que actualmente transitan en la superficie de Marte.

El presente trabajo tiene por finalidad la implementación de un algoritmo de odometría visual usando una cámara estereoscópica para estimar la trayectoria de un robot, y la evaluación del desempeño de éste comparándola con los valores conocidos de posición. La metodología utilizada permite identificar qué parámetros del algoritmo de estimación de movimiento tienen mayor relevancia y cómo influyen en la rapidez y calidad de la solución. También se determina la influencia de las condiciones de iluminación, y se determina qué zona geométrica de la imagen es mejor para realizar la triangulación de puntos.

La solución se compone de un sistema capaz de ejecutar las distintas partes que requiere el algoritmo de manera extensible, siendo fácil reemplazar un método en el futuro con un mínimo impacto en el código. Se obtienen resultados favorables, donde el error de estimación de movimiento es pequeño y, además, se concluye acerca de los factores más importantes en la ejecución del algoritmo. Se discute acerca de la rapidez del algoritmo y se proponen soluciones que ayuden a su implementación en tiempo real.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

Capítulo 1: Introducción

1.1 Fundamentos y Contextualización del Problema

En ciertas aplicaciones de robótica es importante poder estimar la posición de un robot en movimiento. Esto permite que el actor tenga una noción de la ubicación en el entorno por donde se mueve. En aplicaciones como vehículos autónomos es especialmente importante pues es crítico conocer la posición del vehículo con respecto a su mapa interno para evitar colisiones. La odometría se utiliza para poder realizar la estimación. Usualmente los odómetros más utilizados son las ruedas, él y GPS. Sin embargo estos no siempre están disponibles, debido a adversidades del ambiente.

La odometría visual es el proceso incremental de estimación del movimiento de un agente examinando los cambios que el movimiento induce en las imágenes capturadas sus cámaras.

La ventaja de la odometría visual sobre la odometría de ruedas es que la primera no se ve afectada por resbalamientos en las ruedas o condiciones de terreno adversas. Se ha demostrado que la odometría visual provee resultados más acertados que la odometría de ruedas, y por lo tanto la hace un suplemento importante para la odometría de ruedas y otras, como GPS, odometría laser.

La odometría visual permite capturar la posición del vehículo en movimiento en condiciones en que otro tipo de odometría no está disponible. Es especialmente importante en entornos donde no existe señal de GPS, como ciudades altas (alta densidad de edificios), túneles, aplicaciones submarinas y aéreas. También en terrenos extremadamente adversos, como la arena, donde la odometría de ruedas simplemente no funciona.

Los avances en esta área de investigación contribuyen dan luz para una nueva generación de vehículos de conducción autónoma. Estos vehículos son especialmente importantes en la industria minera con los camiones de carga, y últimamente en drones aéreos que se podrían ser utilizados para el transporte de paquetes.

1.2 Objetivo General

El objetivo general de este trabajo es el de implementar un algoritmo de odometría visual que permita a un robot estimar su posición en todo momento utilizando una cámara estereoscópica.

1.3 Objetivos Específicos

- Implementar algoritmos para convertir las imágenes estereoscópicas de la cámara en una imagen con información de profundidad.
- Calcular el movimiento del robot utilizando fotos tomadas en distintos intervalos de tiempo.
- Leer los datos de la cámara utilizando ROS (Robot Operating System).
- Crear una librería estándar que permita ejecutar las partes del algoritmo de manera extensible, de forma que se puedan intercambiar partes de este de manera sencilla.
- Medición del *performance* del algoritmo para distintas configuraciones de parámetros, y comparación con la posición conocida de este.

1.4 Estructura del Documento

En el Capítulo 2 se introducen los conceptos básicos que formaran la base para entender el tema propuesto en este trabajo. Se realiza una revisión bibliográfica acerca de los distintos algoritmos de detección de características, *matching* o emparejamiento. También se revisan conceptos como triangulación de características y algoritmos de rechazo de puntos contaminados.

En el Capítulo 3 se describe la implementación en código del sistema para las distintas etapas del proceso, y se dan a conocer algunos experimentos que permitirán conocer la influencia de algunos parámetros dentro el resultado final.

En el Capítulo 4 se muestran los resultados de la implementación del algoritmo de odometría visual, así como también los resultados de realizar los experimentos definidos en el Capítulo 3.

En el Capítulo 5 se dan a conocer las conclusiones obtenidas a lo largo del desarrollo del trabajo y se proponen ideas de nuevos experimentos y trabajos futuros.

Capítulo 2: Revisión Bibliográfica y Contextualización

2.1 Vista General del Algoritmo

La odometría visual es una técnica utilizada para estimar la posición de un vehículo mediante la utilización de una cámara. En la Ilustración 1 se observa el diagrama de flujo típico del algoritmo, que está dividido en varias etapas. En primera parte, la etapa de recolección de fotografías, en la cual se extrae la información proveniente de una cámara instalada en alguna parte del vehículo en movimiento. La etapa siguiente corresponde a la detección de características en la imagen obtenida en la etapa anterior, típicamente esquinas o manchas. En la etapa tercera se realiza un emparejamiento o *matching* entre las características encontradas en distintas imágenes, es decir, se encuentran los puntos que correspondan a un mismo objeto en observación para imágenes en intervalos de tiempo sucesivos. La etapa posterior toma el grupo de parejas de puntos y remueve los puntos contaminados con ruido, o asociaciones de puntos incorrectas. En la etapa final se realiza el cálculo de movimiento utilizando los puntos considerados buenos. La trayectoria final será la concatenación de todas las posiciones encontradas para cada par de imágenes sucesivas.

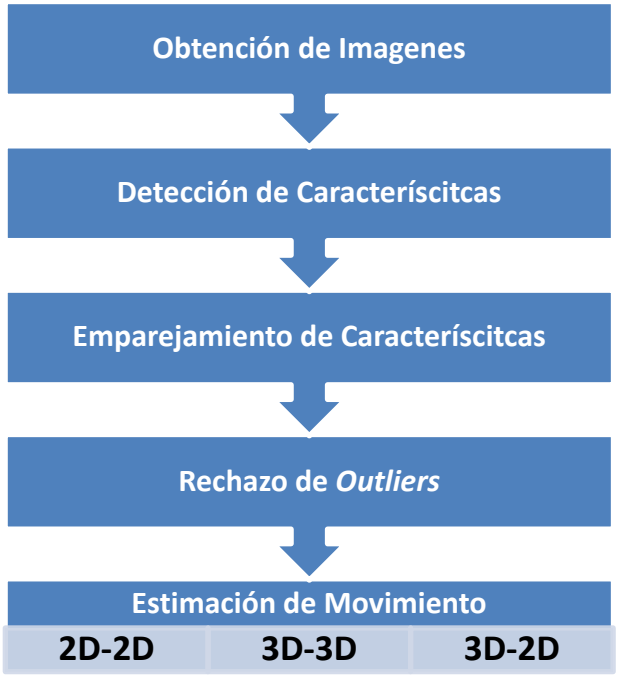


Ilustración 1: Diagrama de flujo típico de odometría visual, dividida en 5 etapas.

La estimación de movimiento se divide básicamente en 3 tipos distintos, que dependen de la configuración física de cámaras y numero de estas: 2D-2D, 3D-3D y 3D-2D.

En primer lugar tenemos la estimación 2D-2D. Sean f_{k-1} y f_k los sets de características correspondientes a las imágenes I_{k-1} e I_k, para algún momento k. Entonces la dimensión

de estas características es dos para los ambos momentos $k-1$ y k. En este tipo se utiliza una sola la información de una sola cámara. En la Ilustración 2 se observa un set de características encontradas en la imagen I_{k-1} y sus correspondencias en I_k. El movimiento de la cámara queda descrito por la matriz de transformación T_k.

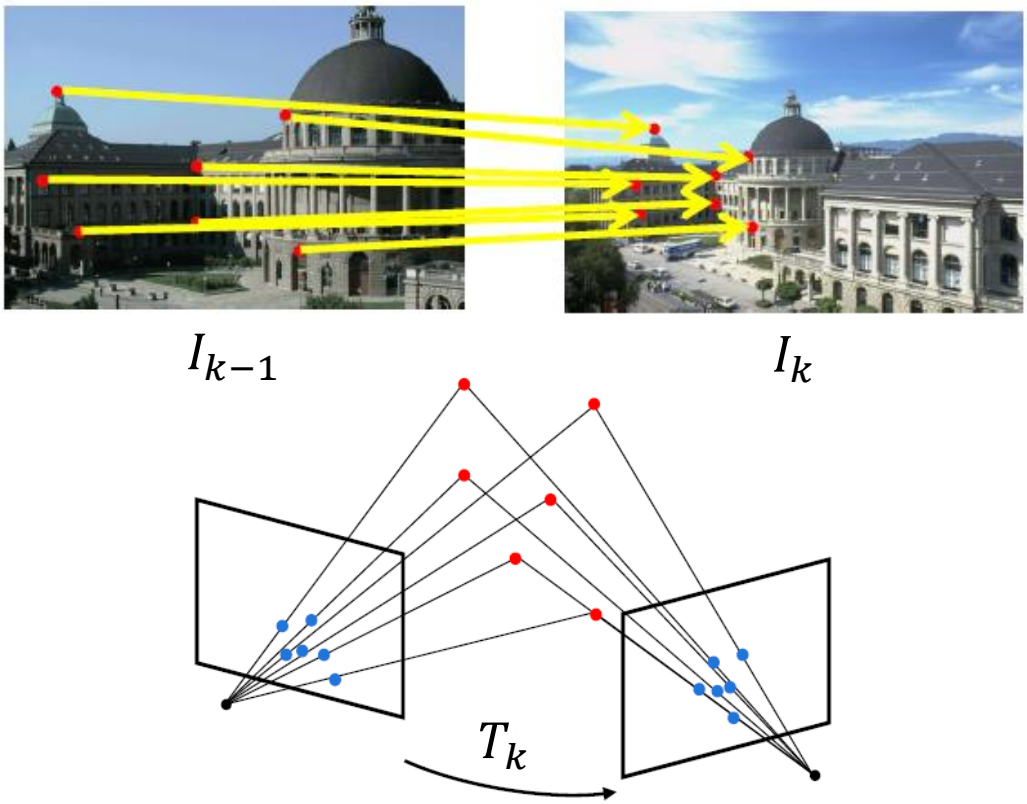


Ilustración 2: Correspondencia de características entre dos imágenes sucesivas para el caso 2D-2D. Autor de la imagen: Davide Scaramuzza.

En la estimación 3D-3D se utiliza un arreglo de 2 cámaras que pueden estar dispuestas en un arreglo paralelo o no-paralelo. Utilizando la información de ambas cámaras para intervalos de tiempo simultáneos, se puede determinar la profundidad para cada punto de la imagen. En este caso ambos sets f_{k-1} y f_k están especificados en tres dimensiones. En la Ilustración 3 se observa la correspondencia del set de características entre I_{k-1} y I_k. El movimiento de la cámara queda descrito por la matriz de transformación T_k. En este caso se observa que cada imagen 3D I_k esta descrita por dos imágenes 2D $I_{izquierda}$ e $I_{derecha}$, de las cuales se puede realizar la triangulación.

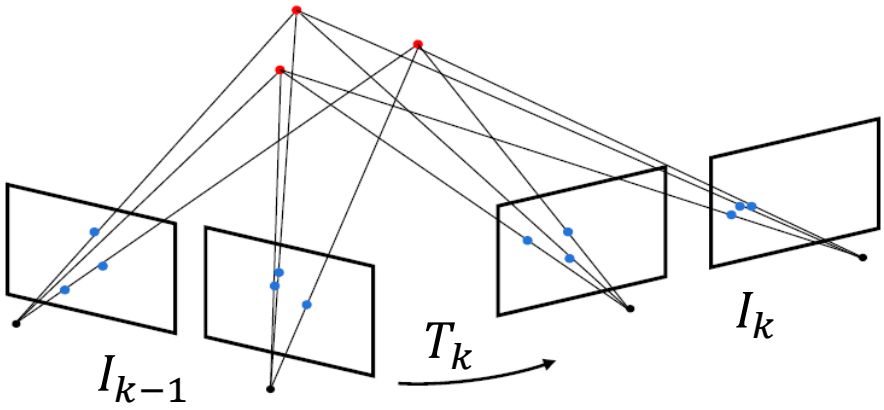


Ilustración 3: Correspondencia de características entre dos imágenes sucesivas para el caso 3D-3D. Autor de la imagen: Davide Scaramuzza.

El caso 3D-2D es un caso híbrido entre los dos anteriores, y se puede realizar con solo una cámara, al igual que en el caso 2D-2D. Ahora f_{k-1} está especificado en tres dimensiones, pero f_k tendrá solo dos dimensiones. Para poder obtener la información de profundidad se realiza una triangulación entre las imágenes I_{k-2} e I_{k-1}. Luego se realiza el emparejamiento con f_k en una tercera imagen I_k.

2.2 Detección de Características

La detección de características consiste en reconocer ciertos puntos de interés en una imagen, llamados *keypoints*, (puntos clave o características, en castellano). Estos puntos pueden ser de distintos tipos: bordes, esquinas, manchas, entre otros.

2.2.1 Detección de esquinas de Harris

Uno de los primeros intentos de detección de características empleado fue *Harris Corner Detection* [1] en 1988. La idea principal es detectar las esquinas presentes en la imagen, donde existe una alta variación de intensidad en todas las direcciones.

Para una coordenada de desplazamiento (u, v), sea la función $E(u, v)$ dada por:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \qquad (1)$$

Con $w(x, y)$ siendo la función de ventana, que está dada por una función gaussiana, asignando pesos a los pixeles descritos por esta. La función denotada por $I(x, y)$ es la intensidad del pixel en las coordenadas (x, y).

Los puntos detectados provienen de los máximos locales de la función descrita en (1). En la Ilustración 4 se muestran los puntos detectados en tres imágenes de prueba diferentes: una cuadricula en blanco y negro recta, deformada, y un escenario en 3 dimensiones.

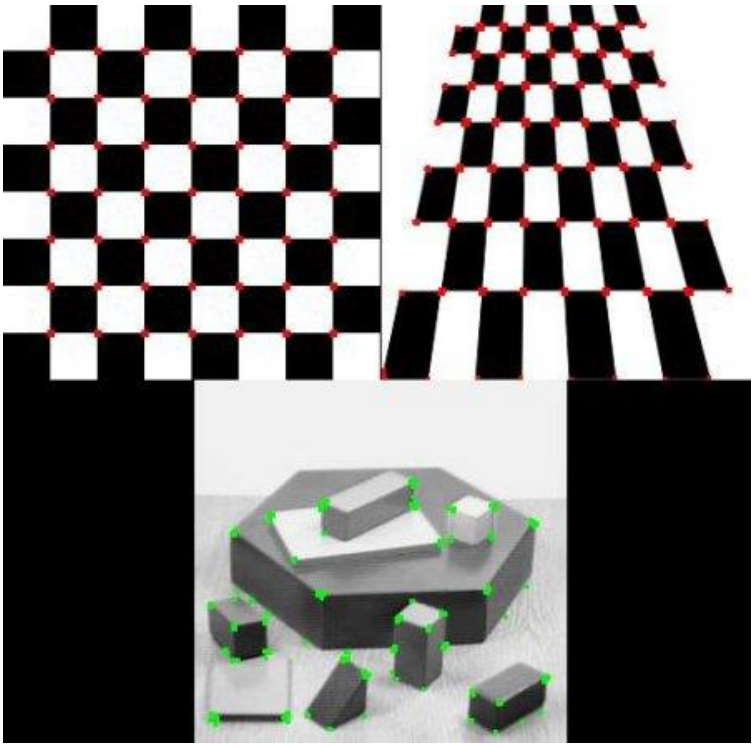


Ilustración 4: Aplicación de la detección de esquinas de Harris, en tres escenarios distintos. Fuente de la imagen: OpenCV.org [2].

La detección de esquinas de Harris posee invariabilidad frente a rotaciones en la imagen; sin embargo no presenta inmunidad frente a imágenes escaladas, es decir, un punto detectado en una imagen de cierto tamaño, podría desaparecer si la imagen es agrandada, pues la ventana se hace comparativamente más pequeña.

2.2.2 SIFT

El método *Scale-Invariant Feature Transform*, o SIFT, fue propuesto en 2004 [3] como una nueva técnica que presenta invariabilidad frente a escalas en la imagen. El algoritmo se divide en 4 partes.

En la primera, la imagen se le aplica un filtro denominado *Laplacian of Gaussian* (LoG), que servirá como detector de manchas de distintos tamaños en la imagen asignándoles un factor de escala σ. Con este factor de escala, se buscan los máximos locales en la imagen, lo que genera una lista de puntos (x, y, σ) que indica que podría existir un punto clave (*keypoint*) en (x, y) a escala σ.

La segunda etapa se encarga filtrar la lista de candidatos, considerando información de contraste y detección de bordes. Si el *keypoint* no tiene suficiente contraste, es decir, es menor a un cierto valor definido, se elimina de la lista. También, se utiliza un detector parecido al de Harris para detectar los bordes en la imagen. Si se encuentra que el *keypoint* está en un borde, entonces también se elimina de la lista.

En la tercera parte del algoritmo se le asigna a cada candidato un ángulo de orientación. Se crea una vecindad alrededor del punto cuyo tamaño dependerá del factor de escala σ y se calcula la magnitud y dirección del gradiente dentro de esta región Luego se crea un histograma de orientaciones de 36 casillas que corresponden a 360º alrededor del punto. Este histograma se realiza en una ventana circular Gaussiana y cada valor de dirección de gradiente esta ponderada por la magnitud de este. La dirección final está dada por el mayor pico del histograma.

En la parte final se crea un descriptor para cada candidato, que considera una vecindad de 16×16 alrededor del punto, que estará dividida en 16 bloques de 4×4. Para cada sub-bloque se crea un histograma de orientación de 8 casillas. Estos se guardan dentro del descriptor, que servirá para diferenciar una característica de otra.

Esta técnica utiliza muchos recursos computacionales y, si bien da resultados muy favorables, es considerada lenta, y es desplazada por otras alternativas.

2.2.3 SURF

El método *Speeded-Up Robust Features*, o SURF, fue propuesto en 2006 [4] como una versión rápida de SIFT. Este algoritmo aproxima el filtro LoG de SIFT por un filtro de caja. En la Ilustración 5 se observa como el filtro de caja simplifica la imagen aproximando gradientes a bloques sólidos. El filtro de caja una capacidad de cómputo más rápida que LoG.

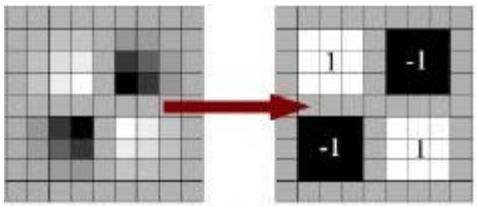


Ilustración 5: Aplicación de filtro de caja a una imagen de muestra. Fuente de la imagen: OpenCV.org [5].

SURF es el algoritmo predeterminado para la búsqueda de características, pues ofrece una velocidad que supera a SIFT y un performance similar. Otra ventaja del algoritmo es que puede ser paralelizado en una GPU, lo que disminuye su tiempo de ejecución aún más.

2.3 Emparejamiento de Características

El emparejamiento de características consiste en determinar cuáles características de un par de imágenes son correspondientes, es decir, corresponden a un mismo objeto en la imagen.

Se destacan dos algoritmos en la librería de OpenCV para este propósito: el algoritmo de fuerza bruta, y el algoritmo FLANN (*Fast Library for Approximate Nearest Neighbors*).

2.4 Triangulación

Triangulación se le llama al proceso de obtención de coordenadas 3D de un objeto en el mundo en base a un par de puntos 2D correspondientes en imágenes provenientes de cámaras distintas. La configuración física de las cámaras es comúnmente paralela, pero también se ocupan en configuraciones en distintos angulos.

Se utiliza el modelo de cámara estenopeica (*pinole camera*, en inglés) para representar la geometría de las cámaras. Este modelo es una aproximación que considera que la distancia focal f es mucho más pequeña que la distancia z a los objetos en la imagen. En la Ilustración 6 se detalla la proyección que recibe un punto p cualquiera al plano de imagen de la cámara.

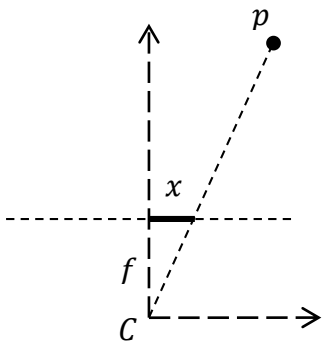


Ilustración 6: Modelo de camera estenopeica.

Este modelo no se puede ocupar en imágenes macro, ya que los objetos están muy cercanos a la cámara; es ideal para las condiciones en que se utilizará en este trabajo.

La relación de proyección entre el punto p y las coordenadas x proyectadas en el plano están dadas por:

$$x = \frac{f}{p_z} \cdot p_x \tag{2}$$

$$y = \frac{f}{p_z} \cdot p_y \tag{3}$$

2.4.1 Modelo de Cámara no Paralelo

En una configuración de cámaras se tiene comúnmente las cámaras alineadas un al lado de la otra, pero con una inclinación hacia adentro. Para esta configuración, mostrada en la Ilustración 7, un punto X en el mundo será proyectado a los centros de ambas cámaras C y

C'. Los puntos e y e', provenientes de la intersección de la recta entre ambos centro con los planos de la imagen, se llaman epipolos. Los puntos x y x' son la proyección del punto X en plano de las imágenes, y estos se encontraran siempre contenidos en la línea epipolar.

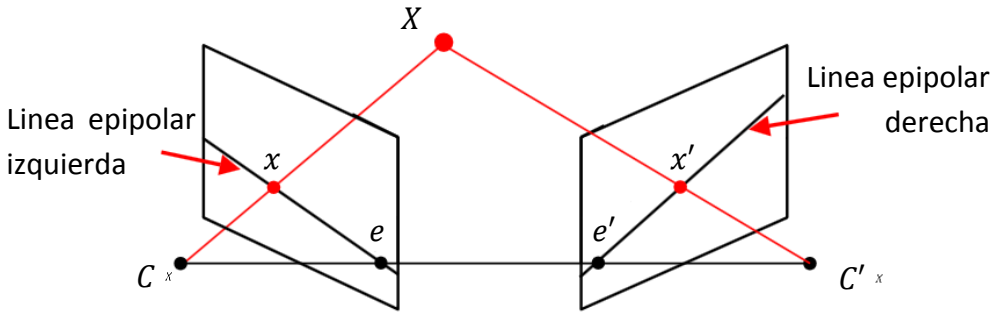


Ilustración 7: Geometría epipolar en una configuración de cámaras no paralela.

Las líneas epipolares generalmente son no paralelas. En el ejemplo mostrado en la Ilustración 8 se realiza la fotografía de un florero utilizando la configuración no paralela de cámaras.

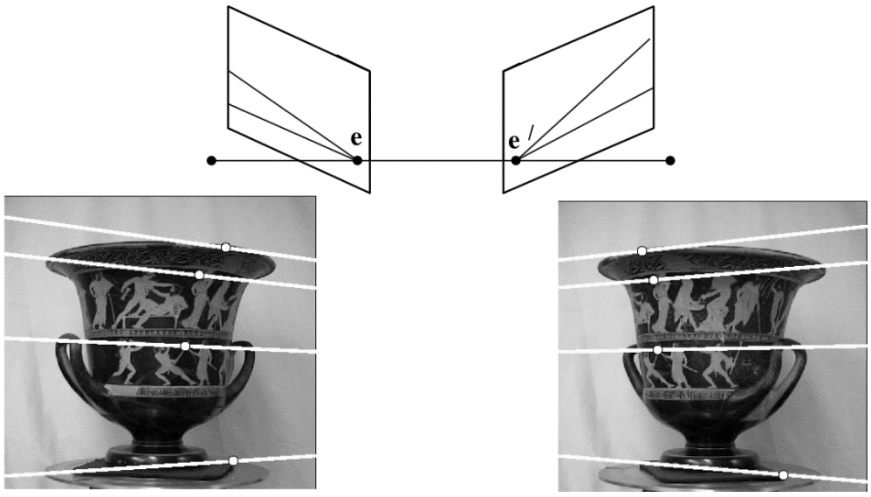


Ilustración 8: Ejemplo de líneas epipolares para una configuración de cámaras no paralelas. Fuente de la imagen: cs.texas.edu [6].

Realizando una calibración de la cámara, se encuentran los epipolos, los que describirán todas las líneas epipolares provenientes de estas. Una vez conocidas estas líneas, la búsqueda de la pareja de cada punto en la otra imagen se realiza solo en la línea epipolar en vez de en toda la imagen, reduciendo considerablemente el tiempo de cómputo del algoritmo de búsqueda.

2.4.2 Modelo de Cámara Paralelo

El modelo paralelo es un caso particular del anterior, donde el ángulo entre las cámaras es cero, y los epipolos están en el infinito hacia los lados. Las líneas epipolares quedan

horizontales y exactamente paralelas en ambas imágenes. La búsqueda de parejas resulta mucho más sencilla en este caso.

En la Ilustración 9 se muestra el modelo geométrico de cámaras paralelas. Las coordenadas x_l y x_r en el eje X corresponden al punto p proyectado en los planos de imagen.

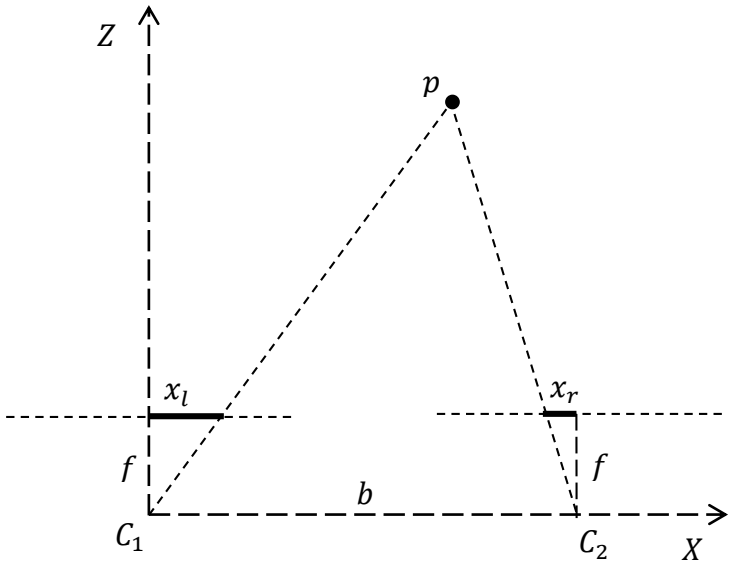


Ilustración 9: Geometría de cámaras paralelas.

La profundidad del punto p queda dada por la ecuación:

$$p_z = \frac{b \cdot f}{x_l - x_r}, \tag{4}$$

donde b es la distancia entre las camaras, y $x_l - x_r$ es la disparidad del punto proyectado, es decir, la diferencia entre las coordenadas en la imagen izquierda y derecha. A mayor disparidad el punto estará más lejos en el eje Z, mientras que si la disparidad es cercana a cero, el objeto se encontrara más lejos de la cámara. No puede existir disparidad negativa pues significaría que el objeto está detrás del arreglo de cámaras. En último caso representa un error en el emparejamiento de características.

De las ecuaciones (2) y (3) podemos extraer las coordenadas p_x y p_y de la proyección en la imagen.

$$p_x = x \cdot \frac{p_z}{f} \tag{5}$$

$$p_y = y \cdot \frac{p_z}{f}. \tag{6}$$

Cabe destacar que las coordenadas x e y normalmente se manejan en pixeles, pero para las ecuaciones expresadas anteriormente estas deben estar expresadas en milímetros. Para pasar de pixeles a milímetros es necesaria hacer la siguiente conversión:

$$x_{mm} = (x_{pixel} - C_x) \cdot S_x \,, \tag{7}$$

$$y_{mm} = (y_{pixel} - C_y) \cdot S_y \,, \tag{8}$$

donde C_x y C_y representa el punto central de la cámara (normalmente la mitad de la imagen, pero no siempre es así), y S_x y S_y el tamaño horizontal y vertical de cada pixel en la cámara (usualmente expresados en μm).

2.5 Estimación de Movimiento

Este paso es el fundamental para la odometría visual y consiste en el cálculo del movimiento de la cámara entre dos imágenes sucesivas. Se construye el movimiento T_k entre la imagen actual y la anterior, cada vez que llega una imagen nueva de la cámara.

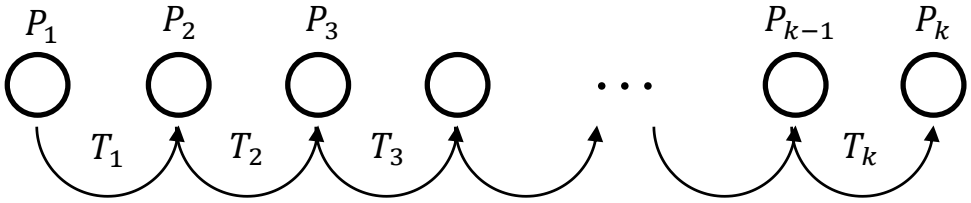


Ilustración 10: Relación entre la matriz de transformación y la posición de la cámara, de fotograma en fotograma.

Donde T_k es la matriz de movimiento de 4x4:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}, \tag{9}$$

con $R_{k,k-1}$ la matriz de rotación de 3x3 entre el momento k y el antecesor $k-1$, y $t_{k,k-1}$ el vector de traslación de 3x1 entre el momento k y $k-1$. La posición actual de la cámara, y por lo tanto la posición del vehículo, queda dada entonces por:

$$P_k = T_k P_{k-1} \,, \tag{10}$$

con P_0 siendo la posición inicial de la cámara, en coordenadas homogéneas. Para el cálculo de R y t se utiliza un algoritmo que utiliza una descomposición en valores singulares (*SVD*, por sus siglas en inglés) [7]. Otras soluciones se conocen en la literatura, como la basada en la matriz esencial [8], o la de máxima similitud [9] utilizada en los robot Spirit y Opportunity en las misiones a Marte [10], sin embargo no se exploran en este trabajo.

2.5.1 Solución basada en SVD

Una vez conocidos dos grupos de características en imágenes consecutivas, y realizada la triangulación de estos, es posible calcular el movimiento que ha tenido la cámara entre estas dos imágenes usando el algoritmo descrito a continuación.

Sean $\{X\}_n$ e $\{Y\}_n$ los grupos de puntos 3D correspondientes entre sí, para a las imágenes antes y después del movimiento. La idea detrás de esta solución es separar la rotación de la traslación. Sean μ_x y μ_y los centroides de los grupos de puntos $\{X\}_n$ e $\{Y\}_n$ respectivamente, que se calculan como el valor promedio de la posición de los puntos, en cada grupo.

Bajo la suposición de que los puntos encontrados pertenecen a un entorno estático, entonces el valor de las coordenadas de cada punto X_i relativo a μ_x, y Y_i relativo a μ_y deben mantenerse antes y después del movimiento. Es decir, se mueven como un cuerpo rígido. Es posible entonces separar el problema en dos partes:

1. Encontrar R que minimice la función de error

$$\epsilon^2 = \frac{1}{n}\sum_{i=1}^{n} \|Y_i - RX_i\|^2 \tag{11}$$

2. Calcular la traslación t, que está dada por $t = \mu_y - R\mu_x$.

Para minimizar la función de error la ecuación se la descomposición en SVD de la matriz de covarianza Σ_{xy}, como se menciona en [11].

Los centroides quedan serán el valor promedio de los puntos de cada grupo:

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} X_i\,, \tag{12}$$

$$\mu_y = \frac{1}{n}\sum_{i=1}^{n} Y_i\,. \tag{13}$$

La matriz de covarianza Σ_{xy} queda dada por:

$$\Sigma_{xy} = \frac{1}{n}\sum_{i=1}^{n} (Y_i - \mu_y)(X_i - \mu_x)^T\,. \tag{14}$$

Sea UDV^T la descomposición SVD de la matriz de covarianza Σ_{xy}, es decir $SVD(\Sigma_{xy})$. Con $(D = diag(d^i), d^1 \geq d^2 \geq \cdots \geq d^m \geq 0)$, y sea la matriz S dada por:

$$S = \begin{cases} I & \det(U) \cdot \det(V) \geq 0 \\ diag(1,1,\dots,1,-1) & \det(U) \cdot \det(V) < 0\,. \end{cases} \quad (15)$$

La corrección de S queda por el caso degenerado de reflexión, donde la solución queda definida para un set de puntos detrás de la cámara. Entonces se tiene que la matriz de rotación R queda dada por:

$$R = USV^T. \quad (16)$$

Finalmente,

$$t = \mu_y - R\mu_x\,. \quad (17)$$

Con estos resultados se compone la matriz de transformación T, que queda descrita en la Ecuación (9).

2.6 Rechazo de O*utliers*

Los puntos emparejados entre las imágenes normalmente vienen contaminados por asociaciones incorrectas, llamadas *outliers.* Estas malas asociaciones repercuten fuertemente en el desempeño final del algoritmo, contribuyendo en conjunto a una desviación en una dirección aleatoria de la posición calculada de la cámara.

Existen múltiples fuentes que contribuyen a la emergencia de *outliers:*

- Ruido en imágenes
- Imagen desenfocada (ya sea por movimiento de la cámara, o fuera de foco)
- Oclusiones
- Cambios en iluminación

Existen varios algoritmos de remoción de o*utliers* que se diferencian en el grado de error final presente en las muestras, y en la rapidez del algoritmo. Uno de las técnicas más utilizadas es RANSAC.

En la Ilustración 11 se muestra la comparación entre la trayectoria de un vehículo calculada con odometría visual utilizando técnicas de remoción de *outliers* (RANSAC y *adaptive bucketing*), en rojo, y sin remoción, en azul. Se muestra una desviación considerable en la recta final de la trayectoria. Este resultado es bastante favorable, pues por lo general la desviación de trayectoria es en la totalidad del trayecto.

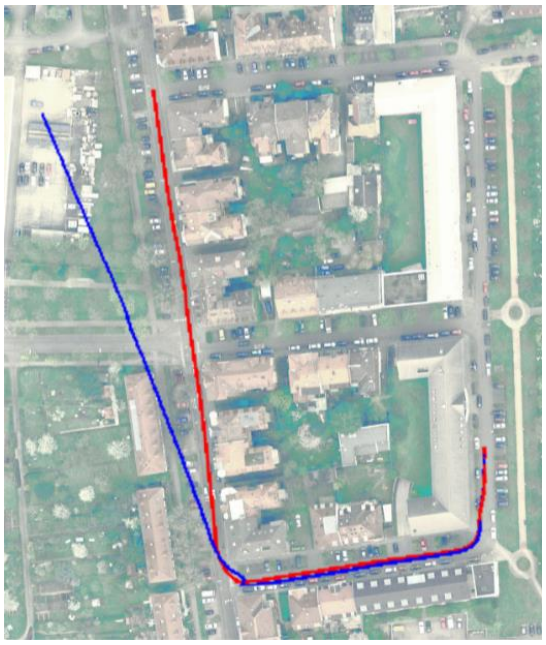


Ilustración 11: Comparación de trayectorias con (azul), y sin (rojo) la presencia de *outliers*. Imagen extraída de [12].

Otra técnica, propuesta en [13], está basada en restricciones geométricas entre las parejas de puntos $\{X\}_n$ e $\{Y\}_n$. Considerando un ambiente estático, es decir, sin objetos en movimiento, la distancia relativa entre dos puntos en cada grupo es igual antes y después de la transformación, es decir:

$$\|X_i - X_j\| = \|Y_i - Y_j\|. \tag{18}$$

Además,

$$(X_i - X_j)(Y_i - Y_j) > \cos(\theta), \tag{19}$$

donde θ se eligió $\frac{\pi}{4}$. Sin embargo la Ecuación (18) no puede ser satisfecha pues la igualdad no puede ser alcanzada dado el ruido en el cálculo de las coordenadas 3D de los puntos, y la inexactitud en la matemática de punto flotante presente en un entorno computacional.

2.6.1 RANASAC

En el algoritmo de RANSAC (*random sample consensus*) es un algoritmo iterativo en donde la idea es tomar una serie de n puntos al azar y calcular a través de estos un modelo que será la hipótesis de la iteración. Luego se aplica el modelo a la totalidad de los puntos y se suman los errores obtenidos. El algoritmo itera sobre distintos grupos de datos y se elige la solución en que la mayor cantidad de datos se ajusten al modelo calculado, para un número total N de iteraciones.

En la Ilustración 12 se observan dos iteraciones distintas para ajustar una recta (modelo) a un set de datos. Sea la iteración de la izquierda, iteración 1, y la iteración de la derecha,

iteración 2. Se observa en línea punteada el margen para el cual los puntos al interior se consideran como *inliers*, pintados de color rojo. En la iteración 1 se observa que hay solo 5 *inliers*, mientras que en la iteración 2 hay 14. Esto quiere decir que el modelo encontrado en la iteración 2 es el modelo más correcto de todos los encontrados.

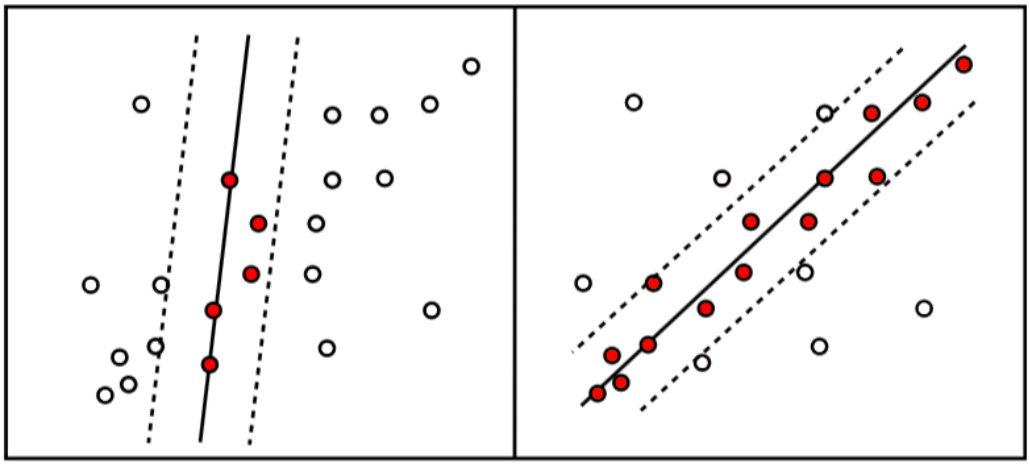


Ilustración 12: Comparación de dos iteraciones del algoritmo RANSAC.

El número de iteraciones N necesarias para garantizar que una solución correcta con cierta probabilidad se ha encontrado está dado por [14]:

$$N = \frac{\log(1-P)}{\log(1-(1-\epsilon)^s)} \tag{20}$$

Donde S es el número de datos mínimo requeridos para la calcular el modelo, ϵ es el porcentaje de *outliers* en los datos, y P es la probabilidad de éxito deseada. En la práctica el número N se multiplica por un factor de 10 para maximizar la robustez del algoritmo.

Como la elección de puntos es al azar, el comportamiento del algorítmico es no determinístico, es decir, en dos ejecuciones distintas del algoritmo, se obtendrán 2 diferentes resultados.

Para el caso de odometría visual, el modelo estimado sería el movimiento relativo $(R,\ t)$ entre dos posiciones sucesivas de la cámara. Para estimar la distancia de los puntos (X, Y) al modelo se calcula el error RMS entre $T(X)$ e Y. Consideramos al punto como *outlier*, si el error RMS de éste se encuentra fuera de un margen $[-d, +d]$, con d un numero arbitrario que será encontrado mediante calibración.

2.7 Ecualización de Histograma

El histograma de una imagen es la representación gráfica de la distribución de la intensidad de los pixeles de ésta, cuantificando el número de pixeles para cada valor de intensidad de 0 a 255.

Una ecualización de histograma es un método para mejorar el contraste de la imagen en el cual se analiza la distribución acumulada de datos y se convierte en una distribución recta.

En la Ilustración 13 se observa una imagen en escala de grises a la cual se le ha calculado su histograma, ubicado a su derecha. Se observa en la imagen superior que el histograma esta acumulado solo en la sección media, lo que significa que la tonalidad media de la imagen es gris, y por lo tanto no existe mucho contraste. En color negro se observa la distribución acumulada del histograma, en la que se observa también que el incremento se realiza en la zona gris.

Al aplicar la ecualización de histograma, se transforma la distribución acumulada a una recta que comienza en el negro y termina en el blanco, abarcando así toda la escala de colores. El resultado es una imagen con mayor contraste, donde el pico del histograma ha sido ensanchado para ocupar una mayor franja de colores.

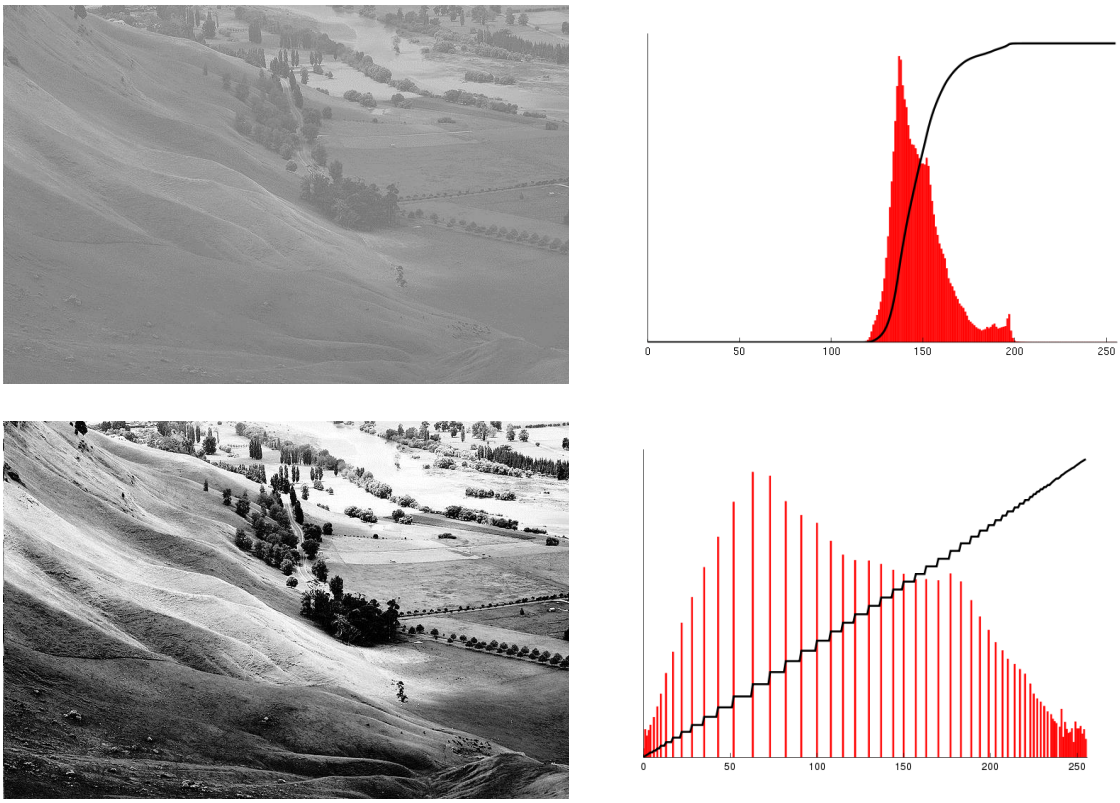


Ilustración 13: Imágenes en escala de grises junto a su histograma. Se observa una mejora en contraste para la imagen inferior, a la cual se le ha ecualizado el histograma. Autor de la fotografía: Phillip Capper. Autor de histogramas: Jarekt, usuario de Wikipedia.

Capítulo 3: Implementación

3.1 Estructuras de datos básicas

En esta sección se presentan algunas de las estructuras de datos fundamentales utilizadas en este trabajo, para el lenguaje de C++. Esto con el objetivo de resolver dudas al momento de leer los diagramas UML contenidos en este capítulo.

Tabla 1: Estructuras de datos básicas en la implementación del código, su descripción y su librería origen.

Estructura	Descripción	Origen
vector<A>	Estructura que agrupa objetos de tipo A en un arreglo ordenado	Standard Template Library (desde ahora, STL)
Mat	Matriz o vector de dimensión variable, con 1, 2 o 3 canales por elemento. Puede ser utilizado en operaciones matemáticas de manera sencilla a través de los operadores básicos (*,/,+,-).	OpenCV
Point3f	Vector de 3 dimensiones de tipo de dato float	OpenCV
Map<A,B>	Tipo de dato que implementa una función de Hash con llave tipo A, para acceder a una estructura tipo B	STL
Keypoint	Estructura básica que encapsula los parámetros básicos de una característica para el algoritmo surf, como el ángulo y el tamaño	OpenCV
DMatch	Estructura que guarda la información acerca del emparejamiento de una característica, específicamente la similitud y los índices de las características pareadas.	OpenCV

3.2 Modelo sistema

El sistema programado en este trabajo se compone de distintas clases de datos o módulos, que interactúan entre sí a través de un *pipeline* que tendrá como resultado la posición del robot para cada imagen que provenga de la cámara.

El objeto *VOMachine* será el que dirija todo el algoritmo, y llevara la cuenta de todos los datos adquiridos en cada iteración del ciclo principal, como valores de posición, y tiempos de ejecución para cada etapa: triangulación y estimación de movimiento.

Para configurar a *VOMachine*, se debe suministrar una implementación de algoritmo de odometría visual. El algoritmo implementado en este trabajo es el 3D3D, el cual se suministra en la clase *VO3D3D*. A su vez, esta clase debe recibir una implementación del algoritmo de detección y del algoritmo de detección, y la clase *Camera*. Los algoritmos utilizados son SURF y FLANN, suministrados por la librería OpenCV.

Como se menciona anteriormente, la ejecución del algoritmo se divide en la etapa de triangulación y la de estimación de movimiento. En la etapa de triangulación, cada par de imagen suministrada por la cámara, I_{izq} e I_{der} pasa al algoritmo de detección de características, donde se encontraran sus puntos clave y sus descriptores. Luego se pasa el algoritmo FLANN para realizar el emparejamiento de características encontradas en ambas imágenes. Finalmente se guardan los resultados en la estructura *StereoImage*, la cual a su vez realiza el algoritmo de triangulación. Una vez triangulados los puntos, se inicia la siguiente etapa.

En la etapa de estimación de movimiento se utilizan las dos últimas imágenes que han pasado por el proceso de triangulación, se realiza un nuevo proceso de emparejamiento entre ambas imágenes, esta vez con sus puntos ya triangulados. Los puntos emparejados son finalmente derivados al algoritmo RANSAC, el que se encargara de encontrar la mejor solución, dado los parámetros configurados. Finalmente se guardan los resultados en un archivo de texto, para ser graficados posteriormente en Excel.

Para mayor detalle, se incorpora el diagrama UML completo del sistema, el cual se encuentra en el Anexo A, en la página 78.

3.3 Cámara

El robot está equipado con una cámara estereoscópica en su parte superior. La cámara utilizada es una BumbleBee XB3 de Point Grey, y está equipada con tres sensores de imagen: dos a cada costado y uno al centro. En este trabajo solo serán utilizadas las cámaras laterales. Específicamente se utiliza el modelo BBX3-13S2C-38, que corresponde al modelo de 1.3MP, 3.8mm de distancia focal, a color.

Está orientada hacia el frente con una inclinación hacia debajo de 15º, de manera la mitad inferior de la imagen muestra el suelo, y la mitad superior muestra el frente. Se observa en la Ilustración 14 una imagen de la cámara, así como también sus dimensiones principales de la vista frontal. Dada la enorme cantidad de datos que provienen de las 3 cámaras de alta resolución, se utiliza una interfaz doble FireWire1394 de 800mbps.

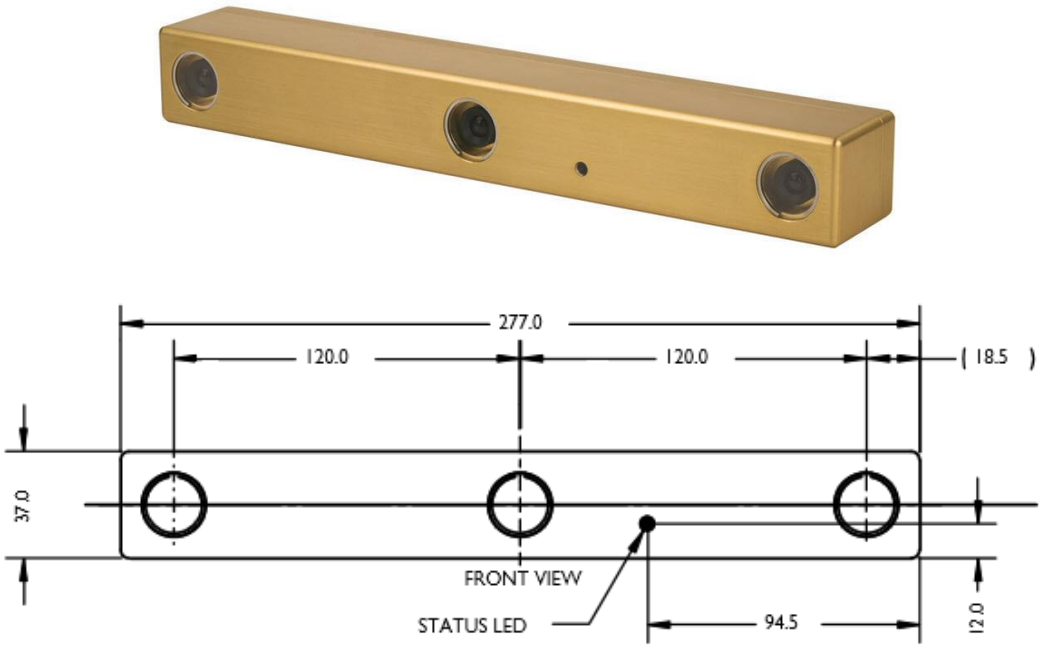


Ilustración 14: Vista general de la cámara del robot, y sus dimensiones frontales. Fuente de la imagen: www.ptgrey.com.

Hay un grupo de parámetros clave que se deben considerar en la aplicación del algoritmo de triangulación. En la Sección 2.4 de triangulación se mostraron una serie de fórmulas que requieren de estos parámetros. En las Ecuaciones (4), (5) y (6) en particular requieren de la distancia entre las cámaras b y la distancia focal f. En las Ecuaciones (7) y (8) se referencian C_x y C_y, valores para el punto central de la cámara, y S_x, S_y, ancho y alto del pixel. En la Tabla 2 se muestran los valores de dichos parámetros junto a otros parámetros importantes.

Tabla 2: Valores de parámetros importantes de la cámara.

Parámetro	Nombre	Valor
b	Distancia entre cámaras	$240mm$
f	Distancia focal	$3.8mm$
C_x	Coordenada central de la imagen en x	$640px$
C_y	Coordenada central de la imagen en y	$480px$
S_x y S_y	Tamaño del pixel	$3.75\mu m$
α	Angulo de inclinación	15º
(R_x, R_y)	Resolución de la cámara	$(1280x960)$
V	Velocidad de adquisición	$16FPS$

Para una estimación más precisa de la posición del vehículo, es importante utilizar los parámetros calibrados de la cámara en lugar de utilizar los valores de fábrica. Esta

calibración se hace mediante un tablero de recuadros en forma de ajedrez, siguiendo el protocolo estándar de calibración. El resultado final de la calibración da a conocer diversas constantes de compensación de distorsión radial y tangencial para ambas cámaras, distancias focales f_x y f_y, y puntos principales C_x y C_y en ambos ejes. El proceso utilizado para extraer las imágenes del archivo .bag en ROS ya aplica la corrección de distorsión a los datos. Los parámetros restantes forman lo que se llama la matriz de cámara (*camera matrix*) y reemplazan a los datos de la Tabla 2 en las formulas (5), (6), (7) y (8). Para esta cámara en particular, los parámetros obtenidos son:

Tabla 3: Valores calibrados que conforman la matriz de camara.

Parámetro	Nombre	Valor
b	Distancia entre cámaras	$239.60mm$
f_x	Distancia Focal Eje x	$3.7393mm$
f_y	Distancia Focal Eje y	$3.7393mm$
C_x	Coordenada central de la imagen en x	$667.57px$
C_y	Coordenada central de la imagen en y	$496.31px$

Estos parámetros quedan representados en código baje las estructuras *Camera* y *CameraProperties*.

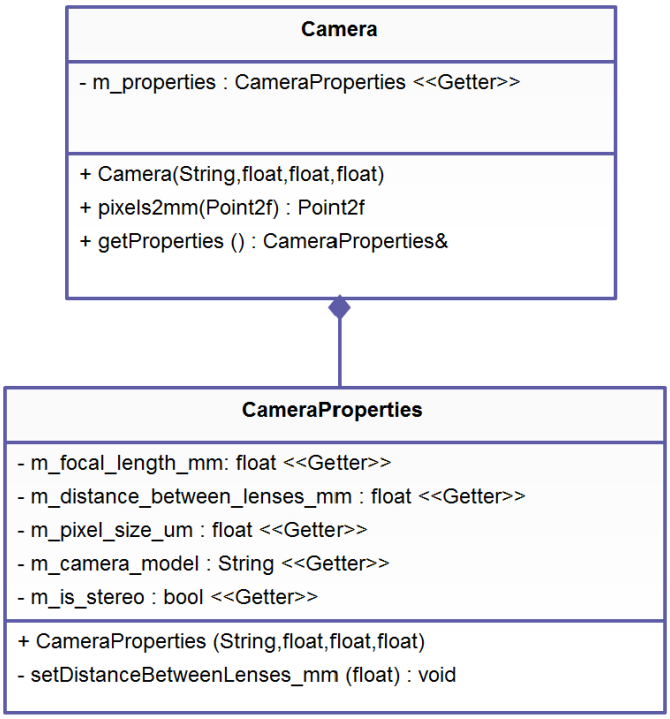


Ilustración 15: Diagrama UML de las estructuras *Camera* y *CameraProperties*.

3.4 Triangulación

En el sistema construido, el algoritmo de triangulación está contenido en la clase *StereoImage*. Esta clase hace el nexo entre todos los resultados importantes de los distintos algoritmos que hayan sido aplicados a la imagen estereoscópica.

En la Ilustración 16 se observa el diagrama UML de la estructura antes mencionada. Podemos ver que ésta extiende la clase *CameraImage*, y contiene a la clase *ImageFeatures* a través de los miembros *m_features_l* y *m_features_r*. Estos dos últimos contienen las características de las imágenes izquierda y derecha respectivamente, así como también los descriptores asociados.

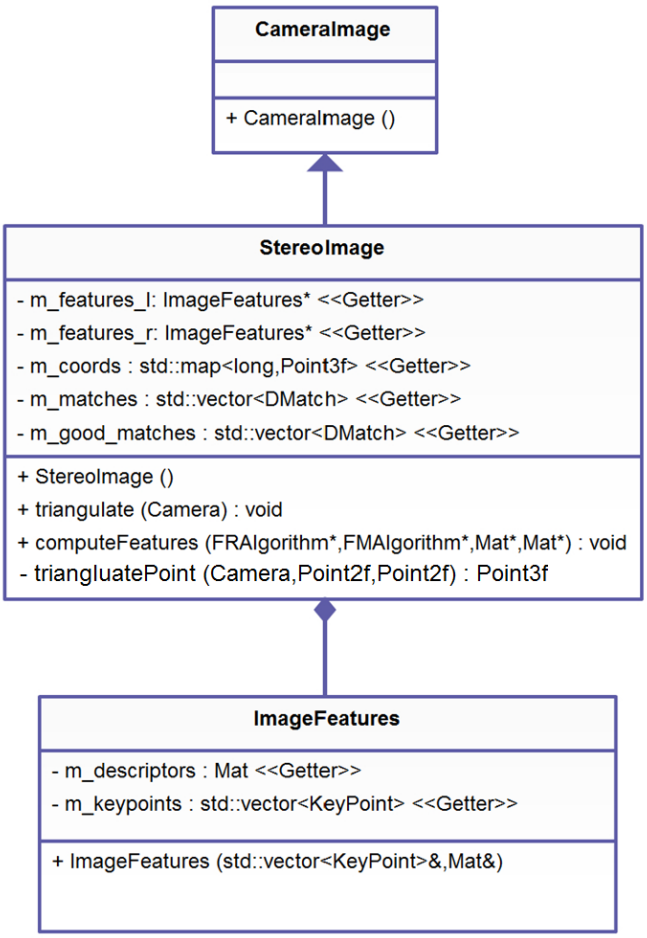


Ilustración 16: Diagrama UML de la estructura *CameraImage*, *StereoImage*, e *ImageFeatures*.

La triangulación se realiza mediante la función *triangulate*, que recibe como parámetro la estructura *Camera*, la cual se detalla en la Sección 3.3, en la Ilustración 15. En esta función se implementan las ecuaciones (4), (5) y (6), presentes en la Sección 2.4.2. El diagrama de flujo de la función se muestra en la figura

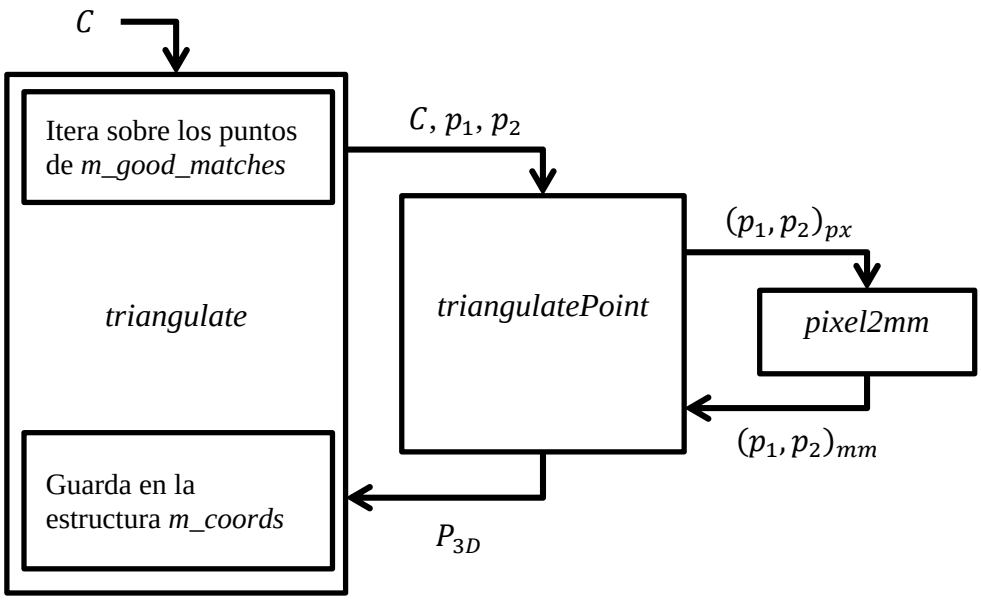


Ilustración 17: Diagrama de ejecución de la función *triangulate*.

Para determinar que porción de la imagen contiene mejores muestras de características, se realizan pruebas independientes en la mitad superior y la inferior de la imagen de la cámara. Para determinar si las características son mejores o peores, se analizan los resultados en la estimación de movimiento en la zona cercana de la imagen y en la zona lejana.

Se realiza un filtro de puntos dado por la ubicación en el eje Z de este. Para esto se ponen como límite inferior 0.5m, y como límite superior 100m, y limite central 30m. Para esto, se convierten los valores de desplazamiento horizontal de puntos $d = (x_1 - x_2)$ expresados en milímetros, a valores expresados en pixeles. Los valores se despejan de la Ecuación (4):

$$d_{mm} = \frac{(f \cdot d)}{p_z} \cdot 0.001 \,,$$

$$d_{mm} = \frac{(3.8 \cdot 240)}{p_z} \cdot 0.001 \,. \qquad (21)$$

Obtenemos los valores de d_{mm} para los valores de profundidad correspondientes a 0.5m, 30m y 100m. Luego se despeja el valor e pixeles de la Ecuación (7):

Convirtiendo los valores a pixeles

$$d_{mm} = \left(\left(x_{1_{px}} - C_x\right) - \left(x_{2_{px}} - C_x\right)\right) \cdot 0.00375$$

$$d_{mm} = (x_{1_{px}} - x_{2_{px}}) \cdot 0.00375$$

$$d_{px} = \frac{d_{mm}}{0.00375} \tag{22}$$

Los valores finales se muestran en la Tabla 4.

Tabla 4: Conversión de valores de profundidad a desplazamiento horizontal.

P_z	d_{mm}	d_{px}
0.5m	1.8240mm	486.40px
30m	0.0304mm	8.11px
100m	0.00912mm	2.43px

3.5 RANSAC

Se programó un algoritmo de RANSAC de 5 puntos para encontrar la matriz de trasformación T. El número de iteraciones utilizadas N se calcula de la Ecuación (20) usando como parámetros a priori: $P = 0.9$, $\epsilon = 0.5$, para $S = 5$.

$$N = \frac{\log(1 - 0.9)}{\log(1 - (1 - 0.5)^5)} = 72 \tag{23}$$

Multiplicamos por un factor de 10 para agregar robustez, lo que da un número de iteraciones $N = 720$. La influencia del número de iteraciones se comprobara experimentalmente.

Con respecto a la implementación en código de RANSAC en código, en la Ilustración 18 se observa el diagrama en UML de la clase RANSAC. La clase está dividida en tres funciones principales, de las cuales solo una es pública, y el resto es de acceso exclusivo a la propia clase.

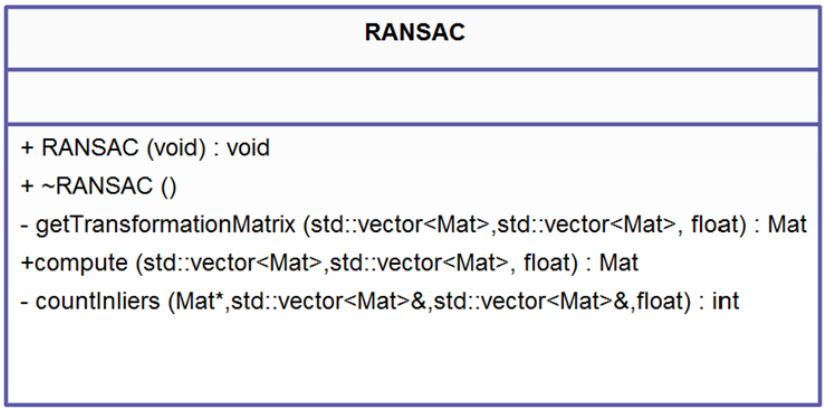


Ilustración 18: Diagrama UML de la clase RANSAC.

La función pública *compute* recibe como parámetros el listado de puntos $\{X\}_i$ provenientes de I_{k-1}, e $\{Y\}_i$ proveniente de I_k y el valor de distancia mínima para la cual un punto es considerado *inlier* (o *outlier*). Esta función es la que contiene el ciclo principal del algoritmo, en la cual se ejecutan las llamadas a las funciones auxiliares *getTransformationMatrix* y *countinliers*.

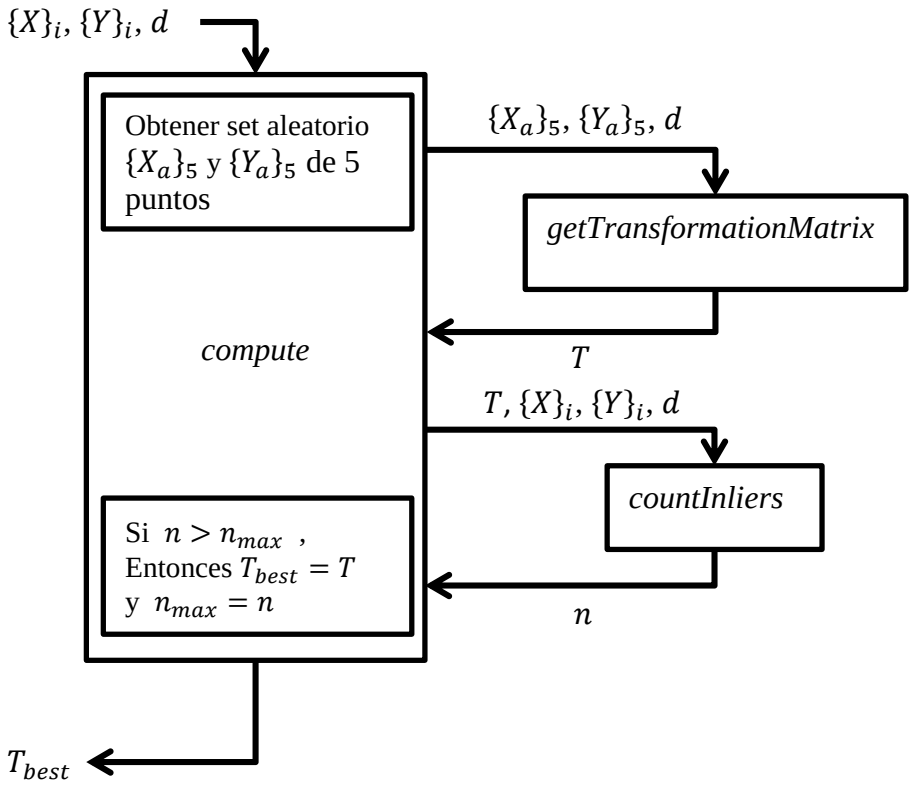


Ilustración 19: Diagrama de flujo del algoritmo RANSAC, como fue implementado en el código.

En la Ilustración 19 se observa el diagrama de flujo del algoritmo. En el comienzo del ciclo se obtiene un subconjunto aleatorio de 5 datos de los vectores $\{X\}_i$ e $\{Y\}_i$. Luego se ejecuta la función *getTransformationMatrix* con estos datos más la distancia d, la que arroja la matriz T como resultado. Posteriormente se ejecuta la función *countInliers* con los datos iniciales, más la matriz T adquirida en el paso anterior, la que devuelve el número de *inliers*. Finalmente, la función *com*pute guarda la matriz T correspondiente a la iteración en la que se obtuvo mayor cantidad de *inliers*, en el diagrama denotado como n_{max}.

3.6 Set de datos

Para probar el algoritmo, se utilizarán sets de datos grabado anteriormente. Estos datos corresponden al movimiento del robot a través de un túnel en la mina El Teniente. Los archivos de datos fueron recolectados utilizando ROS bajo el sistema operativo Linux, y se encuentran en un archivo .bag, que almacena datos de imágenes tomadas por las cámaras, tiempos asignados a cada imagen Se recolectaron en total 2 archivos, correspondientes a

dos tramos consecutivos del mismo túnel con condiciones distintas de luminosidad, como se muestra en la Ilustración 20.

Ilustración 20: Muestra de dos imágenes correspondientes a tramos distintos.

Se llamara al tramo oscuro Tr_1 y al tramo luminoso Tr_2. Se conoce la posición del robot en el inicio y termino de cada tramo, los que se muestran en la Tabla 5. La distancia recorrida se muestra en la Tabla 6. Se observa que el robot recorre una distancia aproximadamente recta, con una inclinación hacia la izquierda.

Tabla 5: Posiciones conocidas para los tramos recorridos.

Tramo	$\boldsymbol{X[m]}$	$\boldsymbol{Y[m]}$	$\boldsymbol{Z[m]}$	**Pitch[º]**	**Yaw[º]**	**Roll[º]**
Tr_1 inicio	64.73	1.08	-201.26	0.88	184.99	2.38
Tr_1 final Tr_2 inicio	63.44	1.12	-144.68	0.53	182.47	2.97
Tr_2 final	62.91	1.13	-110.76	0.73	177.64	2.38

Tabla 6: Diferencia de posiciones iniciales y finales para ambos tramos.

Tramo	$\boldsymbol{\Delta X[m]}$	$\boldsymbol{\Delta Y[m]}$	$\boldsymbol{\Delta Z[m]}$	**ΔPitch[º]**	**ΔYaw[º]**	**ΔRoll[º]**
Tr_1	-1.29	0.04	56.58	-0.35	-2.52	0.58
Tr_2	-0.52	0.01	33.93	0.20	-4.82	-0.59

Capítulo 4: Análisis de Resultados

Los cálculos computacionales se realizaron en un Servidor HP Proliant ML350 G6, cuyas especificaciones se muestran en la Tabla 7.

Tabla 7: Especificaciones del Servidor HP Proliant ML350 G6.

Procesador 1	Intel Xeon X5670 @ 2.93GHz
Procesador 2	Intel Xeon X5670 @ 2.93GHz
Núcleos	12 físicos, 12 virtuales
Memoria RAM	12GB
GPU	ATI ES 1000
Sistema Operativo	Windows Server 2008 R2 Enterprise 64-bit

4.1 Triangulación de Características

En la Ilustración 21 se observa una imagen a la cual se le ha calculado la triangulación a todas las características encontradas. Los rombos en verde corresponden a puntos en los que su coordenada Z es positiva, es decir, se encuentran al frente de la cámara. Los rombos rojos corresponden a puntos con coordenada Z negativa, es decir están detrás de la cámara. Es claro que cualquier punto visible en la imagen no puede estar detrás de la cámara, por lo que estos puntos deben ser errores de emparejamiento. Dado que el arreglo de cámaras es paralelo, se aplica un filtro a las parejas de puntos encontradas en el cual se eliminan puntos que tengan una diferencia en sus coordenadas y mayor que 4 pixeles. Los resultados de dicho filtro se observan en la imagen de la derecha.

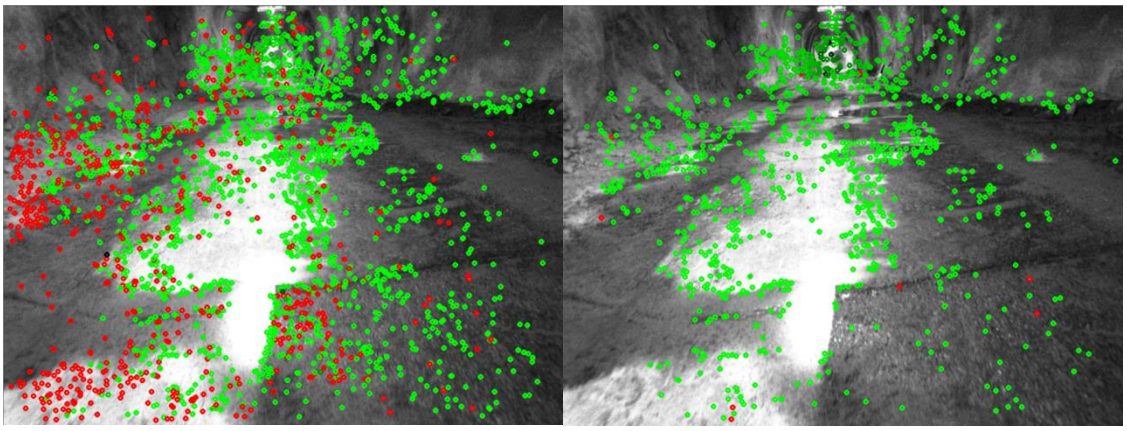


Ilustración 21: Rechazo de outliers en triangulación por observación de la coordenada y.

Se observa una mejora considerable en la remoción de *ouliers* para la etapa de triangulación, con un tiempo de computo de $O(n)$ para el número de parejas encontradas.

Para orientar al lector acerca de la escala a la cual se está trabajando, la Ilustración 22 muestra la coordenada Z para algunos puntos en el campo visual. Puntos a la lejanía están

ubicados alrededor de los 25 metros. Puntos que superan los 25m se encuentran apelotonados en la parte central superior.

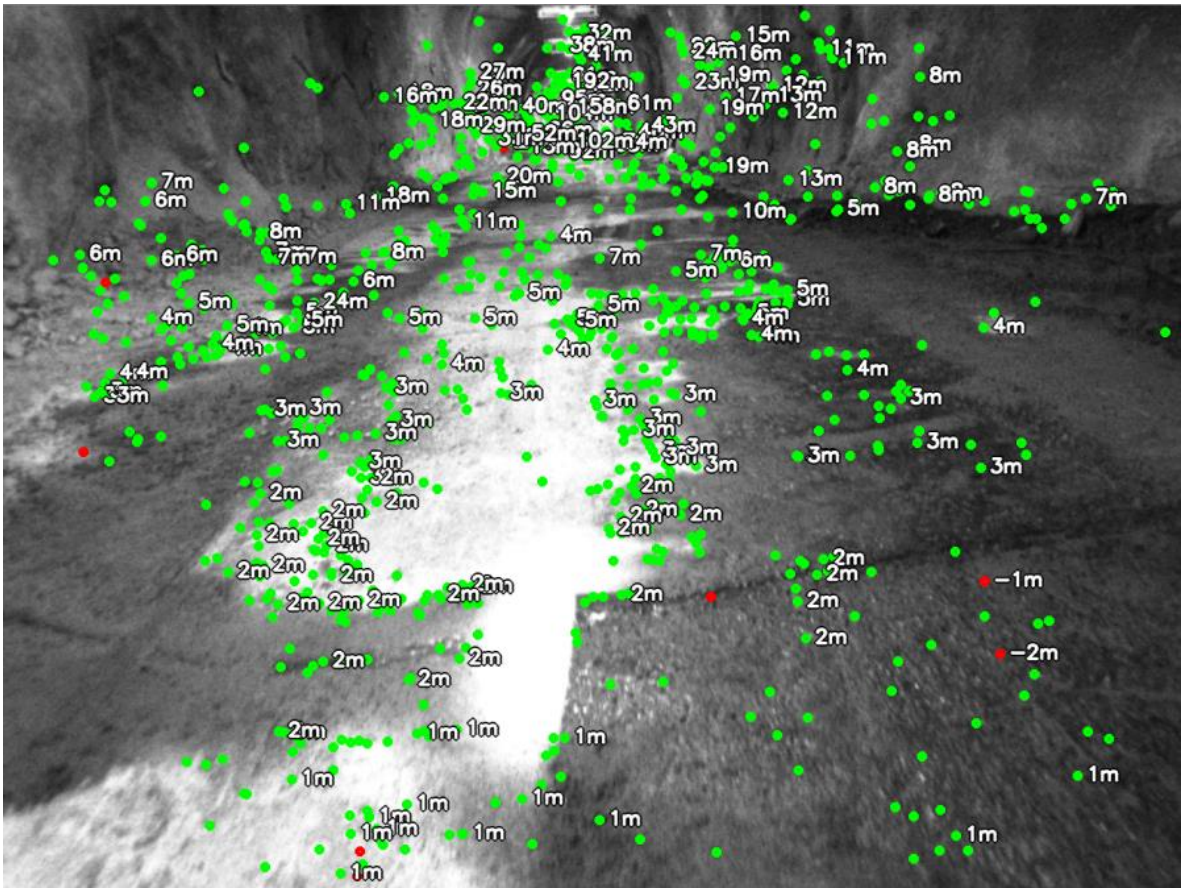


Ilustración 22: Coordenada Z para la triangulación de un grupo de datos.

Se observa también que las características detectadas se encuentran en zonas donde el gradiente del cambio de intensidad de la luz es grande, como en los bordes de las zonas iluminadas. Este resultado se hace más evidente al analizar una imagen del tramo oscuro del túnel, como se muestra en la Ilustración 23. En este caso se observa un vacío de detección en la zona no iluminada de la imagen.

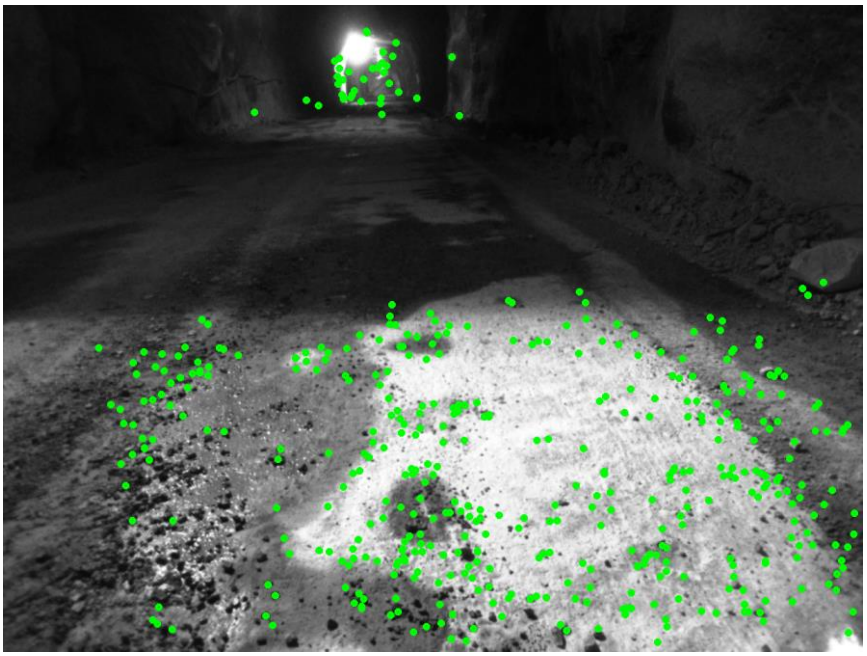


Ilustración 23: Características detectadas en el tramo oscuro del túnel.

4.2 Detección de Características y Emparejamiento

En la Ilustración 24 se observa el *matching* de caracteristicas para dos imágenes consecutivas I_{k-1} e I_k. La parte derecha de la imagen ha sido cortada por razones de espacio en la pagina, sin embargo no es necesaria para comprender lo que esta sucediendo en este proceso. Cada caracteristica esta denotada visualmente por un rombo de color y una linea horizontal hacia la derecha. Los rombos de color rojo denotan puntos donde se conoce la posicion en 3 dimensiones de la caracteristica en ambas imagenes, es decir, han sido trianguladas en una instancia anterior.

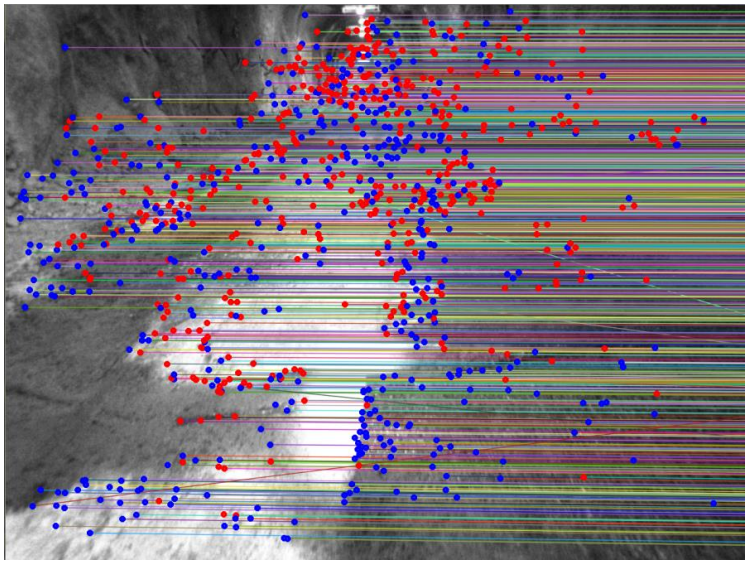


Ilustración 24: *Matching* para dos imágenes tomadas en instantes consecutivos.

Los rombos de color azul representan puntos en los cuales se conoce la información de en 3 dimensiones, pero solo en una imagen, por lo que no es apto para utilizar en la etapa posterior del algoritmo. Dado que para calcular el movimiento solo son utiles las caracteristicas con rombo rojo la otra información se pierde y al final es tiempo de computo perdido.

4.3 Resultados con RANSAC

4.3.1 Comparación de del número de iteraciones N

En esta sección se muestran los resultados de aplicar el algoritmo RANSAC en 50 instancias diferentes para distinto número de iteraciones. Estos resultados son distintos unos de otros dada la naturaleza no determinista del algoritmo. Los parámetros utilizados se muestran en la Tabla 8.

Tabla 8: Parámetros de prueba del algoritmo RANSAC.

Parámetro	Valor
Núm. de puntos	RANSAC-5
d	0.07
N	12,25,50,100,200,400,800,1600

Primeramente se presentan gráficos de media y desviación estándar, mostrando la trayectoria del robot y su orientación, en las Ilustración 25, Ilustración 26, Ilustración 28, Ilustración 29, Ilustración 30, Ilustración 31, Ilustración 32, Ilustración 33 e Ilustración 33 para la sección oscura del túnel, y la Ilustración 38 para la sección iluminada del túnel. Luego se comparan las desviaciones estándar para todos los valores de N, en las Ilustración 34 Ilustración 35 para la sección oscura, y la Ilustración 39 para la iluminada. Posteriormente se presenta un gráfico de medias para los 6 grados de libertad del robot en la Ilustración 36 para la sección oscura, y la Ilustración 46 para la sección iluminada. Finalmente se presentan los gráficos de tiempo del algoritmo en la Ilustración 37 para la sección oscura, y la Ilustración 47 para la sección iluminada.

En la Ilustración 25 e Ilustración 26 se muestra la comparación de trayectorias para distinto número N de iteraciones de RANSAC, de 12 a 1600, para la sección oscura del túnel. En estos gráficos se encuentra la información de 50 realizaciones del algoritmo, representados por la media (línea negra) y la desviación estándar (línea roja). El eje X corresponde a la coordenada X del robot en el túnel (izquierda y derecha), y el eje Y representa la coordenada Z (profundidad). Esto corresponde a una vista superior del robot. El ancho medio del túnel es de aproximadamente 6 metros, sin embargo se muestran solo 4 en el gráfico (eje x de -2 a +2). La posición real final del robot se representa con una cruz azul. Se observa que la desviación estándar disminuye a medida que aumenta el número de

iteraciones de RANSAC, con un aumento prominente para $N = 12, 25, 50, 100$ y 200. Para N superiores la desviación estándar se ve aproximadamente parecida. Para los primeros gráficos (Ilustración 25), la trayectoria se muestra aproximadamente recta, llegando muy cerca al punto objetivo. Para la segunda mitad de los gráficos (Ilustración 26). La trayectoria se muestra ondulada al principio, y presenta una anomalía aproximadamente a los 50 metros del tramo, donde el robot parece desviarse aproximadamente a la derecha. La trayectoria real del robot no es recta, sino que presenta una pequeña ondulación en el eje X.

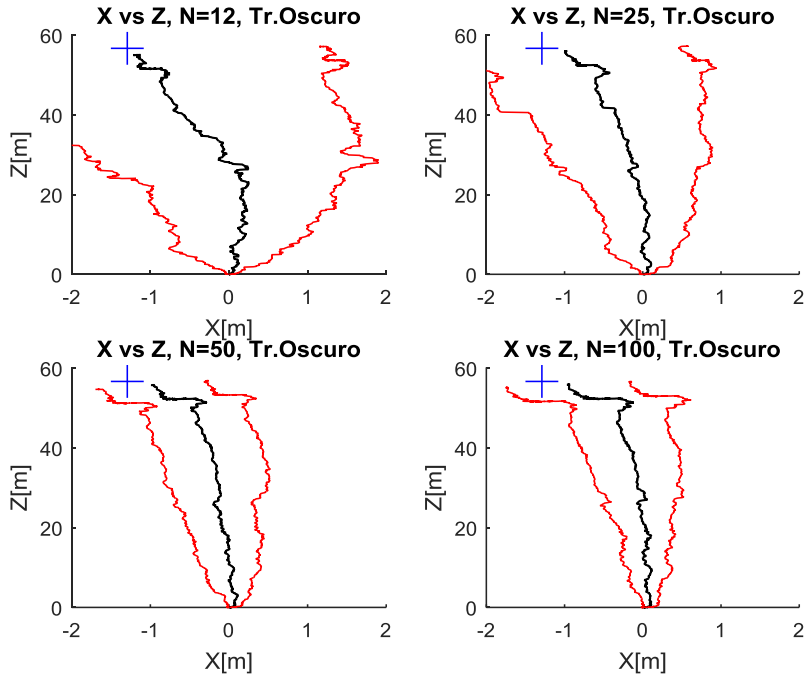


Ilustración 25: Comparación de trayectorias para N de 12 a 100, ejes X y Z. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la Ilustración 27 e Ilustración 28 se muestra la comparación de trayectorias para distinto número N de iteraciones de RANSAC, de 12 a 1600, para la sección oscura del túnel, para los ejes Y (arriba y abajo) y Z (profundidad). Esto corresponde a una vista lateral del robot. Al igual que en caso anterior, se observa una que la desviación estándar disminuye con el aumento de iteraciones, predominantemente para $N = 13, 25, 50, 100$ y 200. En este caso la media se encuentra desviada 0.5m de donde del punto objetivo, y no parece mejorar con N. La forma de la trayectoria es aproximadamente recta y horizontal, lo que parece correcto pues el robot está moviéndose sobre suelo terroso, aproximadamente plano. Pueden existir pequeñas piedras u hoyos que hagan que el robot varíe su altura en unos ± 5 centimetros.

En las siguientes Ilustración 29, Ilustración 30, Ilustración 31, Ilustración 32 e Ilustración 33 se muestra la variación de la orientación del vehículo. El eje X representa el número de la imagen procesada, y el eje Y el ángulo en grados.

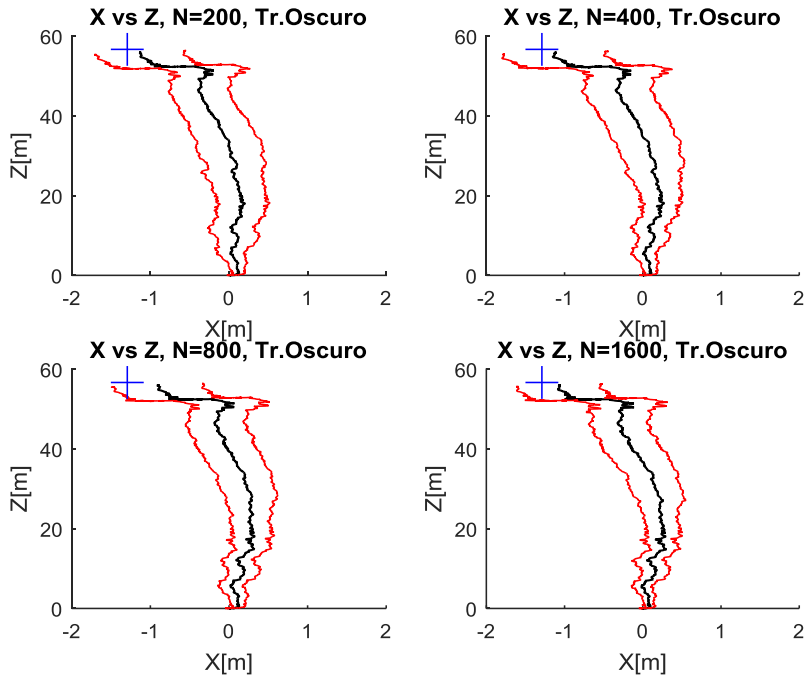


Ilustración 26: Comparación de trayectorias para N de 200 a 1600, ejes X y Z. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

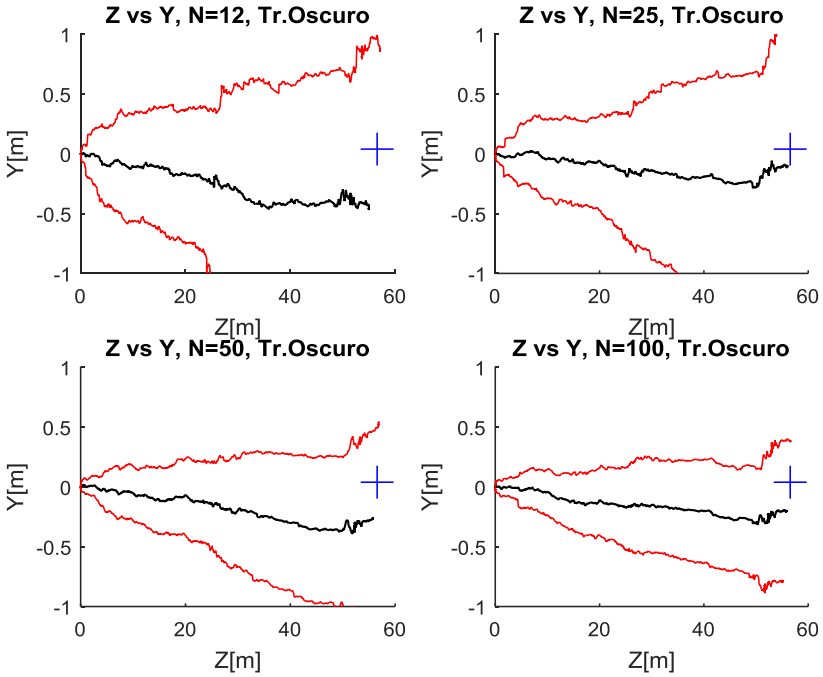


Ilustración 27: Comparación de trayectorias para N de 12 a 100, ejes Z e Y. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

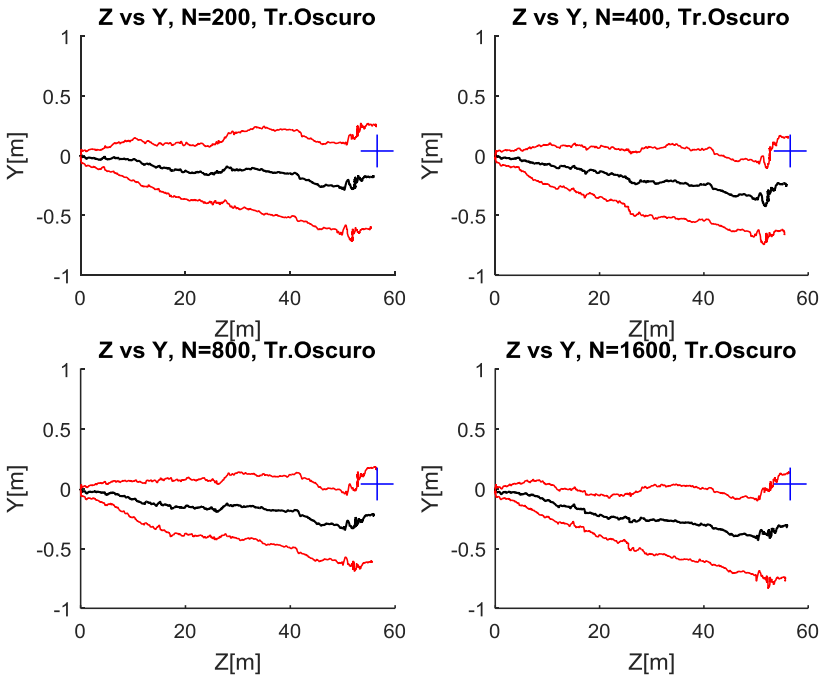


Ilustración 28: Comparación de trayectorias para N de 200 a 1600, ejes Z e Y. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la Ilustración 29 se muestra la orientación en el eje X, conocido como *Pitch*. Primeramente, se observa que para el caso $N = 12$ la desviación estándar se sale del gráfico. Como se utilizan tan pocas iteraciones, es muy probable que todos los intentos ocupen *outliers* para calcular la solución, y la probabilidad aumenta con el grado de contaminación de los puntos. La cercanía de la media al punto objetivo sin embargo no está tan desviada, pero de igual forma se descarta totalmente esta solución por su escaza estabilidad. Para los demás gráficos, la estabilidad mejora progresivamente con el número de iteraciones, y el punto final de la media se acerca cada vez más al punto final medido. La forma de la trayectoria contiene ruido que se explica dado el tipo de terreno presente al interior del túnel.

En la Ilustración 30 e Ilustración 31 se muestra la orientación en el eje Y, conocido como *Yaw*. Se observa, al igual que en los gráficos de *Pitch*, que para el caso $N = 12$ la desviación estándar es muy elevada. Para los demás gráficos, la estabilidad mejora progresivamente con el número de iteraciones. El punto final de la media se mantiene aproximadamente a 5º del punto final medido, para todos los N. La forma de la trayectoria contiene picos medianamente grandes, esto se explica pues el robot no viaja en línea recta, sino que tiene ondulaciones en el eje X y por lo tanto rota con respecto al eje Y.

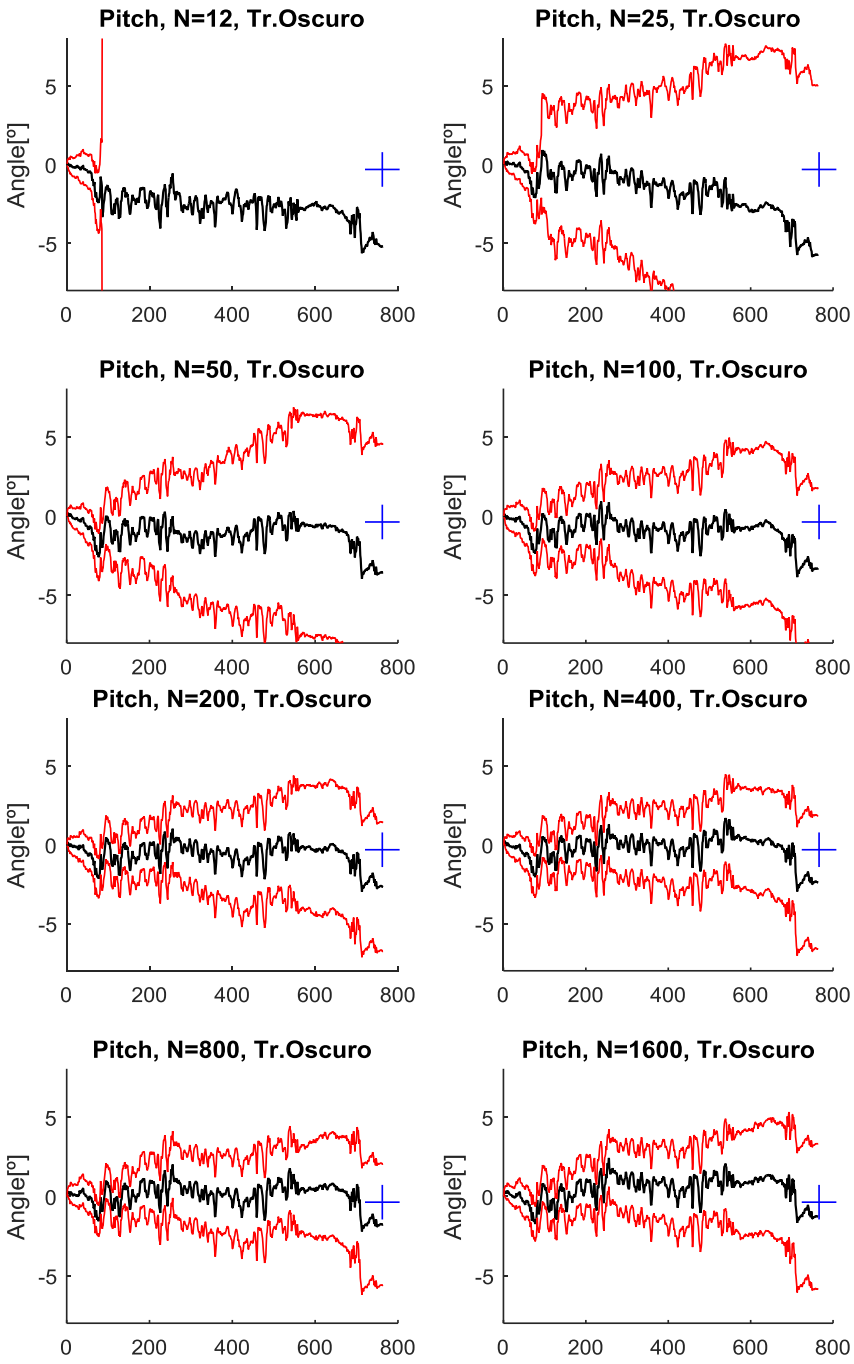


Ilustración 29: Comparación de orientación para N de 12 a 1600. Rotación en el eje *X*. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

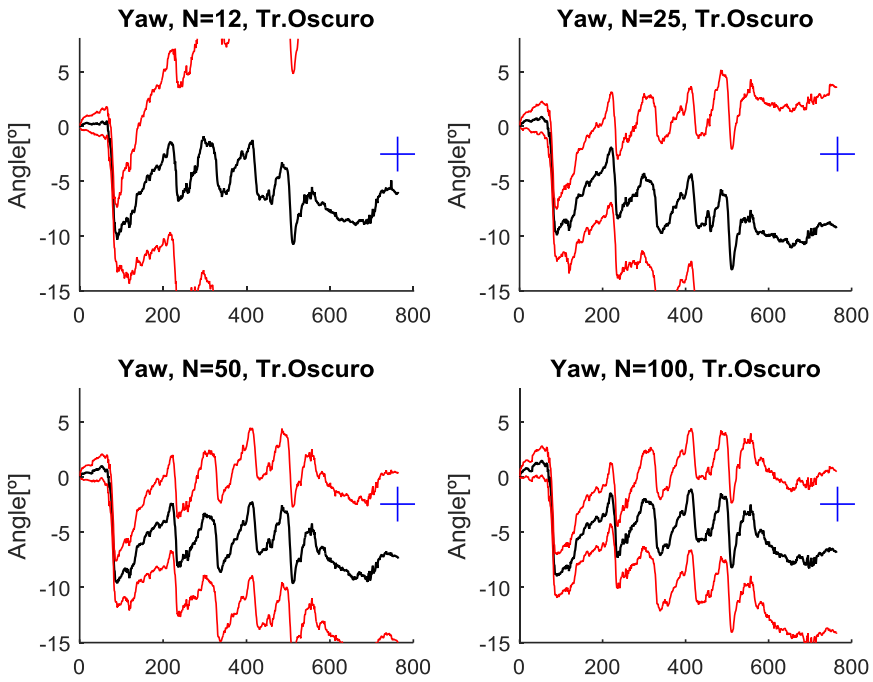


Ilustración 30: Comparación de orientación para N de 12 a 100. Rotación en el eje *Y*. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la Ilustración 32 e Ilustración 33 se muestra la orientación en el eje Z, conocido como *Roll*. Se observa, al igual que en los gráficos de *Pitch* y *Yaw*, que para el caso $N = 12$ la desviación estándar es muy elevada. Para los demás gráficos, la estabilidad mejora progresivamente con el número de iteraciones. En estos casos el punto final de la media se acerca bastante al punto final medido, sobre todo para $N > 50$. La forma de la trayectoria contiene picos medianamente pequeños, esto se explica pues el terreno no es perfectamente plano, y el robot presenta 4 ruedas por lo que a medida que avanza se tambalea en el eje Z.

Un resumen de todos los grafios expuestos anteriormente se encuentra en el Anexo B. Resumen de Gráficos de Resultados.

En las Ilustración 34 e Ilustración 35 se comparan las desviaciones estándar de un cada eje y orientación para todos los valores de N. En éstos se aprecia de mejor manera la tendencia a la baja con el aumento de N. Se observa también que la estabilidad se satura mucho para $N > 200$. Esto puede provenir del hecho de que el parámetro d se mantiene constante en 0.07, y por lo tanto la solución no puede mejorar. La forma de la curva, para todos los casos, presenta forma de escalera, lo que puede provenir del aumento en la dificultad del reconocimiento y emparejamiento de características a medida que se atraviesa el túnel, pues se encuentra en un entorno con luminosidad, con luz variable. Este análisis aplica tanto para la posición como para la orientación.

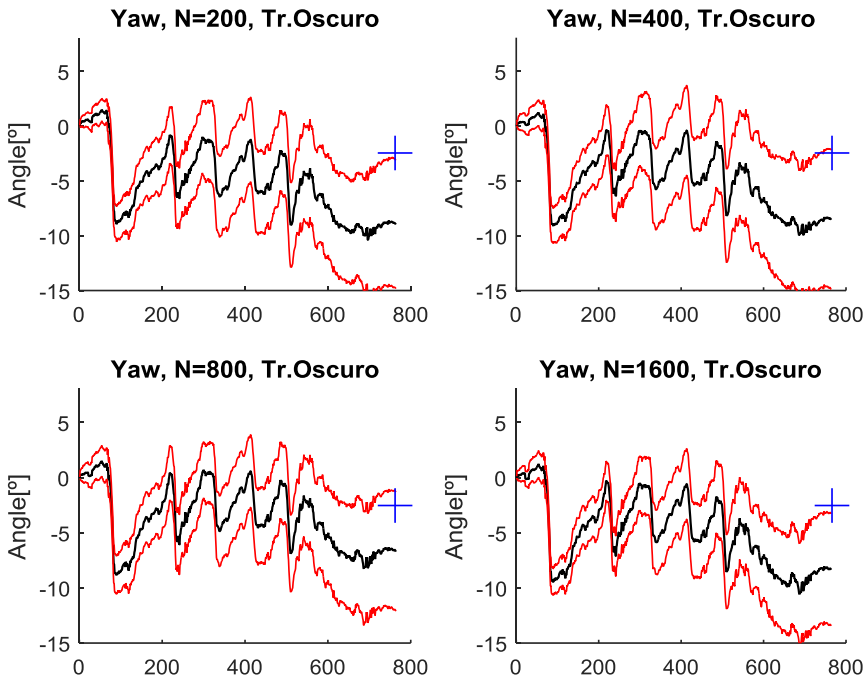


Ilustración 31: Comparación de orientación para N de 200 a 1600. Rotación en el eje *Y*. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

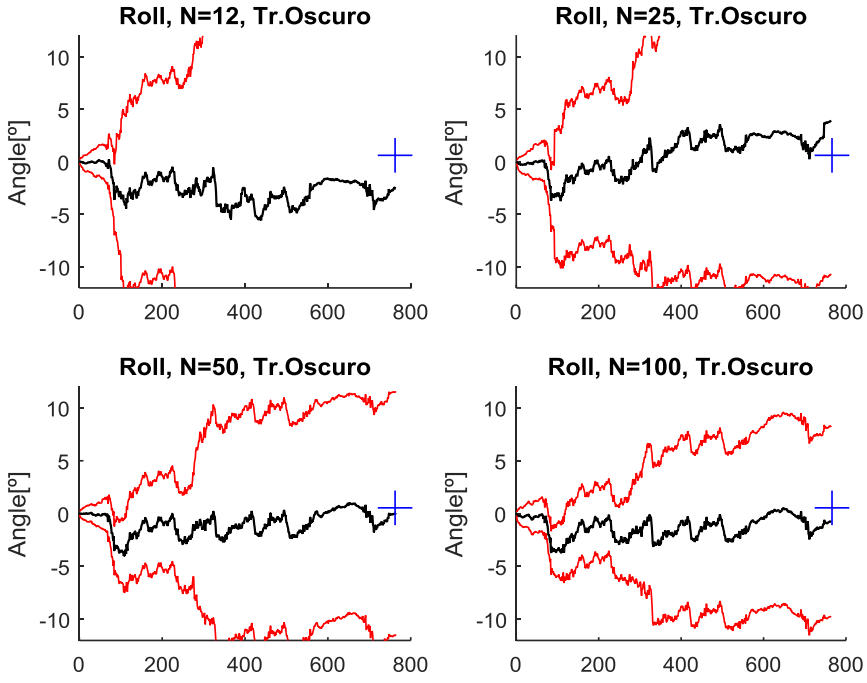


Ilustración 32: Comparación de orientación para N de 12 a 100. Rotación en el eje *Z*. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

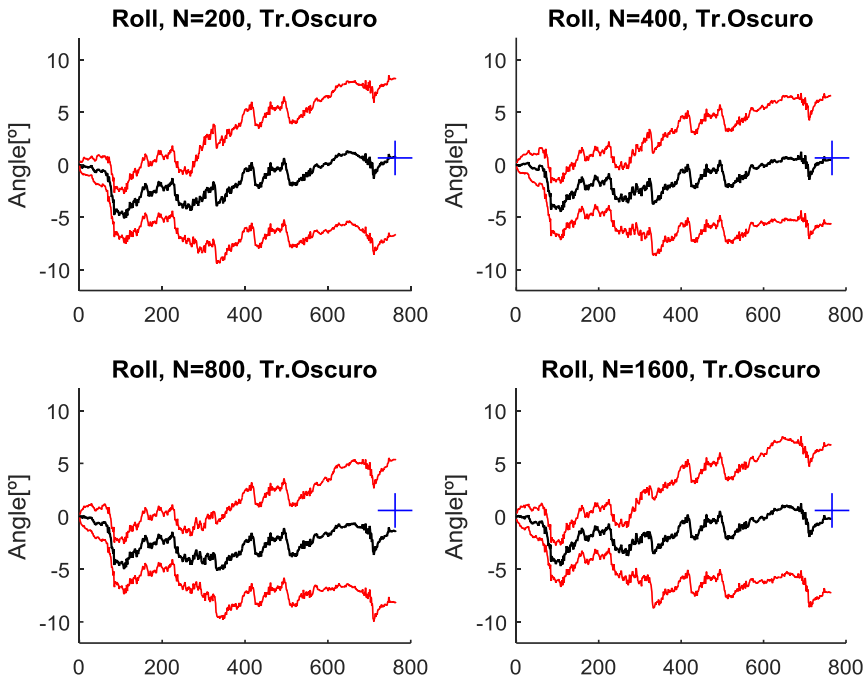


Ilustración 33: Comparación de orientación para N de 200 a 1600. Rotación en el eje *Z*. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

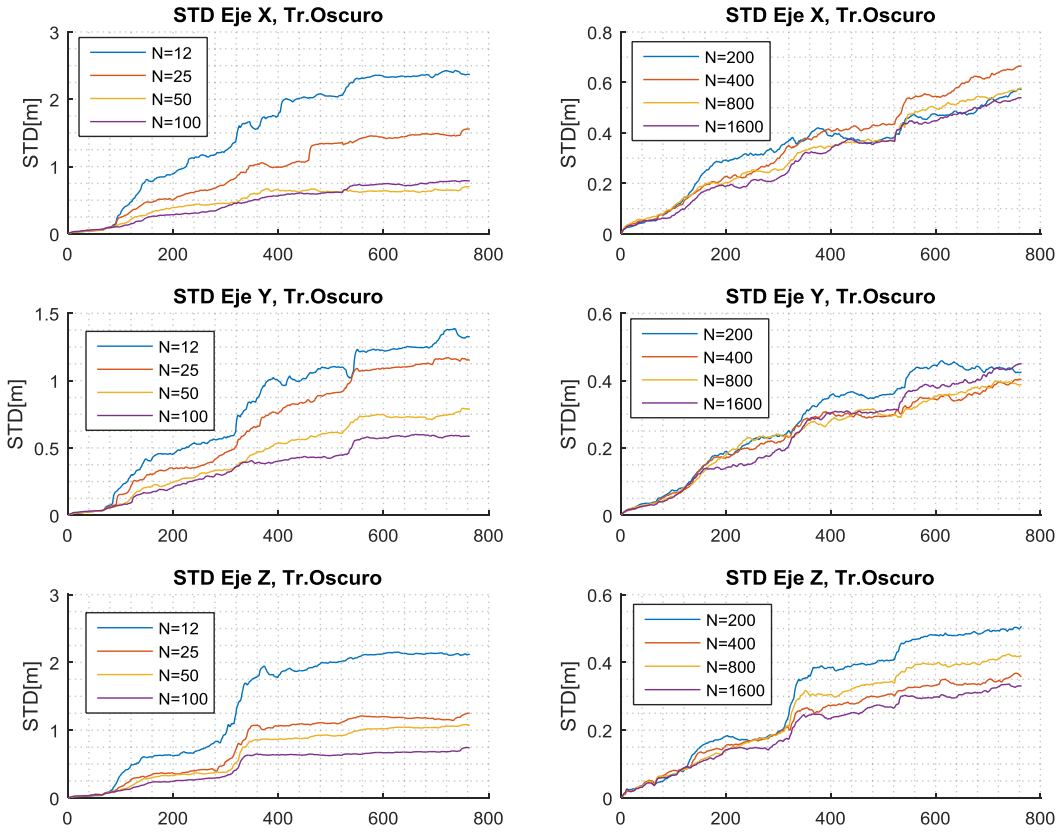


Ilustración 34: Comparación de desviación estándar para los distintos valores de N, para los ejes X, Y y Z. Tramo oscuro.

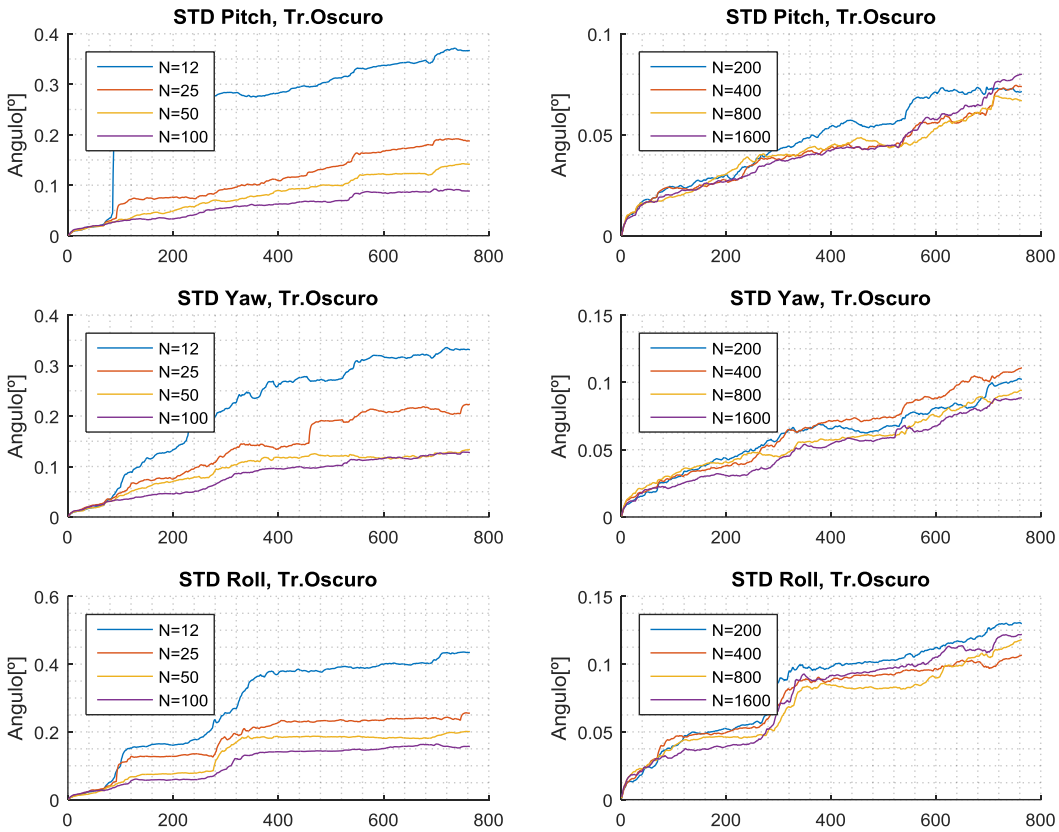


Ilustración 35: Comparación de desviación estándar para los distintos valores de N, para los la orientacion del robot, Pitch, Yaw y Roll. Tramo oscuro.

En la Ilustración 36 se observa el gráfico de medias para los 6 grados de libertad del robot. El eje X representa el número de la imagen procesada y el eje Y el valor de la media. Los valores han sido escalados para entrar en el segmento $[-1.5\,,1\,]$ para un mejor análisis. Primeramente destaca la forma de la curva para el eje Z, la cual no contiene ruido que se destaque a simple vista y está alineada perfectamente con la posición final. Esto se explica debido a que en la detección de profundidad el emparejamiento se hace entre las cámaras izquierda y derecha, y se buscan soluciones que se encuentren dentro de una misma línea epipolar, eliminando la mayoría de los errores de emparejamiento que puedan suceder. Se observa también un tramo en el cual el robot deja de avanzar en este eje, y del video se extrae que el robot esta de hecho detenido en este intervalo de tiempo. Esto significa que en todas las curvas este tramo debería ser recto. Si bien para algunas variables se mantiene aproximadamente recto, en otras se desplaza considerablemente, como ocurre en el eje X, especialmente en $N > 200$. Este error debe provenir de la contaminación de *outliers*.

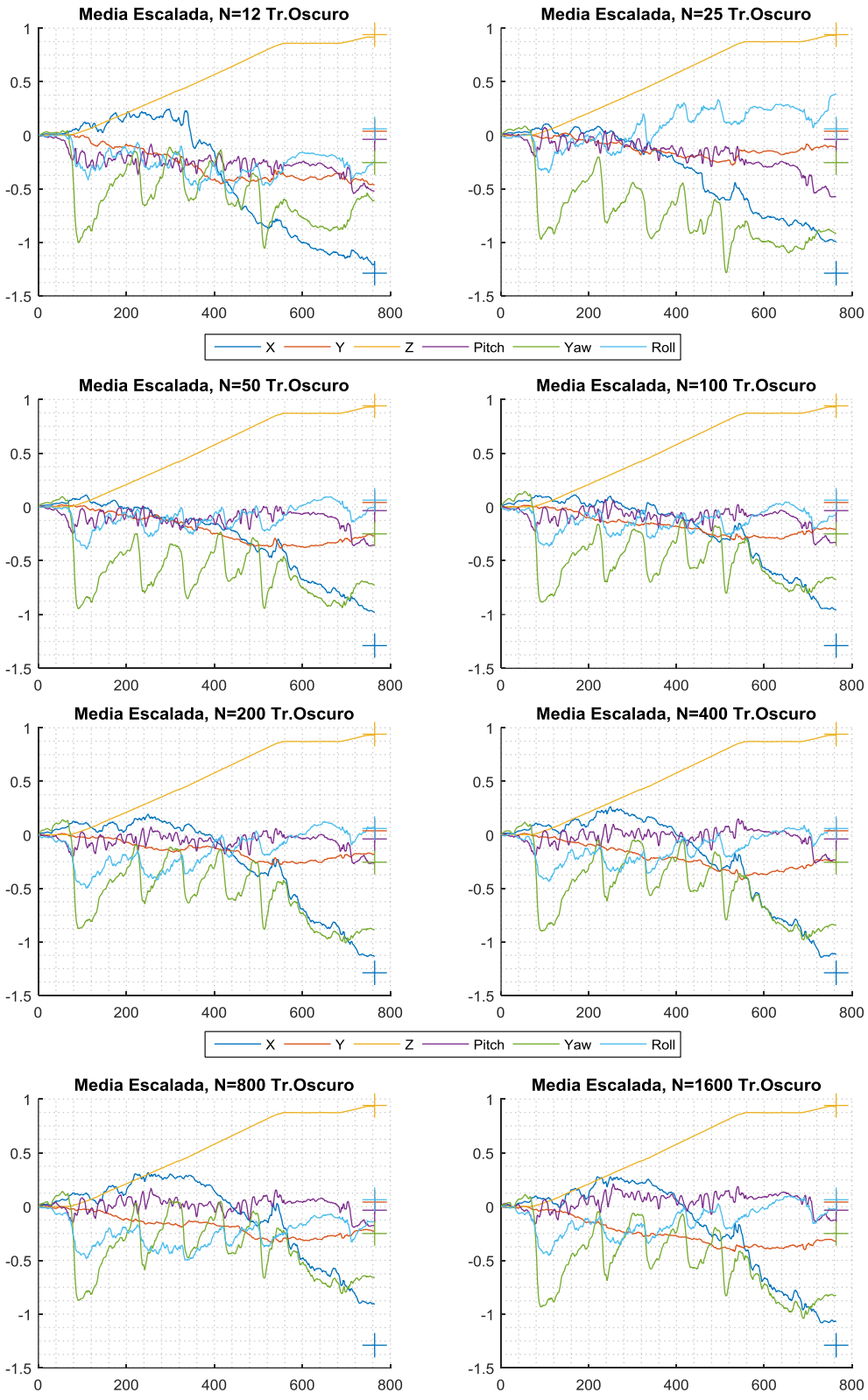


Ilustración 36: Comparación de la forma de la solución para todos los valores de N. Las unidades han sido escaladas. Las cruces representan la posicion real medida. Tramo oscuro.

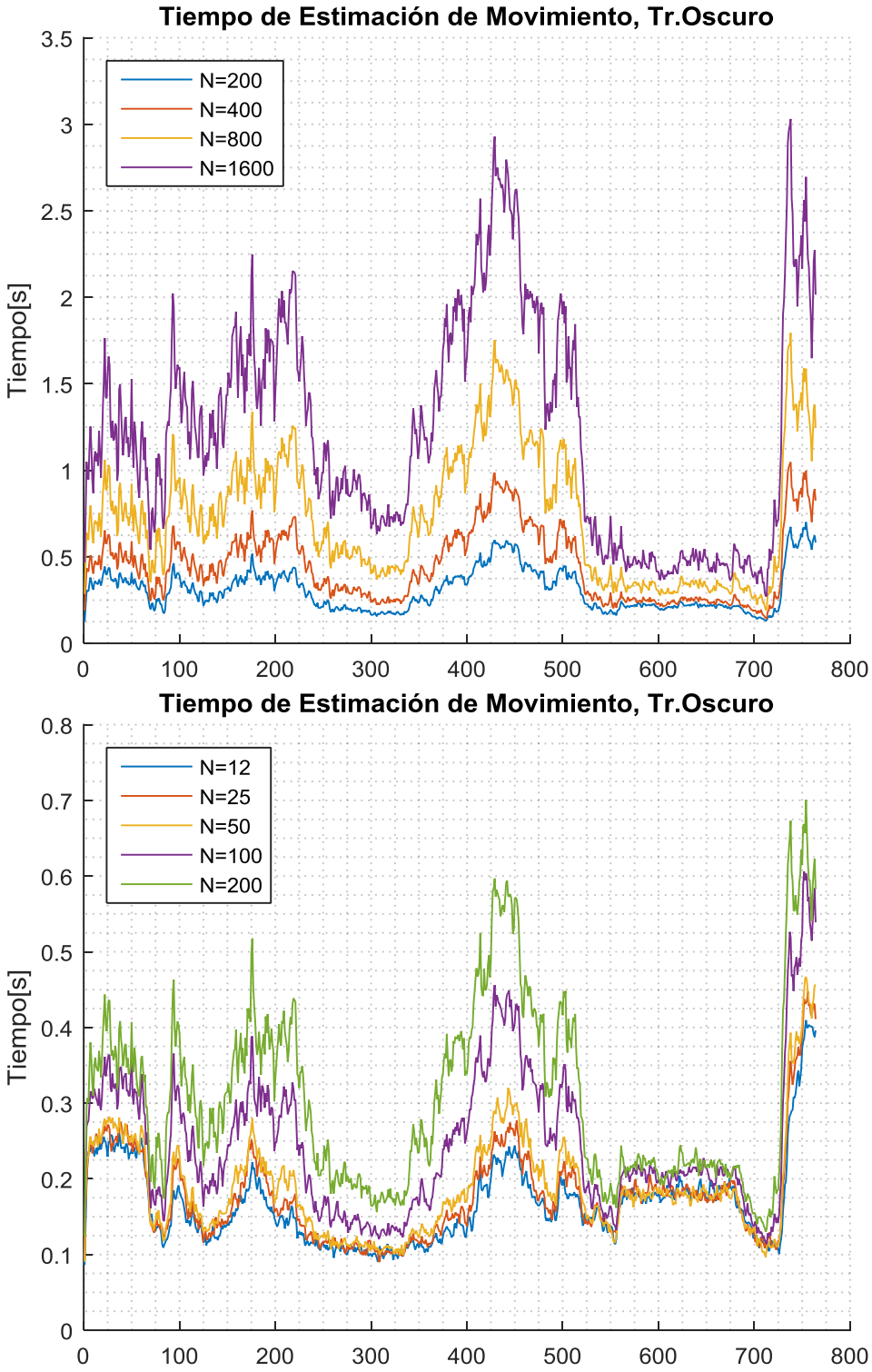


Ilustración 37: Comparación del tiempo de procesamiento de la etapa de estimación de movimiento. Tramo Oscuro.

Los tiempos totales de la etapa de estimación de movimiento quedan representados en la Ilustración 37. Se observa un incremento proporcional al número de iteraciones N. La forma de la curva es similar entre todos los casos. Se observa también que la solución presenta zonas de mayor tiempo de cómputo. Esto dado a que la cantidad de características

varía en cada imagen a medida que avanza el robot, y en las zonas del túnel iluminadas aumenta considerablemente con respecto a las zonas oscuras, lo que incrementa el tiempo de cómputo en la misma proporción. También se observa que la curva converge cerca de $N = 12, 25\ y\ 50$ y el tiempo mínimo puede explicarse debido a la existencia de una breve etapa de *matching* contenida en este análisis, que no varía con el número de iteraciones N, pero si con el número de características encontradas.

En la Ilustración 38 se muestra la comparación de trayectorias para distinto número N de iteraciones de RANSAC, de 12 a 1600, para la sección iluminada del túnel. En estos gráficos se encuentra la información de 50 realizaciones del algoritmo, representados por la media (línea negra) y la desviación estándar (línea roja). El eje X corresponde a la coordenada X del robot en el túnel (izquierda y derecha), y el eje Y representa la coordenada Z (profundidad). Esto corresponde a una vista superior del robot. El ancho medio del túnel es de aproximadamente 6 metros, sin embargo se muestran solo 4 en el gráfico (eje x de -2 a +2). La posición real final del robot se representa con una cruz azul. Al igual que en el caso de la sección oscura del túnel, se observa que la desviación estándar disminuye a medida que aumenta el número de iteraciones de RANSAC, con un aumento prominente para $N = 12, 25, 50, 100$ y 200. Para N superiores la desviación estándar se ve aproximadamente parecida. La trayectoria es aproximadamente recta y presenta pequeñas ondulaciones en la coordenada X. La trayectoria real del robot no es recta, sino que presenta una pequeña ondulación en el eje X. Se observa que existe variabilidad en el destino final del robot para N menor o igual a 50 en el eje X. En el eje Z se observa una solución bastante estable, en el sentido de que en todos los casos se llega al objetivo. La estabilidad de las soluciones es bastante más favorables en este caso, sobretodo en el eje X, comparada con los resultados del tramo oscuro del túnel.

En la Ilustración 39 se muestra la comparación de trayectorias para distinto número N de iteraciones de RANSAC, de 12 a 1600, para la sección iluminada del túnel, para los ejes Y (arriba y abajo) y Z (profundidad). Esto corresponde a una vista lateral del robot. Al igual que en caso anterior, se observa una que la desviación estándar disminuye con el aumento de iteraciones, predominantemente para $N = 13, 25, 50, 100$ y 200. En este caso la media se encuentra desviada 0.5m de donde del punto objetivo, y no parece mejorar con N. La forma de la trayectoria es aproximadamente recta y horizontal, lo que parece correcto pues el robot está moviéndose sobre suelo terroso, aproximadamente plano. Pueden existir pequeñas piedras u hoyos que hagan que el robot varíe su altura en unos ± 5 centimetros.

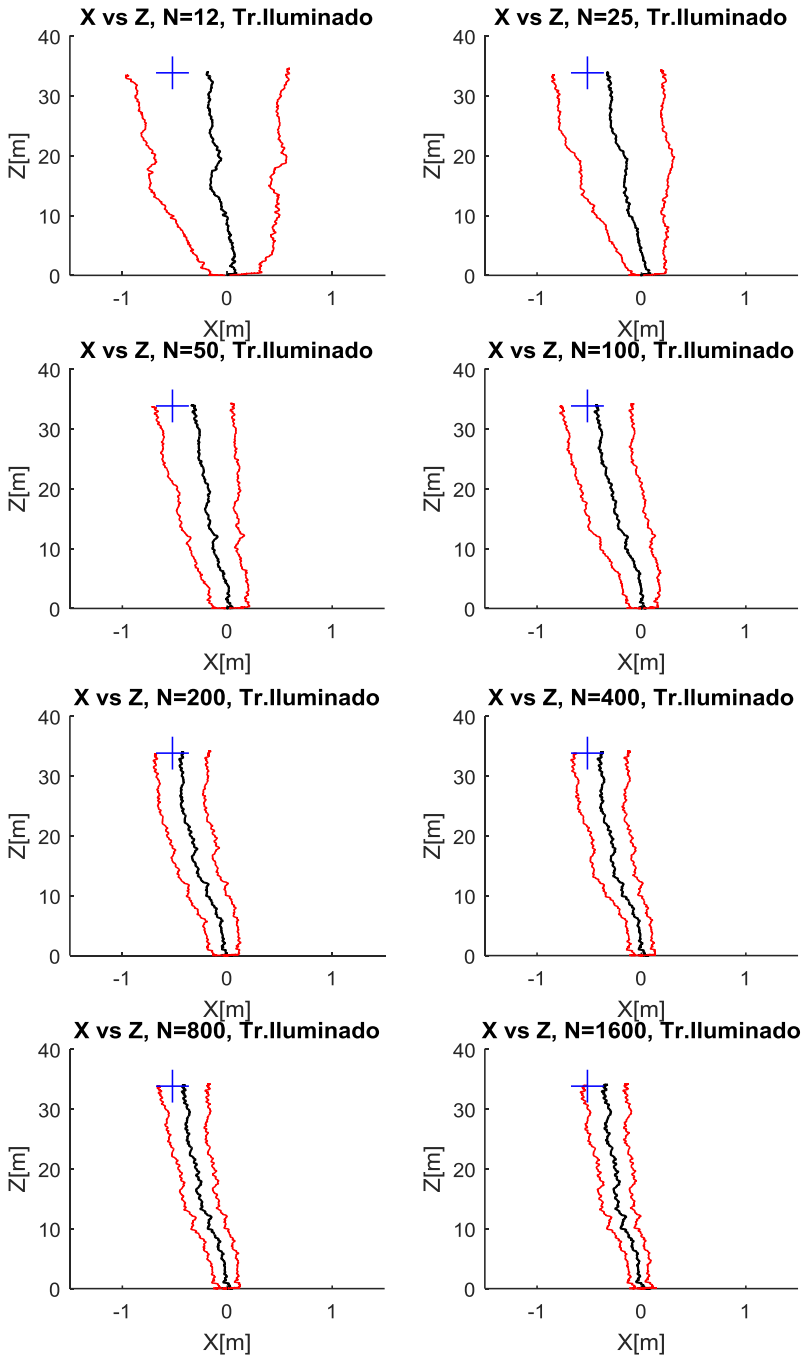


Ilustración 38: Comparación de trayectorias para N de 12 a 1600, ejes X y Z. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

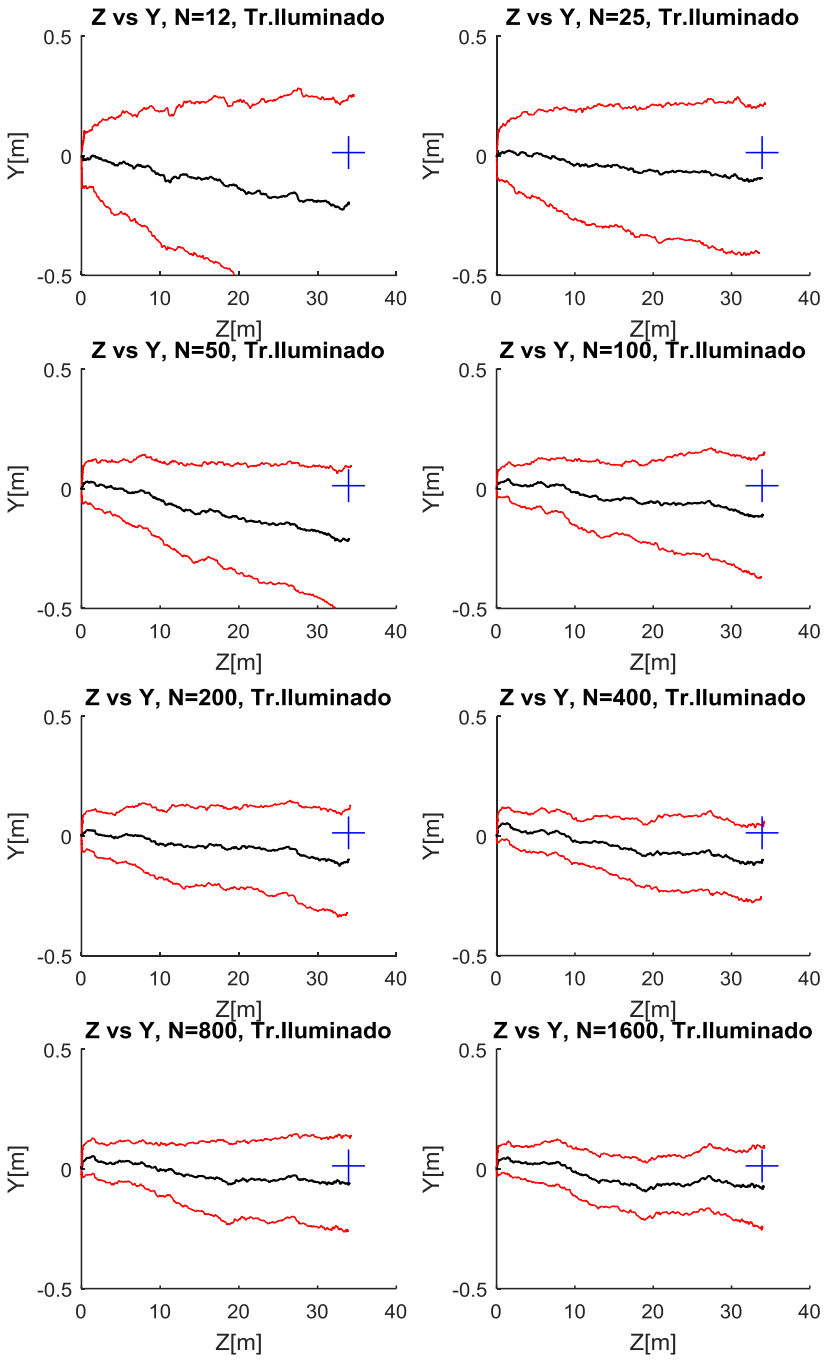


Ilustración 39: Comparación de trayectorias para N de 12 a 1600, ejes Z e Y. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

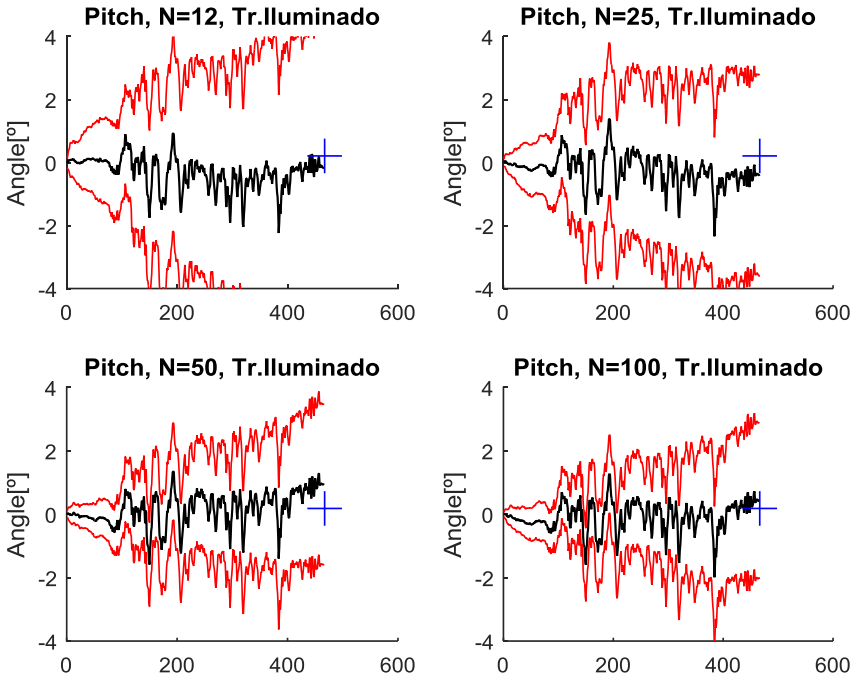


Ilustración 40: Comparación de orientación para N de 12 a 100. Rotación en el eje *X*. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En las siguientes Ilustración 40, Ilustración 41, Ilustración 42 e Ilustración 43 se muestra la variación de la orientación del vehículo. El eje X representa el número de la imagen procesada, y el eje Y el ángulo en grados.

En la Ilustración 40 y la Ilustración 41 se muestra la orientación en el eje X, conocido como *Pitch*. La estabilidad mejora progresivamente con el número de iteraciones, y el punto final de la media se acerca cada vez más al punto final medido, y llega justo a éste para $N \geq 200$. La forma de la trayectoria contiene ruido que se explica dado el tipo de terreno presente al interior del túnel. La solución encontrada presenta mayor estabilidad que en el intervalo oscuro del túnel, probablemente por la calidad de imagen.

En la Ilustración 42 se muestra la orientación en el eje Y, conocido como *Yaw*. Se observa, al igual que en los gráficos de *Pitch*, que para el caso $N = 12$ la desviación estándar es muy elevada. Para los demás casos, la estabilidad mejora progresivamente con el número de iteraciones. El punto final de la media se mantiene aproximadamente a 1º del punto final medido, para todos los N. La forma de la trayectoria contiene picos medianamente grandes, esto se explica pues el robot no viaja en línea recta, sino que tiene ondulaciones en el eje X y por lo tanto rota con respecto al eje Y. Comparado con el tramo oscuro del túnel, ésta solución presenta mayor estabilidad.

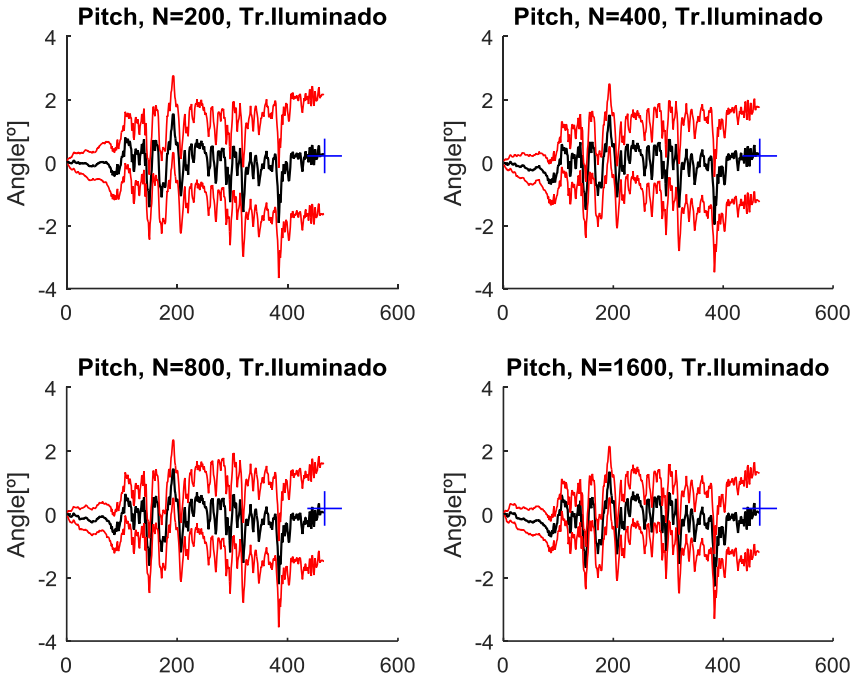


Ilustración 41: Comparación de orientación para N de 200 a 1600. Rotación en el eje *X*. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la Ilustración 43 se muestra la orientación en el eje Z, conocido como *Roll*. Se observa, al igual que en los gráficos de *Pitch* y *Yaw*, que para el caso $N = 12$ la desviación estándar es muy elevada. Para los demás casos, la estabilidad mejora progresivamente con el número de iteraciones. El punto final de la media se acerca bastante al punto final medido, sobre todo para $N > 25$. La forma de la trayectoria contiene picos medianamente pequeños, esto se explica pues el terreno no es perfectamente plano, y el robot presenta 4 ruedas por lo que a medida que avanza se tambalea en el eje Z. Comparado con el tramo oscuro del túnel, las soluciones tienen mayor estabilidad.

Un resumen de todos los grafios expuestos anteriormente se encuentra en el Anexo B. Resumen de Gráficos de Resultados.

En la Ilustración 44 e Ilustración 45 se comparan las desviaciones estándar de un cada eje y orientación para todos los valores de N. En éstos se aprecia de mejor manera la tendencia a la baja con el aumento de N. Se observa también que la estabilidad se satura para $N > 200$ en el caso de la posición, y $N > 50$ para la orientación. Esto puede provenir del hecho de que el parámetro d se mantiene constante en 0.07, y por lo tanto la solución no puede mejorar. La forma de la curva en este caso es más suave y no es escalonada, esto pues la luminosidad en esta sección del túnel es mucho más estable.

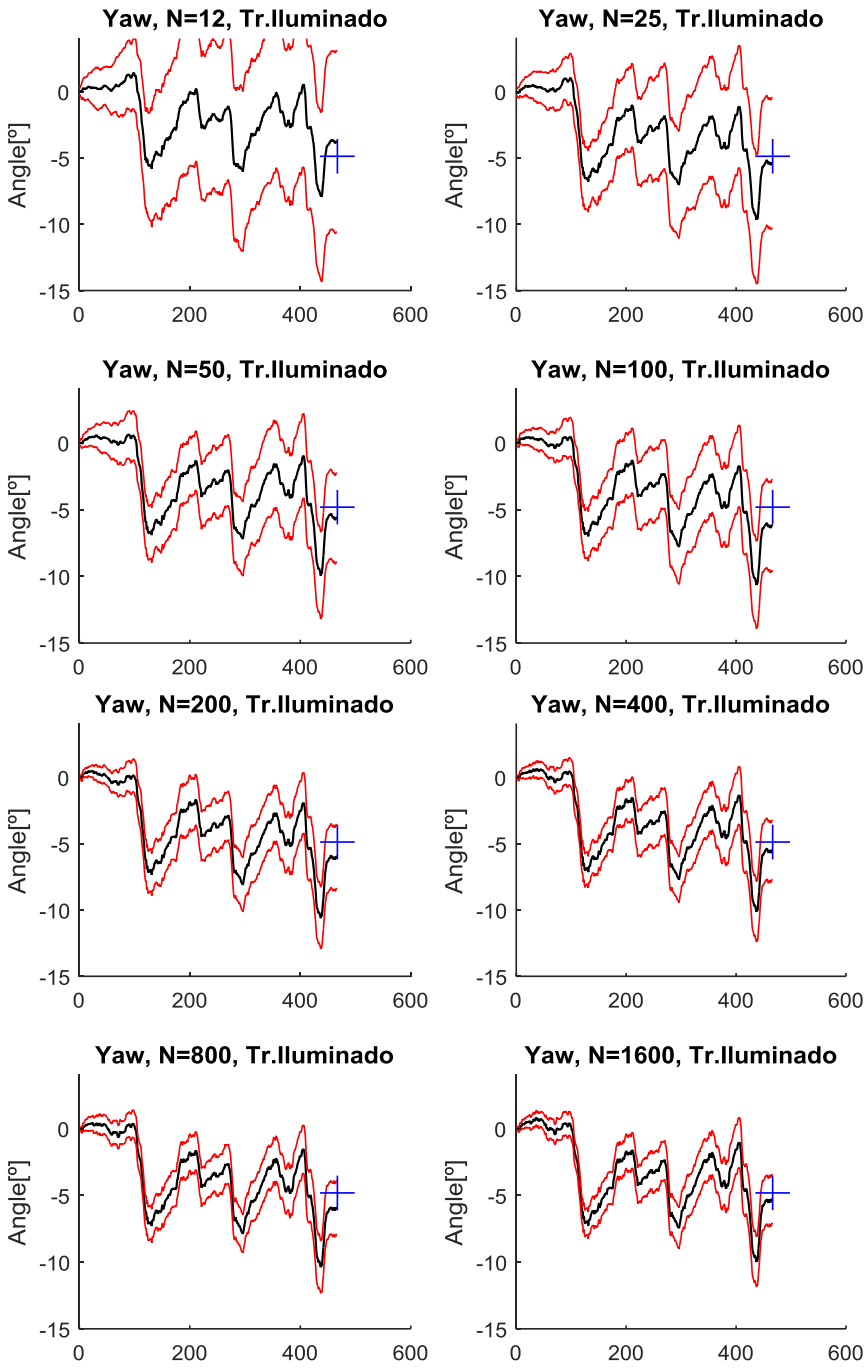


Ilustración 42: Comparación de orientación para N de 12 a 1600. Rotación en el eje *Y*. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

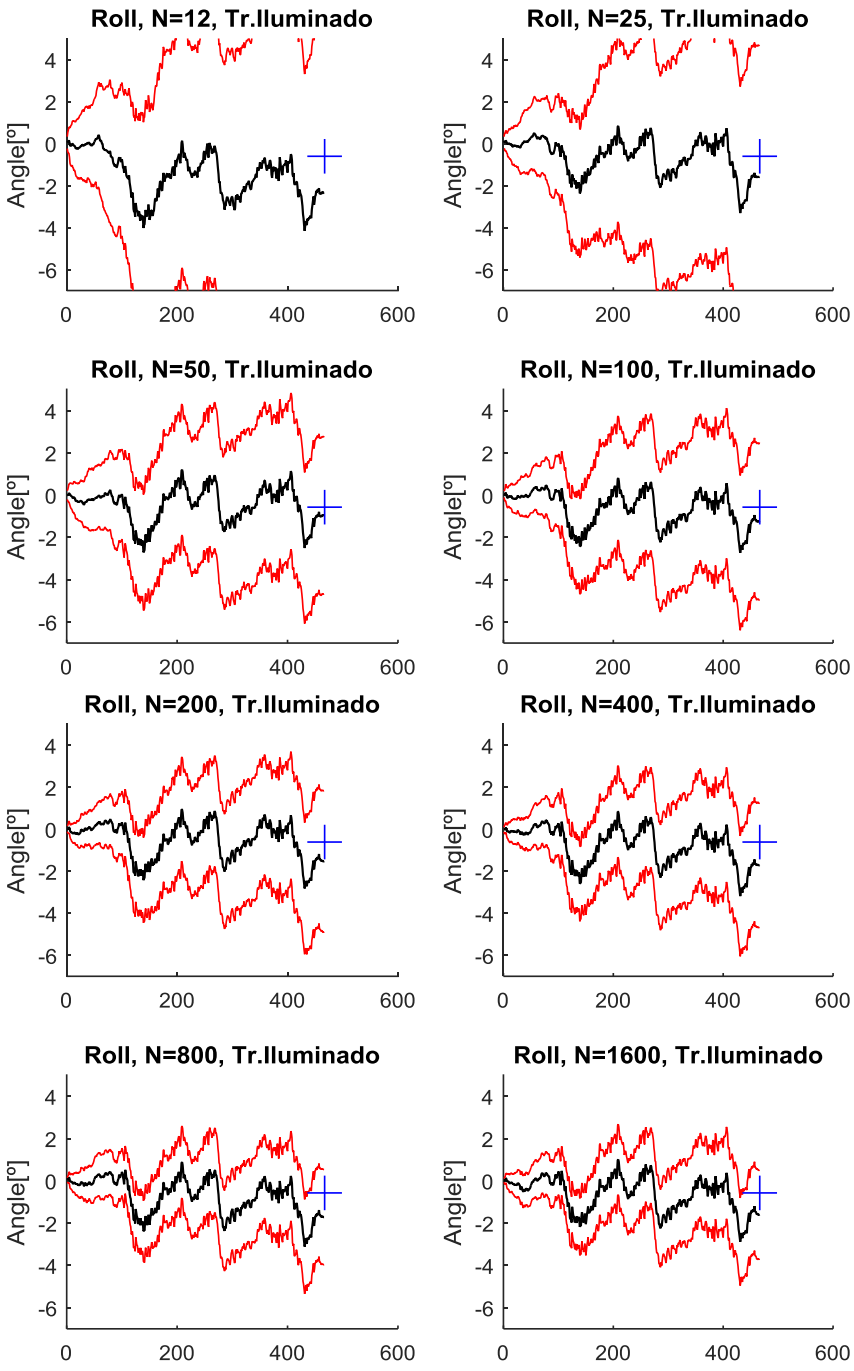


Ilustración 43: Comparación de orientación para N de 12 a 1600. Rotación en el eje *Z*. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

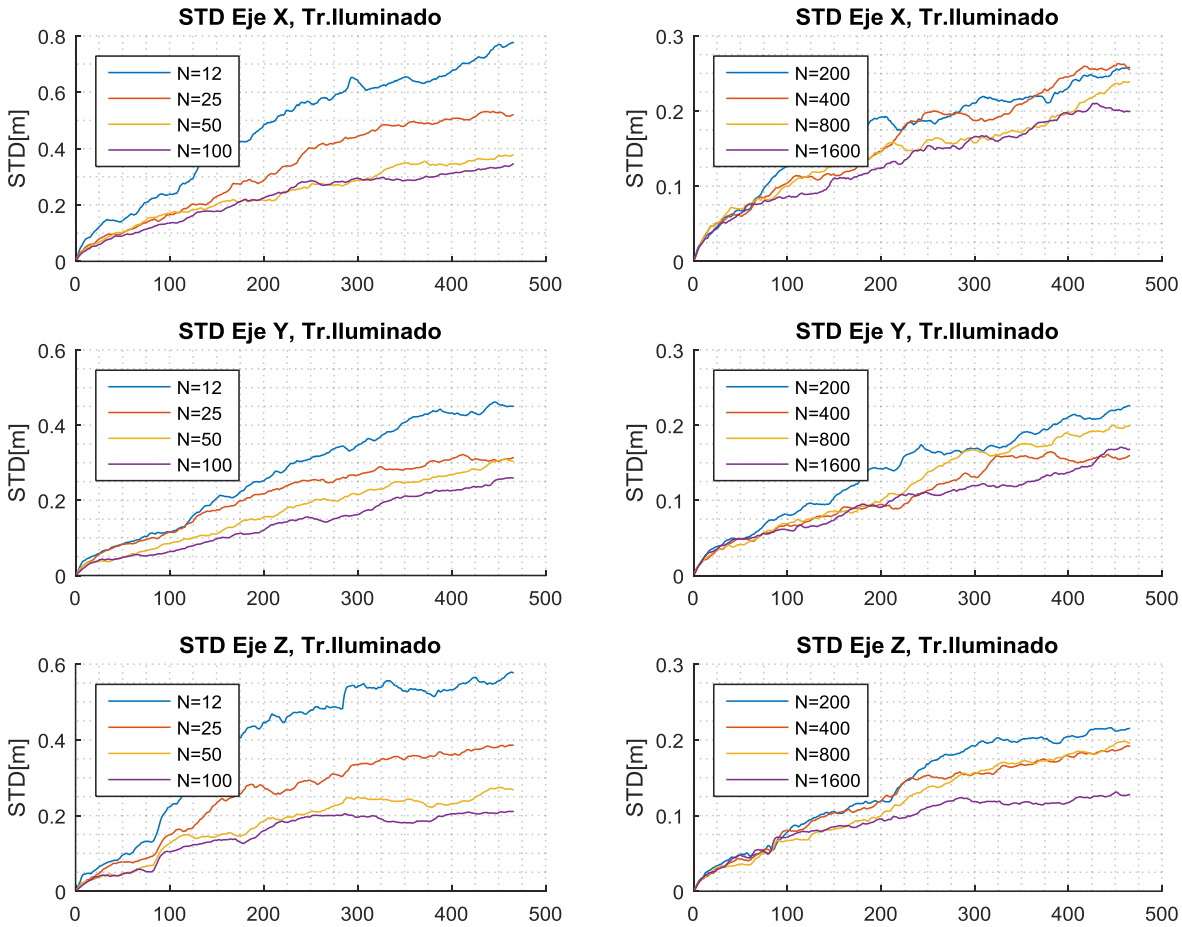


Ilustración 44: Comparación de desviación estándar para los distintos valores de N, para los ejes X, Y y Z. Tramo iluminado.

En la Ilustración 46 se observa el gráfico de medias para los 6 grados de libertad del robot. El eje X representa el número de la imagen procesada y el eje Y el valor de la media. Los valores han sido escalados para entrar en el segmento $[-1.5\ ,1\ ]$ para un mejor análisis. Primeramente destaca la forma de la curva para el eje Z, la cual no contiene ruido que se destaque a simple vista y está alineada perfectamente con la posición final. Esto se explica debido a que en la detección de profundidad el emparejamiento se hace entre las cámaras izquierda y derecha, y se buscan soluciones que se encuentren dentro de una misma línea epipolar, eliminando la mayoría de los errores de emparejamiento que puedan suceder. Se observa una sección plana al principio, pues el robot se mantiene quieto por un pequeño periodo de tiempo, y luego comienza a acelerar lentamente. Esto significa que en todas las curvas este tramo debería ser recto. En el caso de esta sección del túnel, todas las curvas se mantienen rectas, en este pequeño tramo, lo que da fe de una solución más confiable. Se observa también que las medias en general calzan con el punto objetivo, excepto para la altura en el eje Y, en donde se encuentra un poco desplazada.

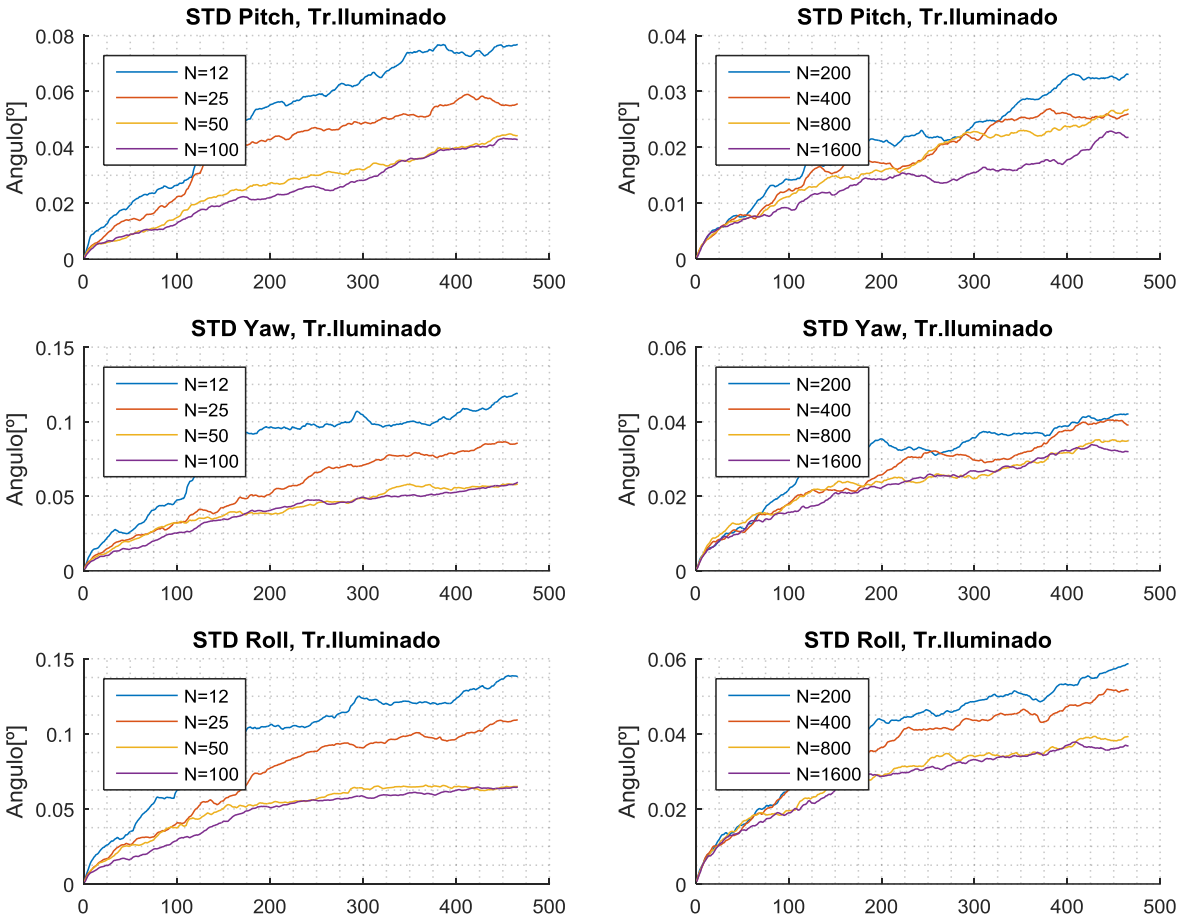


Ilustración 45: Comparación de desviación estándar para los distintos valores de N, para los la orientacion del robot, Pitch, Yaw y Roll. Tramo iluminado.

Los tiempos totales de la etapa de estimación de movimiento quedan representados en la Ilustración 47. Se observa un incremento proporcional al número de iteraciones N. La forma de la curva es similar entre todos los casos. Se observa también que la solución es más estable comparada con el tramo oscuro del túnel. Esto dado a que la cantidad de características es aproximadamente constante en cada imagen a medida que avanza el robot, pues esta sección del túnel presenta iluminación con mayor estabilidad. También se observa que la curva converge cerca de $N = 12, 25\ y\ 50$ y el tiempo mínimo puede explicarse debido a la existencia de una breve etapa de *matching* contenida en este análisis, que no varía con el número de iteraciones N, pero si con el número de características encontradas.

No se notan mayores diferencias en el tiempo medio de ejecución para ambos tramos del túnel, sin embargo se observa que la forma de la curva es bastante más estable en el tramo iluminado que en el oscuro, puesto que en este ultimo las condiciones de iluminación eran variables. Se concluye que la estabilidad de la iluminación influye en el tiempo de ejecución del algoritmo. Esto posiblemente por el menor número de características detectadas en los tramos oscuros, como se muestra en la Ilustración 23.

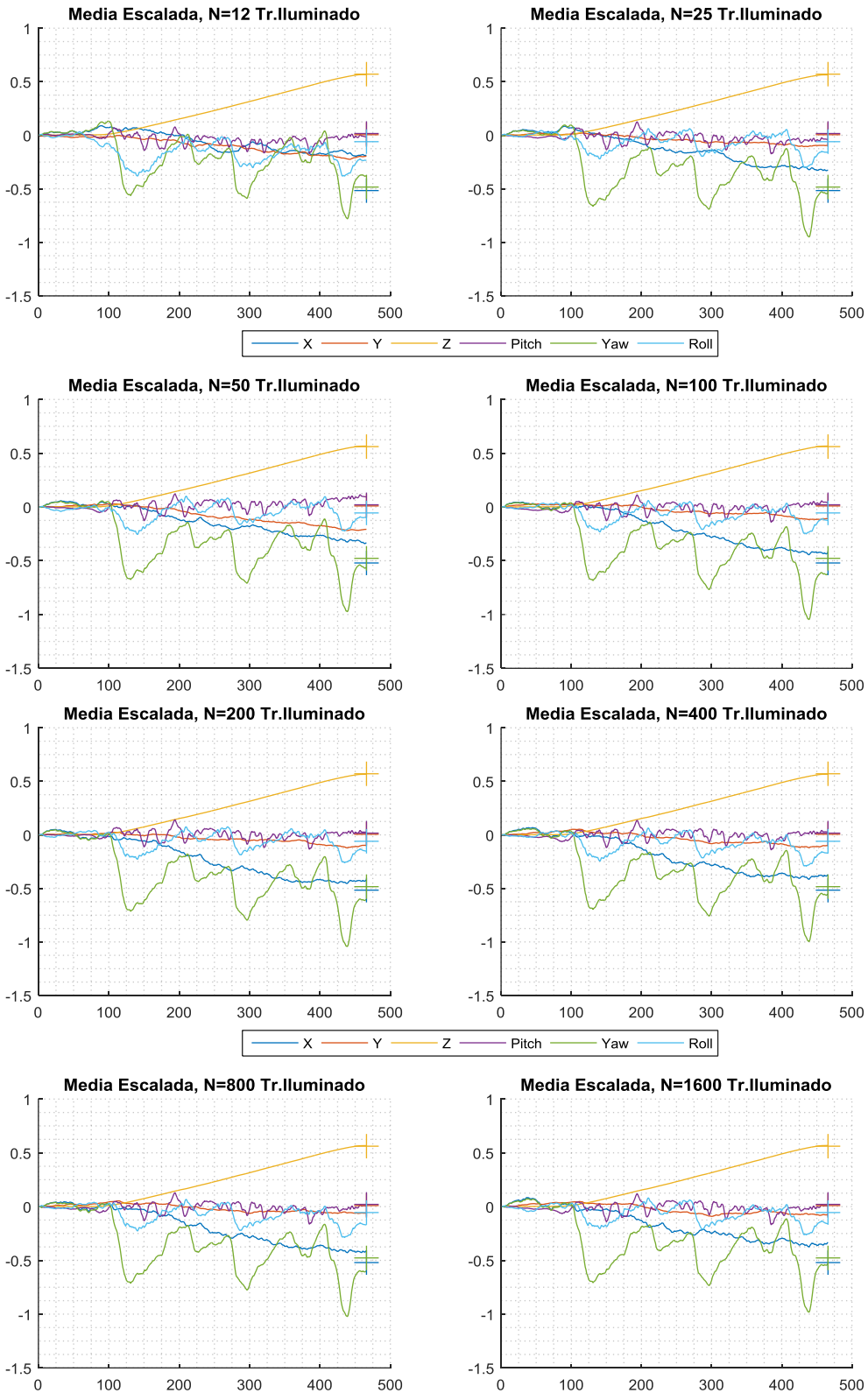


Ilustración 46: Comparación de la forma de la solución para todos los valores de N. Las unidades han sido escaladas. Las cruces representan la posicion real medida. Tramo iluminado.

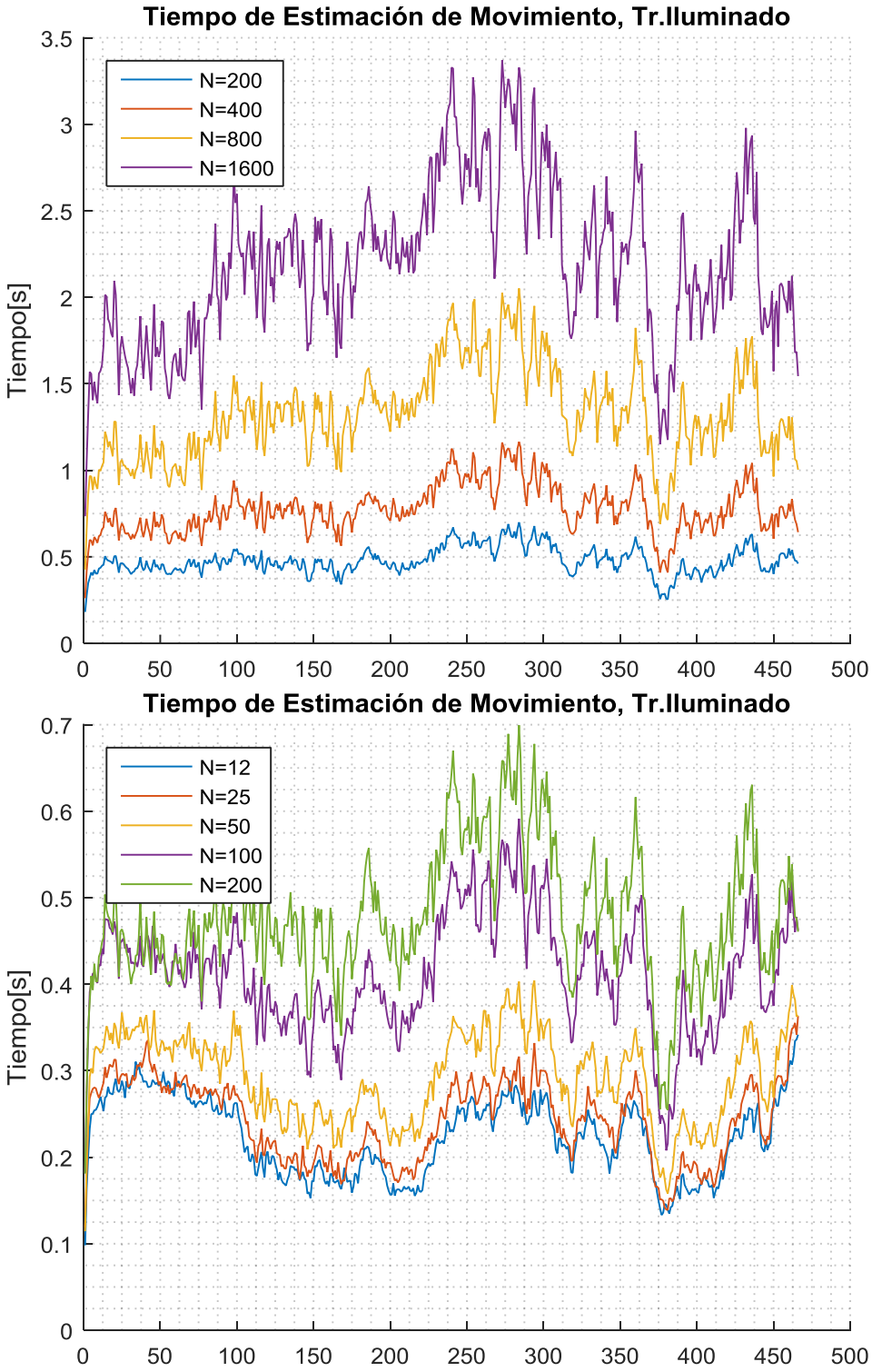


Ilustración 47: Comparación del tiempo de procesamiento de la etapa de estimación de movimiento. Tramo iluminado.

En general dada la baja luminosidad del tramo oscuro del túnel, las soluciones encontradas para aquel tramo en general presentan peor calidad y estabilidad comparadas con las soluciones del tramo iluminado del túnel. Esto se explica pues el algoritmo de detección de características tiene problemas para funcionar en ambientes de poco contraste, en donde el

número de características encontradas es baja. Esto se ve reflejado de cierta manera en los gráficos de tiempo de ambos tramos en donde en las partes oscuras del túnel (Ilustración 37) el tiempo de procesamiento se reduce drásticamente al contener menos puntos, comparado con la sección iluminada del túnel (Ilustración 47) en donde la luminosidad es mucho más estable.

4.3.2 Pre procesamiento de las Imágenes

En esta sección se muestran los resultados de aplicar pre procesamiento a las imágenes del tramo oscuro del túnel, para mejorar la efectividad del algoritmo. El algoritmo se corre en 50 instancias diferentes. Estos resultados son distintos unos de otros dada la naturaleza no determinista del algoritmo. Los parámetros de RANSAC utilizados en este experimento se muestran en la Tabla 9.

Tabla 9: Parámetros de RANSAC utilizados en el experimento.

Parámetro	Valor
Núm. de puntos	RANSAC-5
d	0.07
N	200,400

Dada las anomalías detectadas en la sección oscura del túnel para el caso anterior debido a errores de *matching* de características, se decidió mejorar el contraste de las imágenes provenientes de la cámara. La técnica utilizada para mejorar el contraste está basada en la ecualización de histograma de la imagen, como se explica en la Sección 2.7.

En la Ilustración 48 se observa la comparación de trayectorias para número N de iteraciones de RANSAC de 200 y 400, para la sección oscura del túnel para el caso de las imágenes ecualizadas y las sin pre procesar. En estos gráficos se encuentra la información de 50 realizaciones del algoritmo, representados por la media (línea negra) y la desviación estándar (línea roja). El eje X corresponde a la coordenada X del robot en el túnel (izquierda y derecha), y el eje Y representa la coordenada Z (profundidad). Esto corresponde a una vista superior del robot. El ancho medio del túnel es de aproximadamente 6 metros, sin embargo se muestran solo 4 en el gráfico (eje x de -2 a +2). La posición real final del robot se representa con una cruz azul. Se observa que la anomalía presente en las imágenes no pre procesadas es eliminada en las imágenes con ecualización de histograma, para ambos casos de número de iteraciones. Para el caso ecualizado, la trayectoria es aproximadamente recta y presenta pequeñas ondulaciones en la coordenada X, y se acerca más al camino realizado por el robot en la realidad. La posición final del robot se encuentra cerca del punto objetivo en el caso ecualizado. Si bien en el caso no ecualizado también se da que la posición final está cerca del objetivo, al ver los gráficos de medias en la sección anterior, se concluye que la precisión alcanzada es coincidencia, dada

por el error introducido en la curva anómala. La desviación estándar es similar en ambos casos y existe una pequeña mejora dependiente de N para el caso ecualizado.

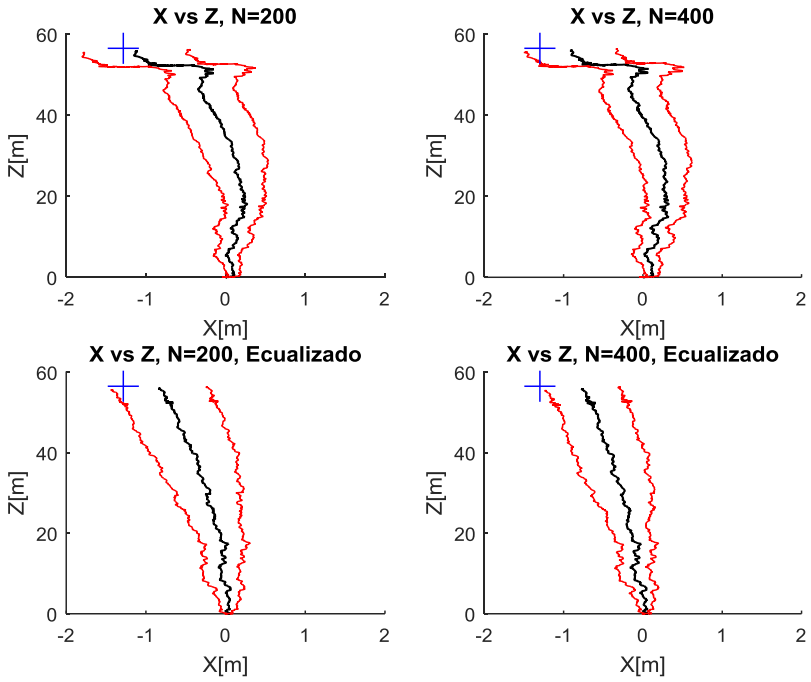


Ilustración 48: Comparación de trayectorias para N=200 y 400, ejes X y Z, para imágenes ecualizadas y no ecualizadas. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la Ilustración 49 se muestra la comparación de trayectorias para número N de iteraciones de RANSAC de 200 y 400, para la sección oscura del túnel, para los ejes Y (arriba y abajo) y Z (profundidad). Esto corresponde a una vista lateral del robot. La cercanía al punto objetivo es menor en el caso de la imagen ecualizada. La forma de la trayectoria más recta y horizontal para este caso también, lo que parece correcto pues el robot está moviéndose sobre suelo terroso, aproximadamente plano. No hay una mejora clara con respecto a la desviación estándar de un tipo de imagen con respecto al otro, ni de la influencia de N para los casos estudiados.

En las siguientes Ilustración 50, Ilustración 51 e Ilustración 52 se muestra la variación de la orientación del vehículo. El eje X representa el número de la imagen procesada, y el eje Y el ángulo en grados.

En la Ilustración 50 se muestra la orientación en el eje X, conocido como *Pitch*. La cercanía de la media al punto objetivo es parecida en ambos casos de estudio. La sección recta de la curva es más horizontal en el caso de la imagen con corrección de contraste. La cercanía del punto final al punto objetivo es similar en ambos casos, sin embargo, particularmente

para el paso de N 200 a 400 en la imagen ecualizada, se observa una tendencia a desviarse. La desviación estándar es similar en todos los casos mostrados.

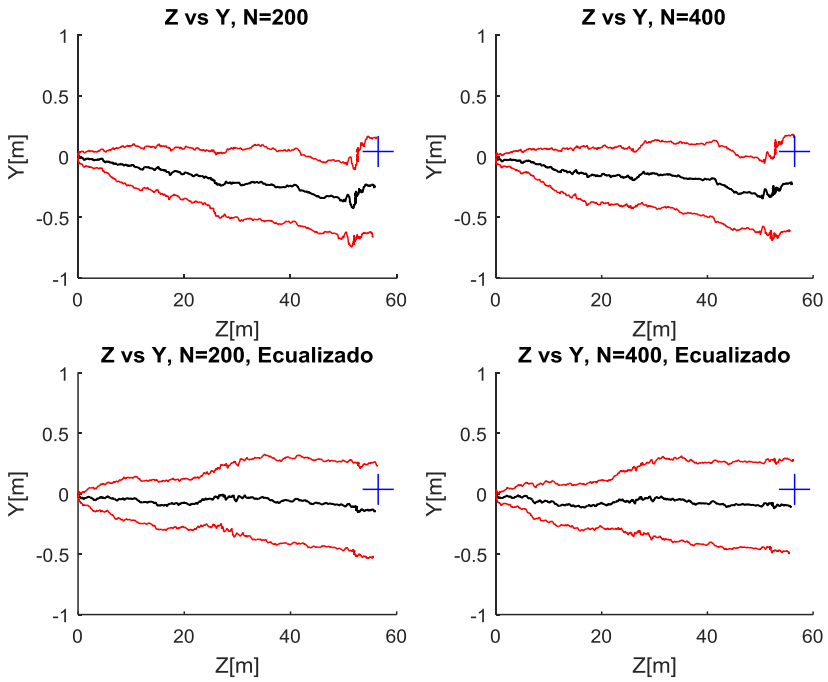


Ilustración 49: Comparación de trayectorias para N=200 y 400, ejes Z e Y, para imágenes ecualizadas y no ecualizadas. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

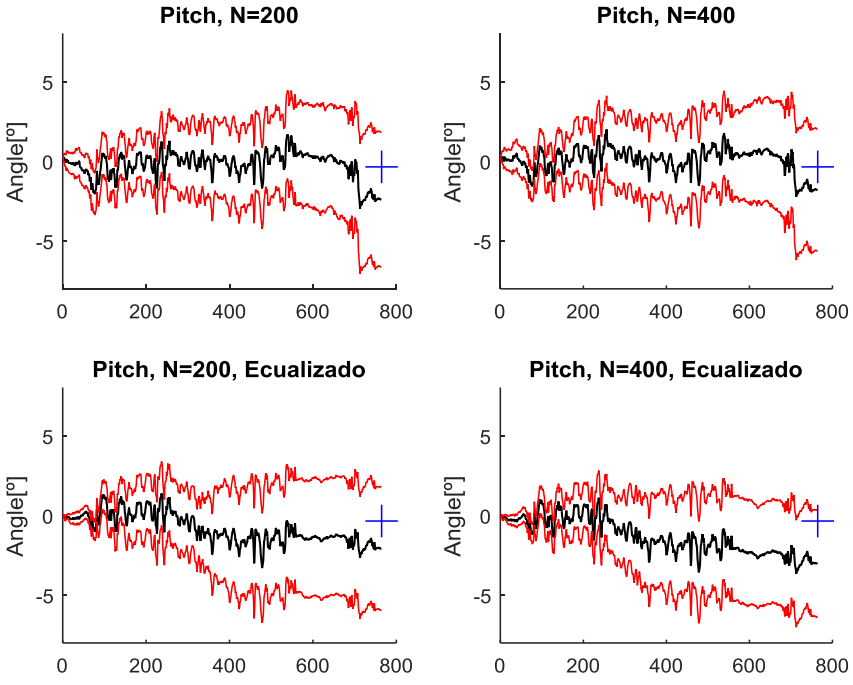


Ilustración 50: Comparación de orientación para N=200 y 400, Rotación en el eje *X*, para imágenes ecualizadas y no ecualizadas. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la Ilustración 51 se muestra la orientación en el eje Y, conocido como *Yaw*. El punto final de la media parece estar más acertado para el caso ecualizado, y parece mejorar con el aumento del número de iteraciones N. La parte recta de la curva mejora notoriamente al aplicar pre procesamiento, lo que explica la mejora en la cercanía del punto medio. No se observa una mejora clara en la desviación estándar dependiente de N.

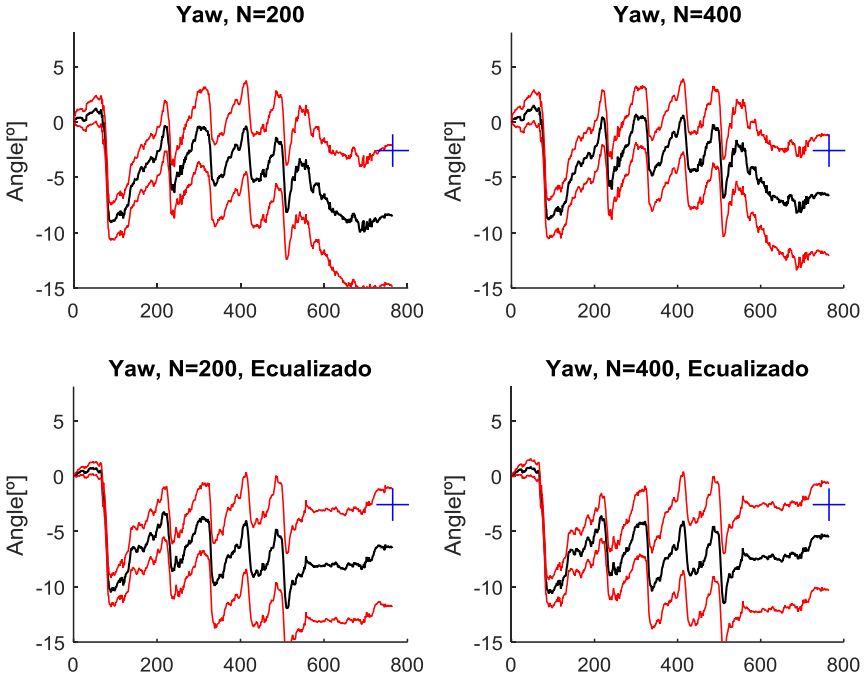


Ilustración 51 Comparación de orientación para N=200 y 400, Rotación en el eje *Y*, para imágenes ecualizadas y no ecualizadas. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la Ilustración 52 se muestra la orientación en el eje Z, conocido como *Roll*. Se observa, al igual que en los gráficos de *Pitch* y *Yaw*, una mejora en la parte plana de la curva al incorporar pre procesamiento en las imágenes. El punto final de la media se acerca bastante al punto final medido para el caso ecualizado. Ni la estabilidad de la solución ni la cercanía al punto objetivo mejora notoriamente para los casos expuestos en el gráfico con la variación del parámetro N.

Cabe destacar que para todos los ejes de orientación las características presentes en la curva se mantienen para ambos casos de estudio, solo que presentan desviaciones distintas. Esto da fe de que los picos y valles presentes en la solución no son ruido sino que son movimientos realizados por el robot.

Un resumen de todos los grafios expuestos anteriormente se encuentra en el Anexo B. Resumen de Gráficos de Resultados.

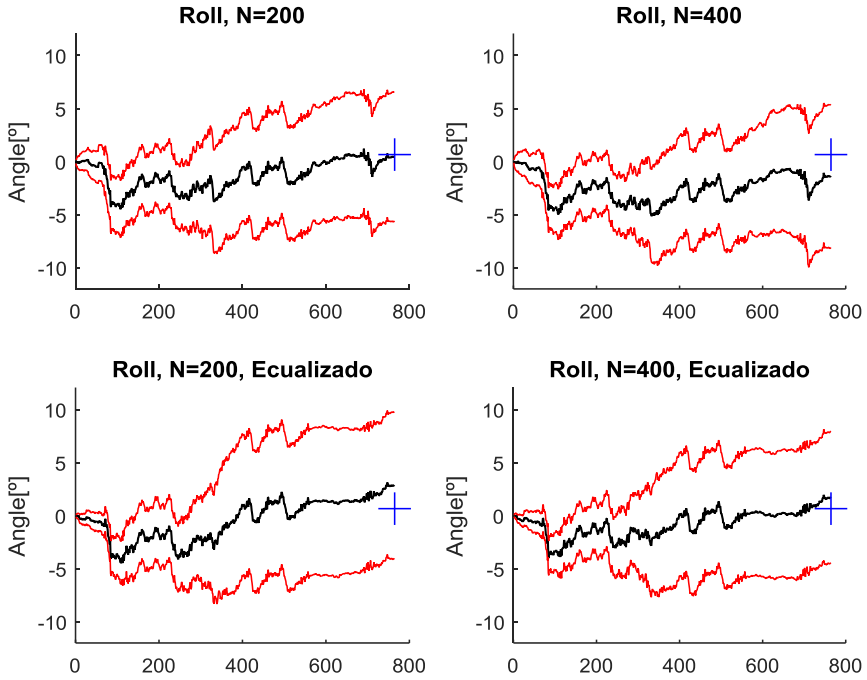


Ilustración 52 Comparación de orientación para N=200 y 400, Rotación en el eje *Z*, para imágenes ecualizadas y no ecualizadas. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

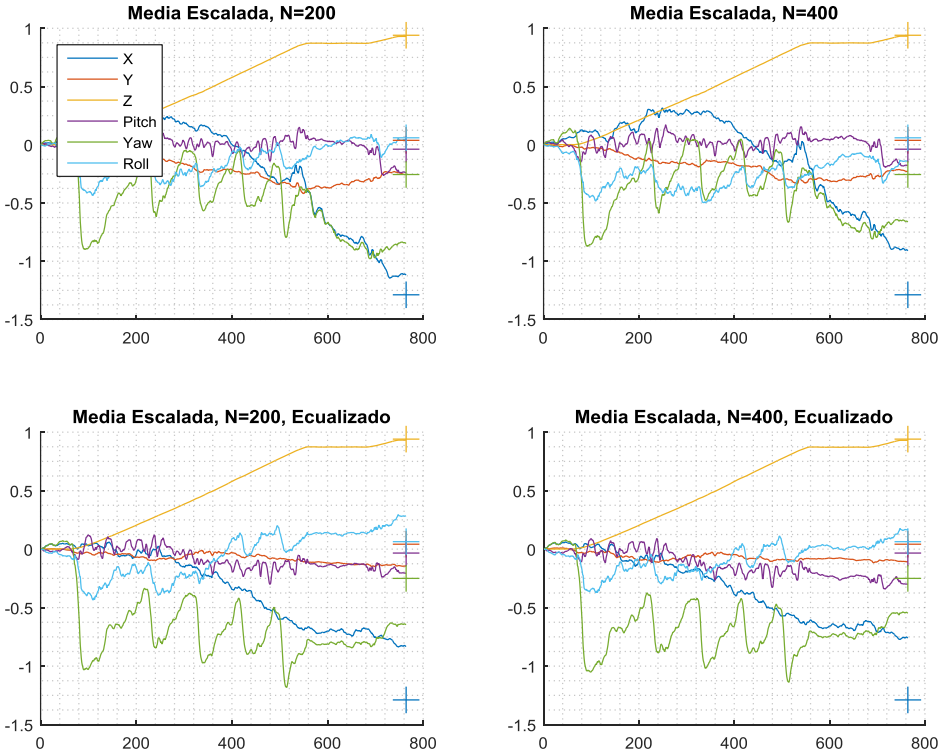


Ilustración 53: Comparación de la forma de la solución para N=200 y 400, para imágenes ecualizadas y no ecualizadas. Las unidades han sido escaladas. Las cruces representan la posicion real medida. Tramo oscuro.

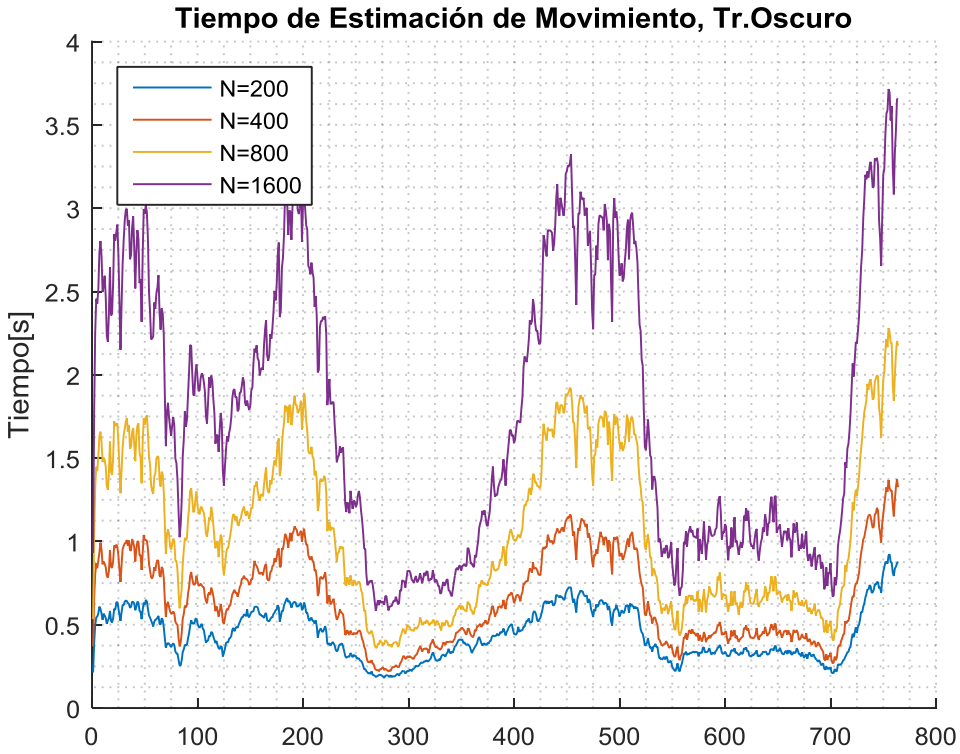


Ilustración 54: Comparación del tiempo de procesamiento de la etapa de estimación de movimiento, para imágenes ecualizadas.

En la Ilustración 53 se observa el gráfico de medias para los 6 grados de libertad del robot para los valores de N de 200 y 400, para los casos de imágenes pre procesadas y sin pre procesar. El eje X representa el número de la imagen procesada y el eje Y el valor de la media. Los valores han sido escalados para entrar en el segmento $[-1.5\ ,1\ ]$ para un mejor análisis. Primeramente no se destaca ninguna mejora para el eje Z, la cual no contiene ruido que se destaque a simple vista y está alineada perfectamente con la posición final para ambos casos. Para el tramo recto de la curva Z, las demás curvas también deben mantenerse rectas. Vemos que esto no se cumple para el caso sin pre procesar. Para el caso ecualizado se cumple a cabalidad, lo que mejora la calidad de los resultados.

Los tiempos totales de la etapa de estimación de movimiento para el caso de imágenes con ecualización de histograma quedan representados en la Ilustración 54. Se observa un incremento proporcional al número de iteraciones N. La forma de la curva es similar entre todos los casos. Se observa también que la solución presenta zonas de mayor tiempo de cómputo, al igual que en el caso base. También se observa una tendencia convergente en la curva. El tiempo medio para todos los casos es mayor que el tiempo medio del caso no ecualizado, lo cual está ligado al nuevo número de características presentes debido a la mejora en el contraste de la imagen.

Se concluye que el pre procesamiento de imágenes utilizando ecualización de histograma mejora en gran medida la calidad de la solución para el tramo oscuro del túnel debido a que

se disminuyen los errores que se producen en la etapa de emparejamiento y se aumenta el número de características encontradas en la imagen debido a la mejora de contraste.

4.3.3 Comparación del Parámetro de Ajuste d

En esta sección se muestran los resultados de aplicar el algoritmo RANSAC en 50 instancias diferentes para distinto número de iteraciones y de distancia d. Estos resultados son distintos unos de otros dada la naturaleza no determinista del algoritmo. Los parámetros utilizados se muestran en la Tabla 10.

Tabla 10: Parámetros de prueba del algoritmo RANSAC.

Parámetro	**Valor**			
Núm. de puntos	RANSAC-5			
d	0.07	0.05	0.03	0.01
N	200	400	800	1600

Debido a la mejora alcanzada en la sección anterior, se emplea el método de ecualización de histograma a todas las imágenes, ya sean de la sección oscura o la iluminada.

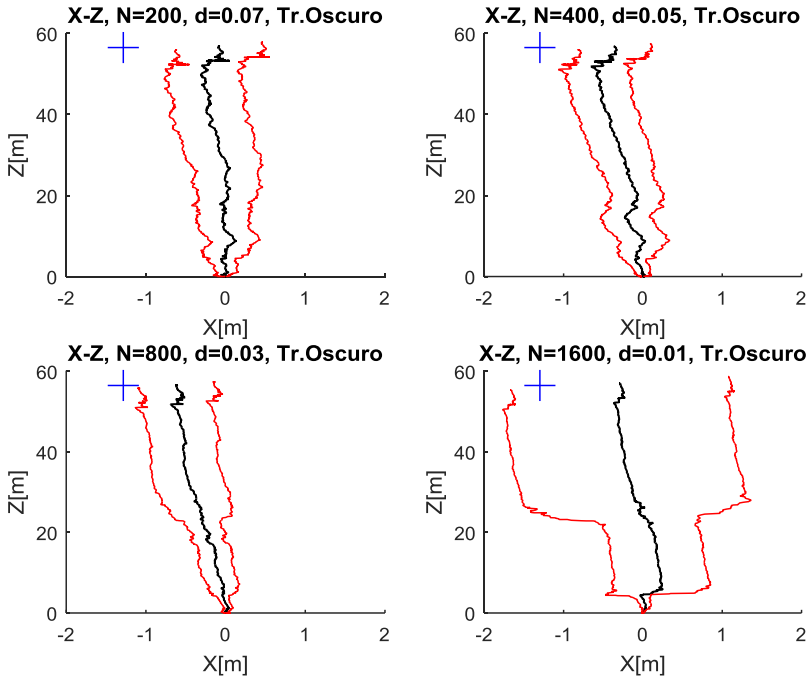


Ilustración 55: Comparación de trayectorias para N de 200 a 1600, y d de 0.07 a 0.01, ejes X y Z. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

Se realiza el análisis para los tramos oscuro e iluminado del túnel en conjunto, a diferencia de en las secciones anteriores. Los valores de d están emparejados con los valores de N de la forma siguiente: para valores de N de 200, 400, 800 y 1600 se asocian valores de d de 0.07, 0.05, 0.03 y 0.03 respectivamente.

En la Ilustración 55 e Ilustración 56 se muestra la comparación de trayectorias para distinto número N de iteraciones de RANSAC, de 200 a 1600 y d de 0.07 a 0.01, para ambas secciones del túnel. En estos gráficos se encuentra la información de 50 realizaciones del algoritmo, representados por la media (línea negra) y la desviación estándar (línea roja). El eje X corresponde a la coordenada X del robot en el túnel (izquierda y derecha), y el eje Y representa la coordenada Z (profundidad). Esto corresponde a una vista superior del robot. El ancho medio del túnel es de aproximadamente 6 metros, sin embargo se muestran solo 4 en el gráfico (eje x de -2 a +2). La posición real final del robot se representa con una cruz azul.

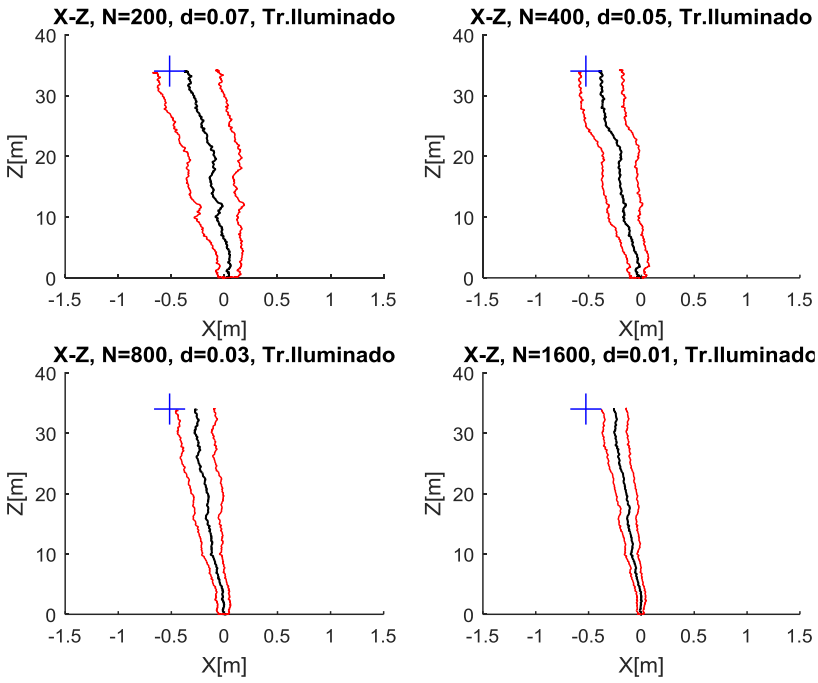


Ilustración 56: Comparación de trayectorias para N de 200 a 1600, y d de 0.07 a 0.01, ejes X y Z. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la sección oscura del túnel, se observa que la desviación estándar aumenta con la disminución del número de distancia d del algoritmo RANSAC, con un aumento prominente para $N = 1600$. Se observan secciones de gran incremento en este último caso. Esto puede explicarse por la poca cantidad de *inliers* existentes debido al parámetro de distancia, y la alta susceptibilidad a errores por este motivo. Este comportamiento no altera mucho la trayectoria media, que se ve aproximadamente recta en todos los casos, presentando pequeñas anomalías. Con respecto al punto objetivo, ningún resultado se

acerca convincentemente. En el eje Z se observa una solución bastante estable, en el sentido de que en todos los casos se llega al objetivo.

En la sección iluminada la desviación estándar, y por lo tanto la estabilidad de la solución, se comportan de manera inversa al caso oscuro. Esta disminuye con la disminución de d y aumento de N. La posición media sin embargo se aleja de la posición objetivo. La solución parece tener mejor calidad en las configuraciones de menor d .

En la Ilustración 57 e Ilustración 58 se muestra la comparación de trayectorias para distinto número N de iteraciones de RANSAC, de 200 a 1600 y d de 0.07 a 0.01, para ambas secciones del túnel, para los ejes Y (arriba y abajo) y Z (profundidad). Esto corresponde a una vista lateral del robot. Se observa un caso similar al anterior, donde en la sección oscura la desviación estándar aumenta a medida que disminuye d, y en la sección iluminada disminuye con la disminución de d. Se observan las mismas secciones de incremento en el caso de $N = 1600$. En el caso del eje Y, el punto medio parece estar mucho más cercano al punto objetivo, sobre todo para la sección oscura del túnel.

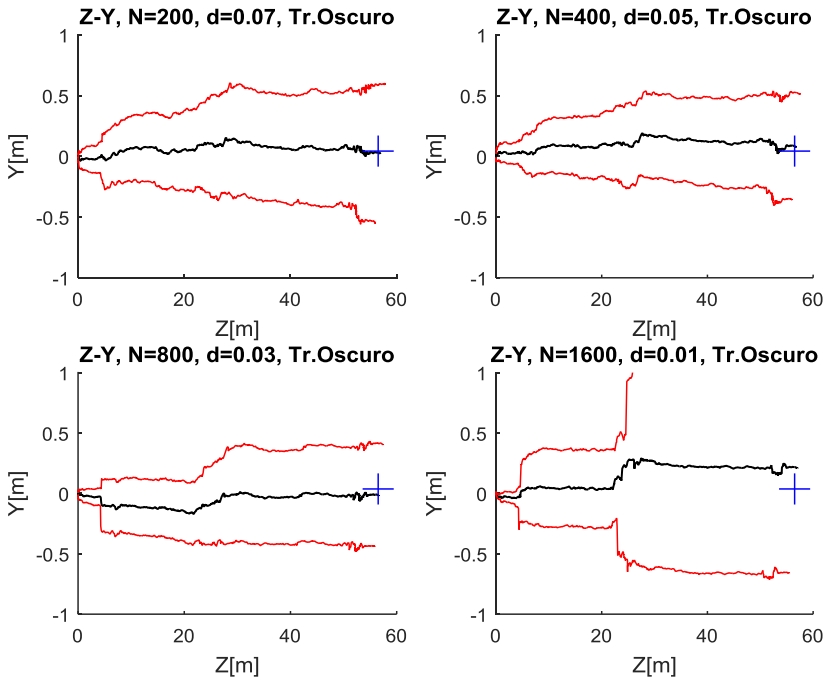


Ilustración 57: Comparación de trayectorias para N de 200 a 1600, y d de 0.07 a 0.01, ejes Z e Y. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

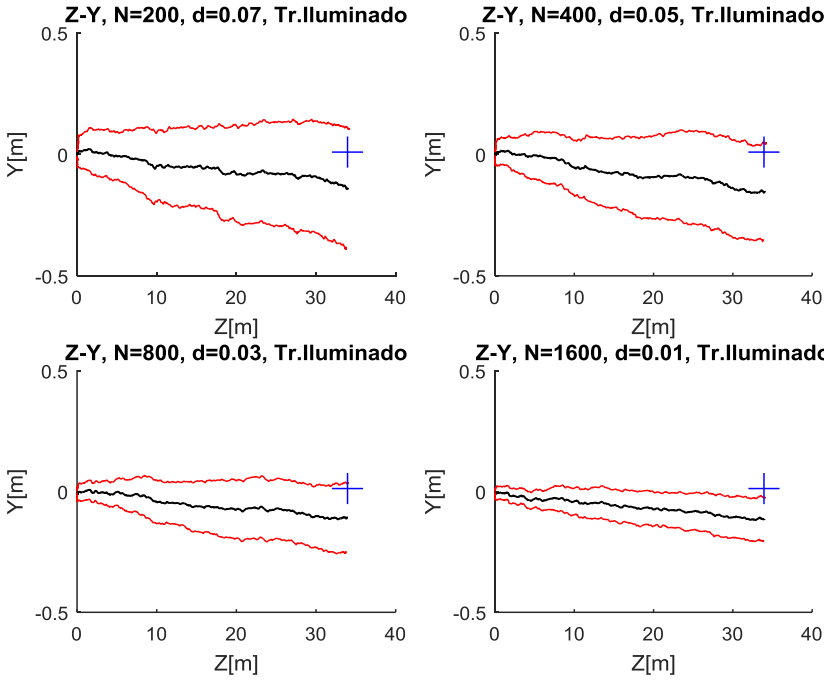


Ilustración 58: Comparación de trayectorias para N de 200 a 1600, y d de 0.07 a 0.01, ejes Z e Y. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En las siguientes Ilustración 59, Ilustración 60, Ilustración 61 e Ilustración 62 se muestra la variación de la orientación del vehículo. El eje X representa el número de la imagen procesada, y el eje Y el ángulo en grados.

En la Ilustración 59 se muestra la orientación en el eje X, conocido como *Pitch*. La cercanía de la media al punto objetivo es menor para el caso de $N = 200$ y 800, para el caso de la sección oscura del túnel. Para la sección iluminada, todos los casos están suficientemente cercanos. Para ambas secciones del túnel la estabilidad parece mejorar en pequeña escala. Para el caso oscuro, se puede ver que la sección plana de la curva se hace más recta con menor distancia d, alcanzando además completa horizontalidad en $N = 1600$. No se aprecia a simple vista un aumento descontrolado de la desviación estándar, como en el caso del eje X e Y.

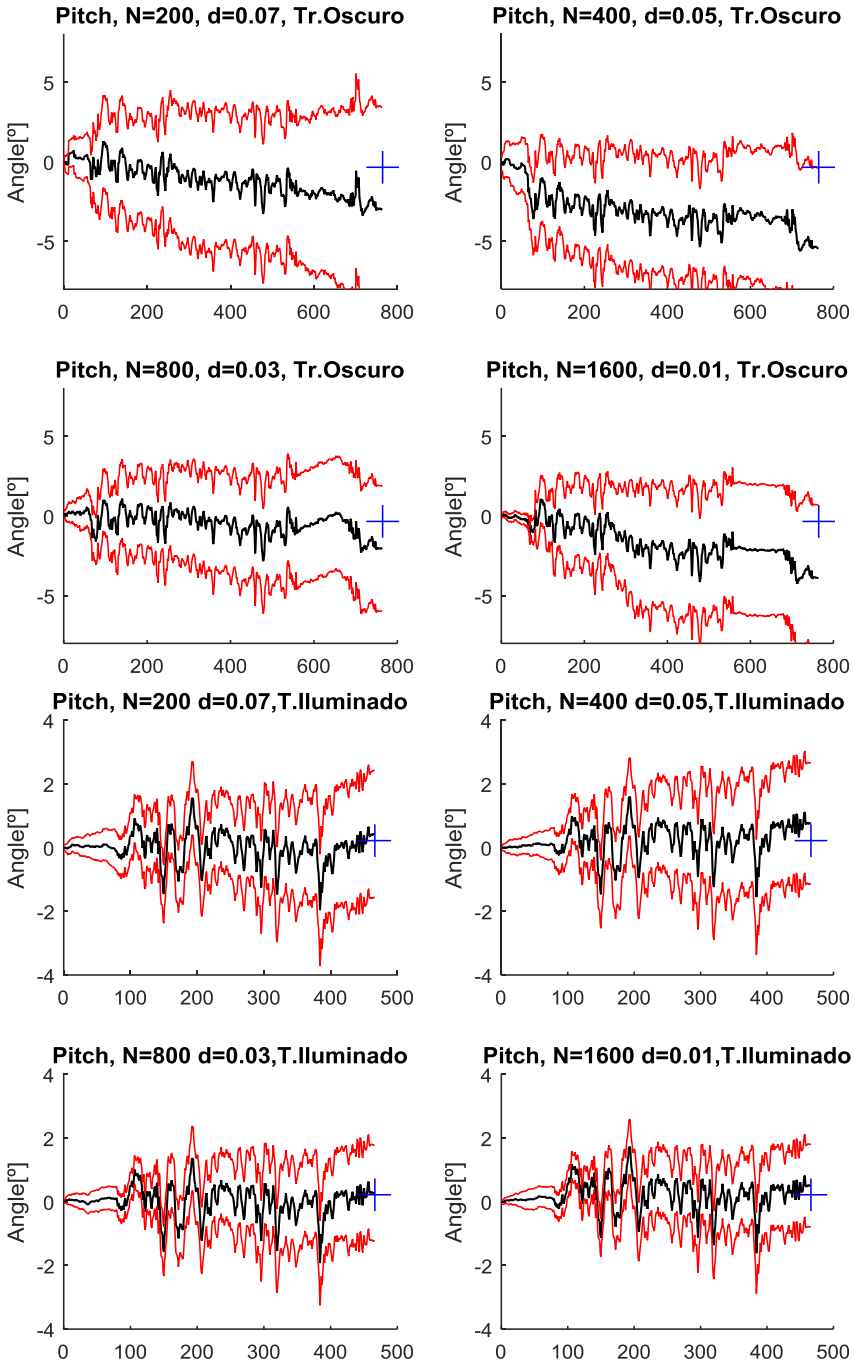


Ilustración 59: Comparación de orientación para N de 200 a 1600, y *d* de 0.07 a 0.01, rotación en el eje X. Tramo oscuro e iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En la Ilustración 60 e Ilustración 61 se muestra la orientación en el eje Y, conocido como *Yaw*. La cercanía de la media al punto objetivo va disminuyendo con la disminución de d, para el caso de la sección oscura del túnel. Para la sección iluminada, la cercanía del punto medio parece mantenerse dentro de los mismos intervalos para todos los casos. La estabilidad de la solución mejora en el caso de la sección iluminada, pero no queda claro en este grafico si mejora para la sección oscura.

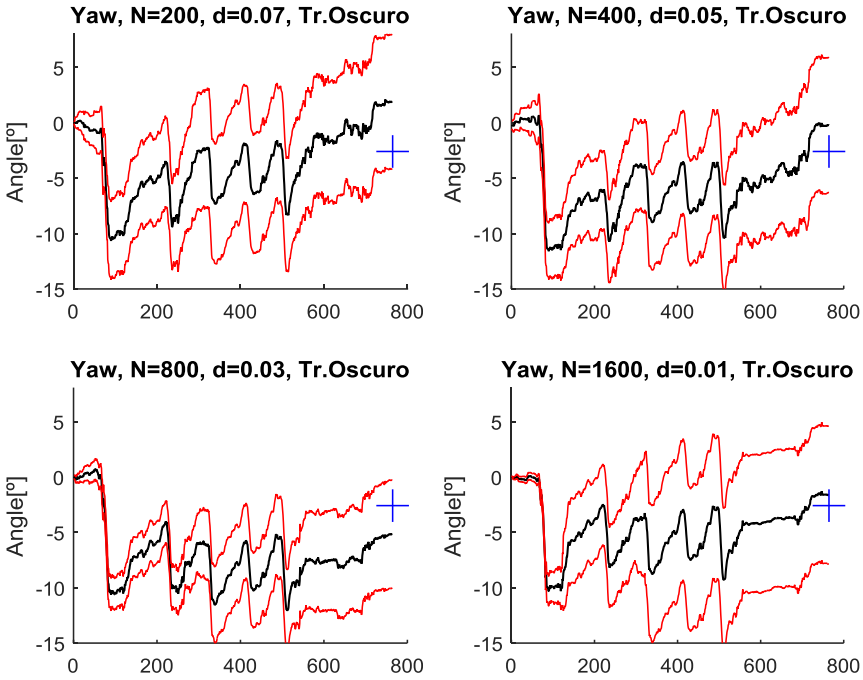


Ilustración 60 Comparación de orientación para N de 200 a 1600, y d de 0.07 a 0.01, rotación en el eje Y. Tramo oscuro. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

Al igual que para *Pitch*, en el caso oscuro la sección plana de la curva es más recta y horizontal en el caso $N = 1600$. En el caso iluminado, también se nota la parte plana al principio de la curva, pero está bien definida para todos los casos. Si bien la curva de la media sufre desviaciones hacia arriba o hacia abajo, su forma característica permanece constante para todos los casos de estudio.

En la Ilustración 62 se muestra la orientación en el eje Z, conocido como *Roll*. La cercanía de la media al punto objetivo va aumentando con la disminución de d, para el caso de la sección oscura del túnel.

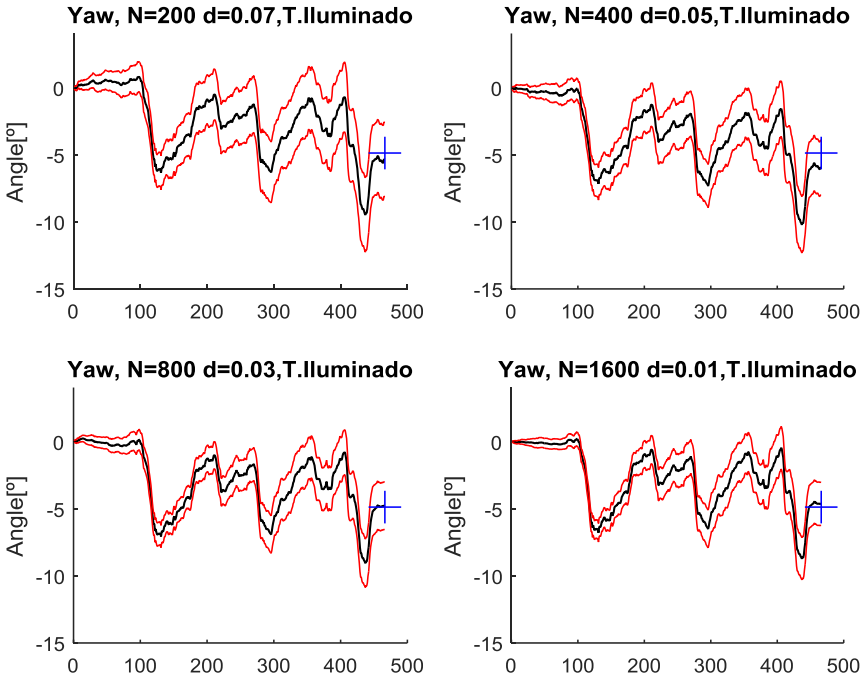


Ilustración 61: Comparación de orientación para N de 200 a 1600, y d de 0.07 a 0.01, rotación en el eje Y. Tramo iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

Para la sección iluminada, la cercanía del punto medio parece no variar consistentemente con los parámetros. La estabilidad de la solución mejora en pequeña escala en el caso de la sección iluminada, pero no queda claro en este grafico si mejora para la sección oscura. Al igual que para *Pitch* y *Yaw*, en el caso oscuro la sección plana de la curva es más recta y horizontal en el caso $N = 1600$. En el caso iluminado, también se nota la parte plana al principio de la curva, pero está bien definida para todos los casos, al igual que en *Yaw*. Si bien la curva de la media sufre desviaciones hacia arriba o hacia abajo, su forma característica permanece constante para todos los casos de estudio.

Un resumen de todos los grafios expuestos anteriormente se encuentra en el Anexo B. Resumen de Gráficos de Resultados.

En la Ilustración 63 e Ilustración 64 se comparan las desviaciones estándar de un cada eje y orientación para todos los pares de valores de N y d.

Para los ejes X, Y, y Z, presentados en la Ilustración 63, se aprecia de mejor manera la tendencia a la baja con el aumento de N. A diferencia de los resultados de la sSección 4.3.1es posible ver esta tendencia claramente en el tramo iluminado del túnel. La forma de la curva, para todos los casos de la sección oscura, presenta forma de escalera, lo que puede provenir del aumento en la dificultad del reconocimiento y emparejamiento de características a medida que se atraviesa el túnel, pues se encuentra en un entorno con escasa luminosidad, con luz variable. Es posible también ver el aumento desproporcionado

de la desviación estándar para el caso $N = 1600$, donde se ve que aumenta en el inicio de la zona escalonada. Cabe destacar que para la sección iluminada se alcanzan valores de hasta 0.1 metros, es decir 10 centímetros.

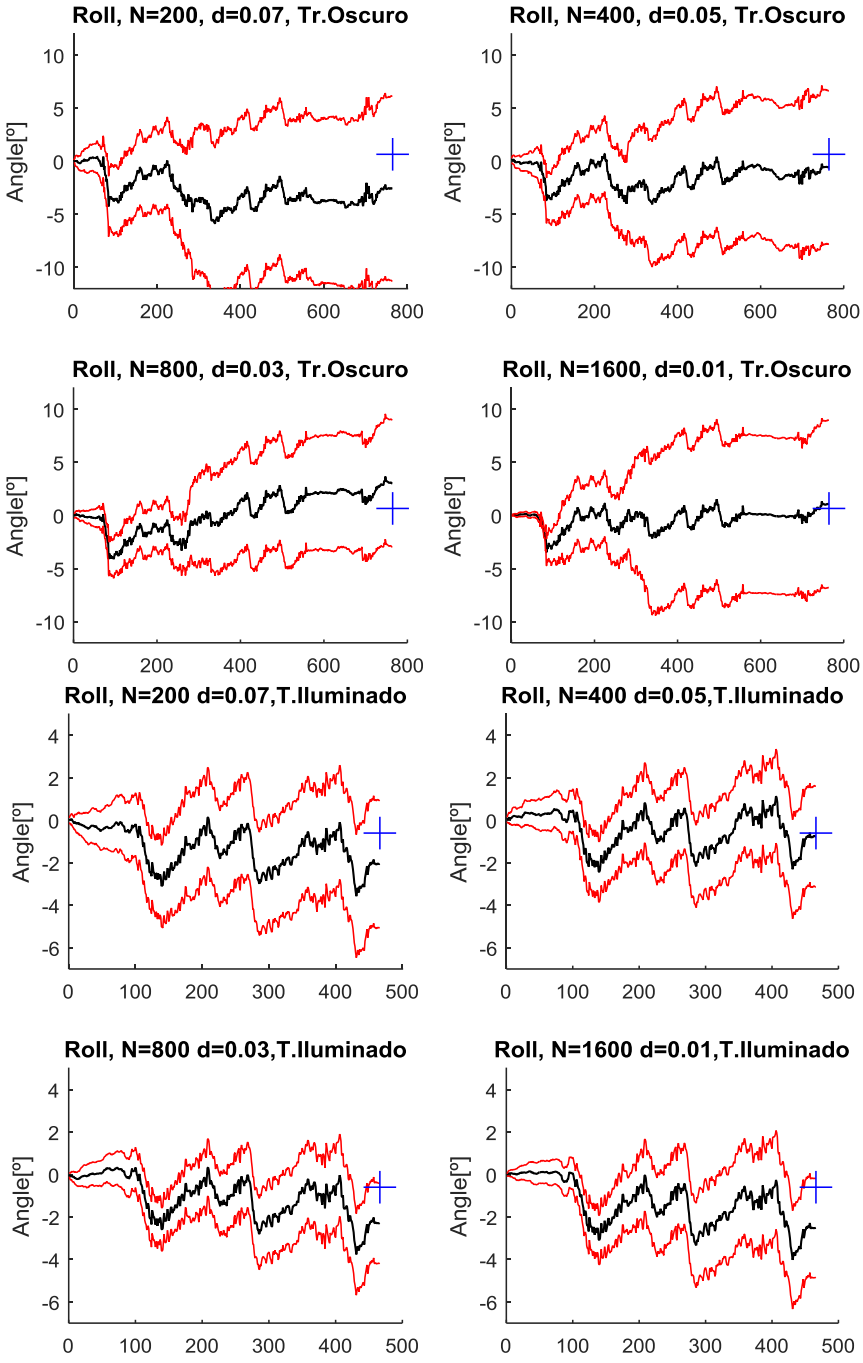


Ilustración 62: Comparación de orientación para N de 200 a 1600, y *d* de 0.07 a 0.01, rotación en el eje Z. Tramo oscuro e iluminado. La cruz azul representa la posición final verdadera, la línea negra representa la media, y la roja la desviación estándar.

En el caso de los gráficos de orientación, mostrados en la Ilustración 64, la tendencia a la mejora no es tan explícita, sin embargo está presente. Se observa el mismo aumento escalonado para la sección oscura del túnel, sin embargo no existe un aumento desproporcionado como en el caso anterior, es decir, el algoritmo se comporta de manera más flexible al calcular la orientación del vehículo. Cabe destacar que para la sección iluminada se alcanzan valores de hasta 0.02 grados, lo cual parece ser lo suficientemente preciso para la navegación.

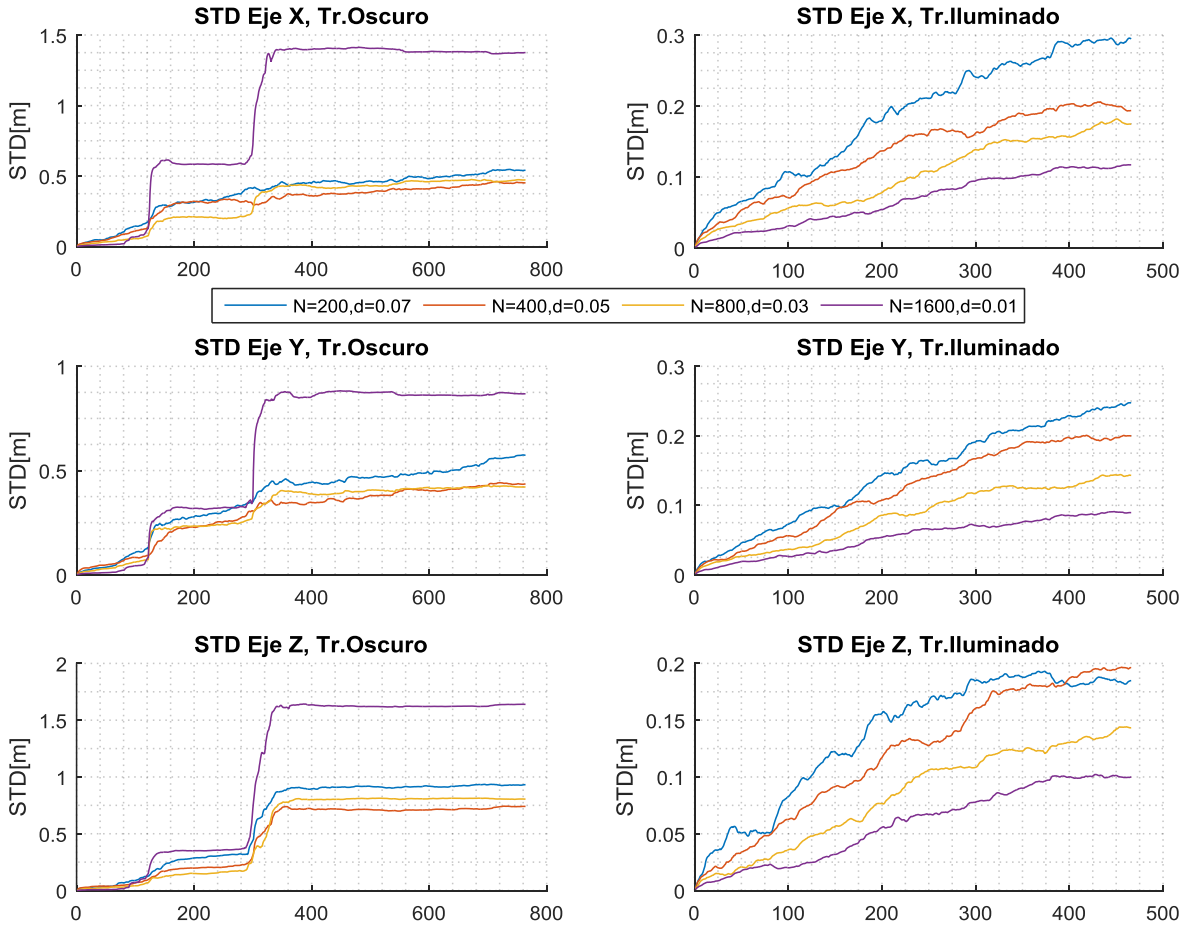


Ilustración 63: Comparación de desviación estándar para los distintos valores de N y *d*, para los ejes X, Y y Z. Tramo oscuro e iluminado.

En la Ilustración 65 se observa el gráfico de medias para los 6 grados de libertad del robot. El eje X representa el número de la imagen procesada y el eje Y el valor de la media. Los valores han sido escalados para entrar en el segmento [-1.5 ,1] para un mejor análisis. Primeramente destaca la forma de la curva para el eje Z, la cual no contiene ruido que se destaque a simple vista y está alineada perfectamente con la posición final, al igual que en las secciones 4.3.1y 4.3.2. En estos gráficos se observa de mejor forma la situación de la curva plana, donde es predominante en los casos de mayor numero de iteraciones y menor distancia d para el algoritmo de RANSAC, en ambas secciones del túnel. La forma característica de las curvas es consistente en todos los casos, sufriendo pequeñas

desviaciones en su valor medio. La distancia entre la posición final media del robot y la posición objetivo medida varía dependiendo de la sección del túnel y d. Sin embargo no queda claro de si existe una proporcionalidad de esta distancia con el parámetro d.

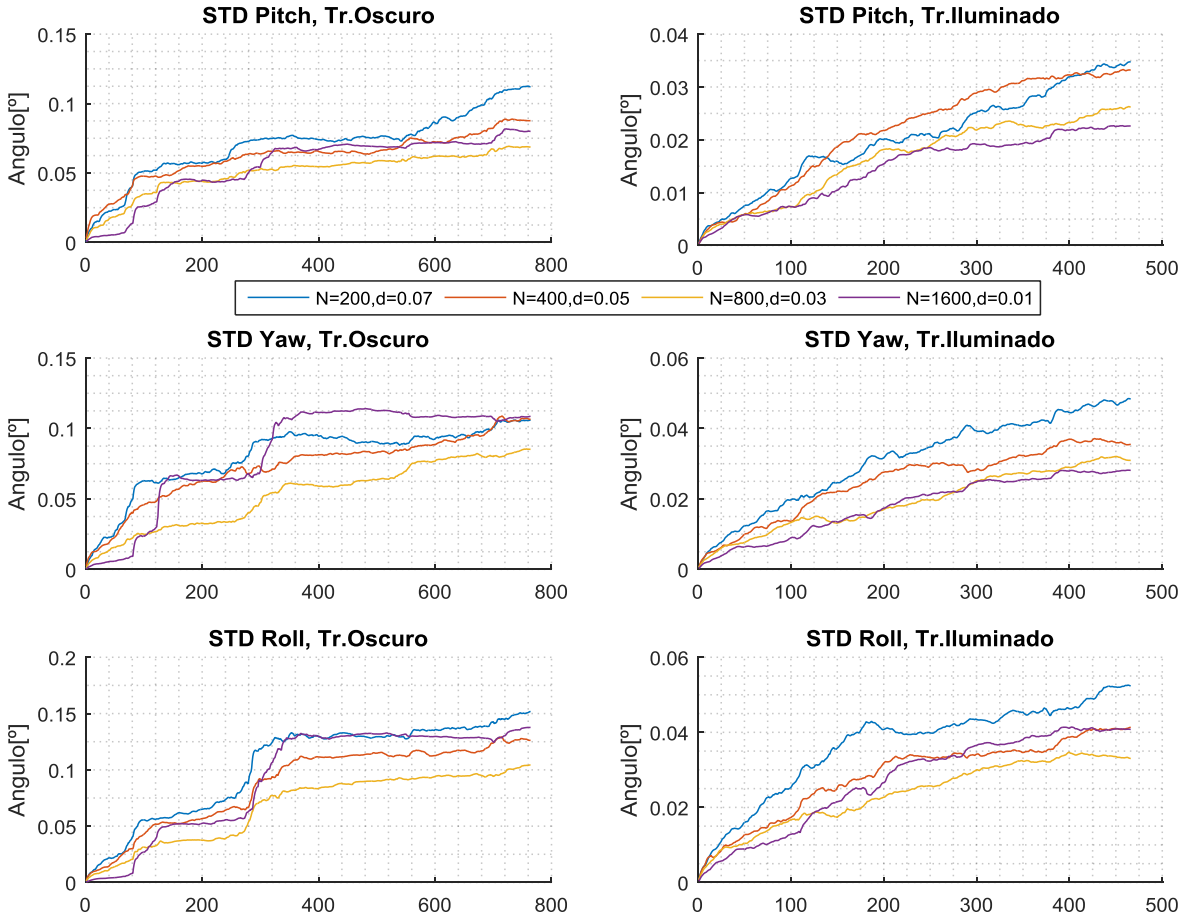


Ilustración 64: Comparación de desviación estándar para los distintos valores de N y d, para los la orientacion del robot, Pitch, Yaw y Roll. Tramo oscuro e iluminado.

Se concluye que la disminución de d en presencia de u aumento del número de iteraciones N contribuye a la disminución de la desviación estándar para el caso de la sección iluminada del túnel. Sin embargo una baja en la cantidad de características encontradas, en la calidad del emparejamiento puede introducir una cantidad muy elevada de error, como se ve para el resultado de la sección oscura.

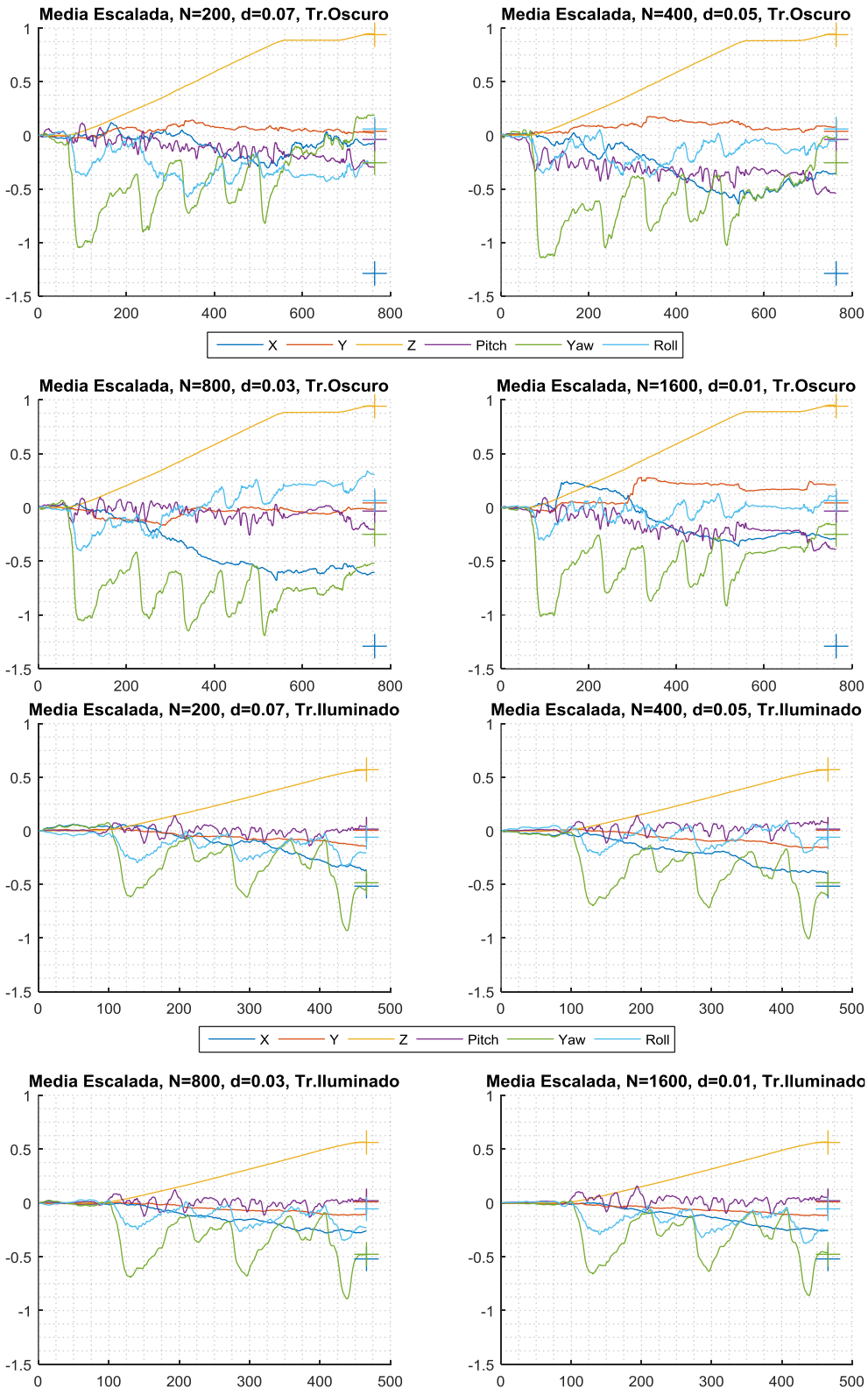


Ilustración 65: Comparación de la forma de la solución para todos los valores de N y *d*. Las unidades han sido escaladas. Las cruces representan la posicion real medida. Tramo oscuro e iluminado.

4.3.4 Comparación de la Ubicación de las Características

En esta sección se muestran los resultados de aplicar el algoritmo RANSAC en 3 instancias diferentes utilizando filtro de datos según la profundidad de estos. Estos resultados son distintos unos de otros dada la naturaleza no determinista del algoritmo. Los parámetros del filtro ocupados en cada instancia se muestran en Tabla 11, y fueron deducidos utilizando la Ecuación (22).

Tabla 11: Intervalos de filtro de distancias según parámetro d.

Filtro	**Intervalo d (px)**	**Intervalo d (m)**
Cercano	[8.11 , 486.4]	[0.5 , 30]
Lejano	[2.43 , 8.11]	[30 , 100]

Los parámetros de RANSAC utilizados en este experimento se muestran en la Tabla 12.

Tabla 12: Parámetros de RANSAC utilizados en el experimento.

Parámetro	**Valor**
Núm. de puntos	RANSAC-5
d	0.1
N	1000

Los resultados del experimento correspondientes al tramo oscuro Tr_1 se muestran en la Ilustración 66. Se descarta por simple vista la solución que utiliza solamente los puntos lejanos en la imagen. Con respecto a las otras soluciones, a simple vista se ve que tienden a llegar al mismo punto, sin embargo su trayectoria presenta mayores variaciones para el caso sin filtro.

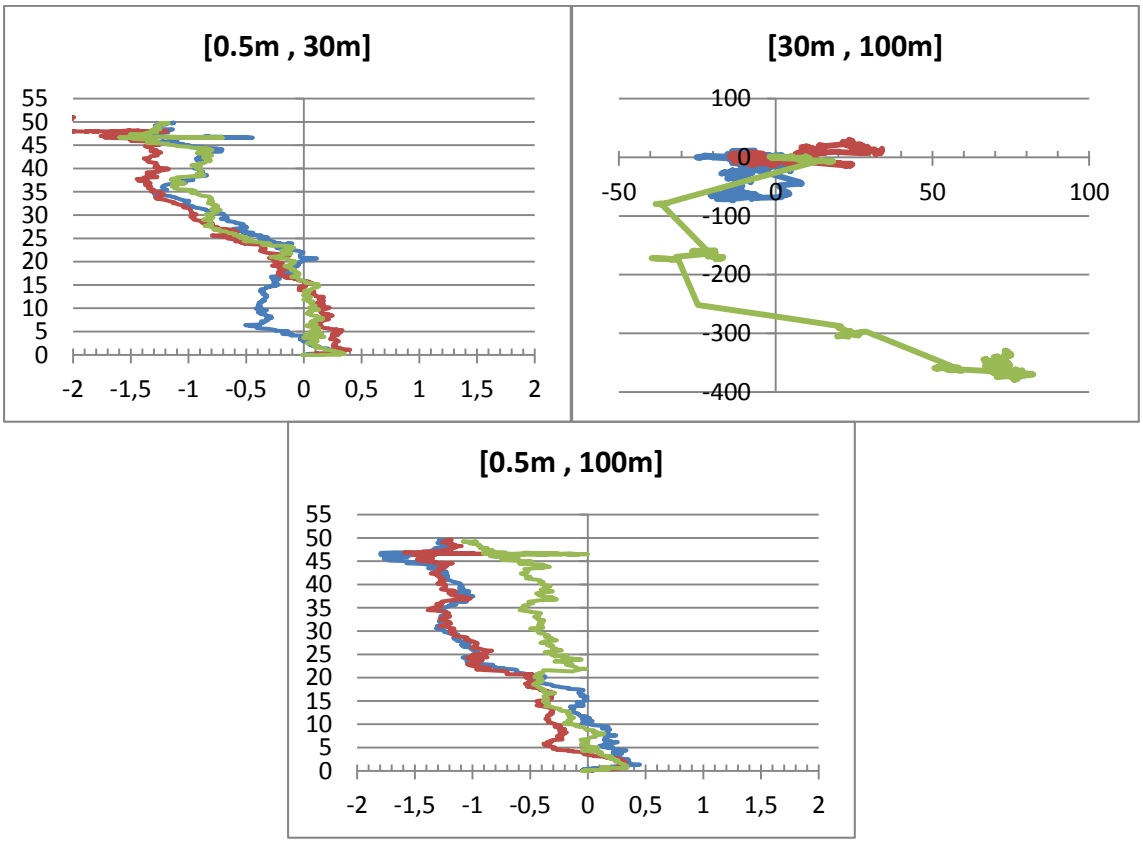


Ilustración 66: Resultados del experimento para distintos intervalos de profundidad. Coordenadas X horizontal y Z vertical, en metros. Tramo oscuro.

Se observa que la solución utilizando los puntos cercanos al vehículo tiene en promedio una desviación estándar menor para el eje X comparados con la opción que utiliza todos los puntos. Para el eje Z se observa un tramo en que el resultado es mejor, sin embargo luego de un punto de inflexión en la curva este resultado se invierte la mayor parte del tramo, siendo mejor quitar el filtro. La causa del punto de inflexión es desconocida, pero se presume que hubo un cambio brusco en las condiciones de iluminación.

Los resultados del experimento correspondientes al tramo iluminado Tr_2 se muestran en la Ilustración 67. Se descarta nuevamente por simple vista la solución que utiliza solamente los puntos lejanos en la imagen. Con respecto a las otras soluciones, a simple vista se ve que tienden a tomar una trayectoria parecida. La solución correspondiente al filtro cercano posee una mayor estabilidad.

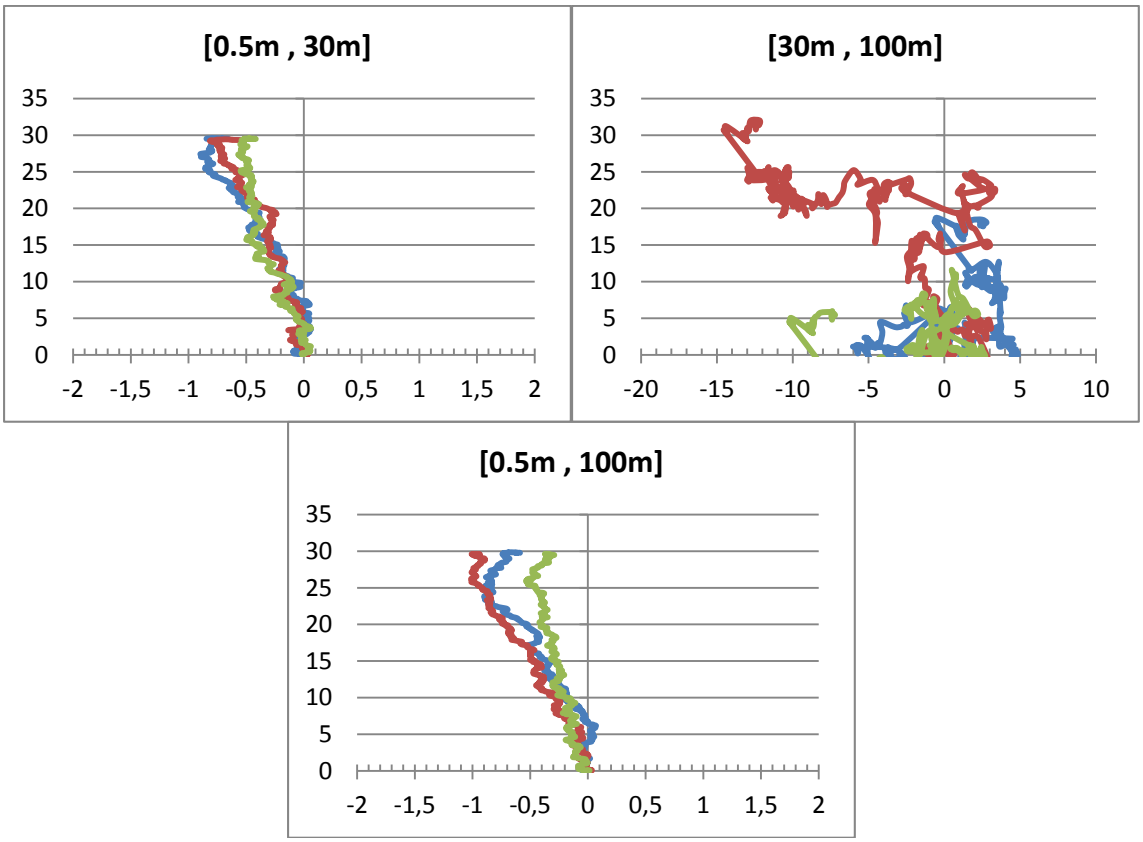


Ilustración 67: Resultados del experimento para distintos intervalos de profundidad. Coordenadas X horizontal y Z vertical, en metros. Tramo iluminado.

Con respecto a las desviaciones estándar de los resultados, los cuales se pueden apreciar en la Ilustración 68 e Ilustración 69, se observa que la solución utilizando los puntos cercanos al vehículo es en promedio menor para ambos ejes. El resultado para el eje *Z* encontrado en el caso del tramo oscuro no se repite en el tramo iluminado, lo que refuerza la hipótesis del punto de inflexión encontrado en los resultados para aquella sección.

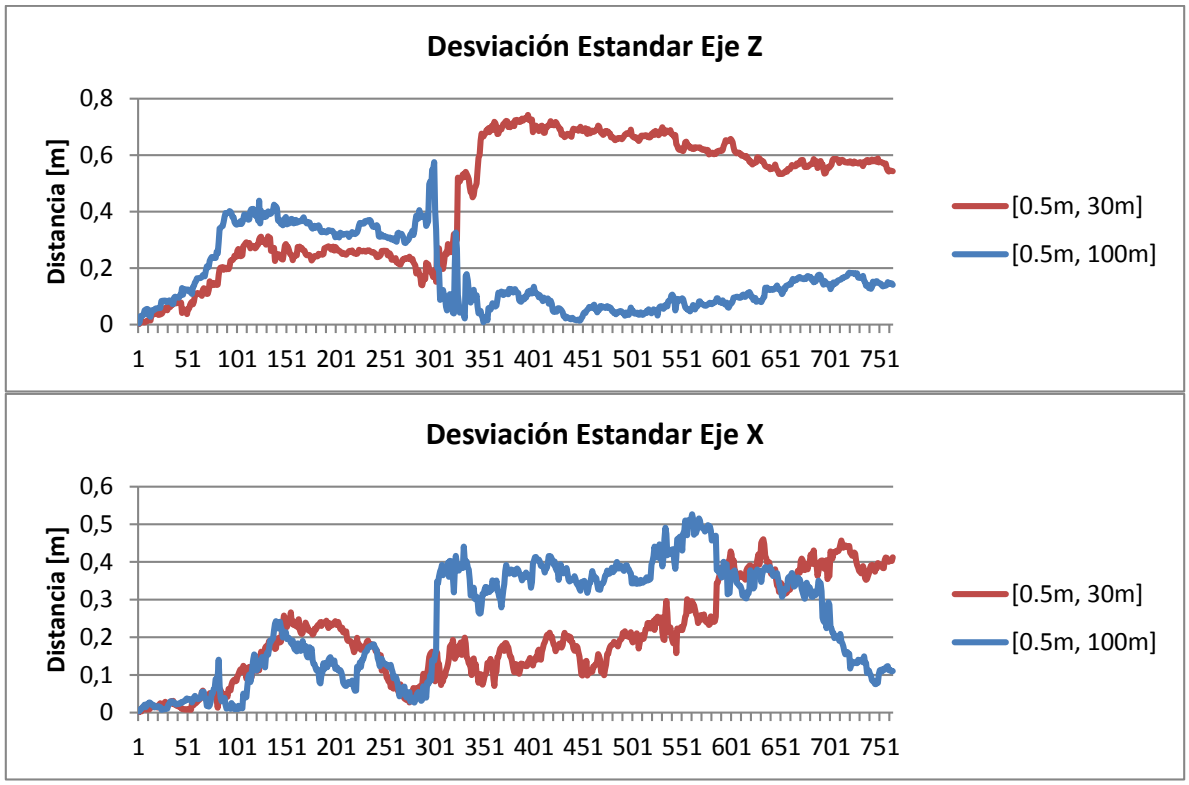


Ilustración 68: Desviación estándar para el caso cercano y el caso total. Tramo oscuro.

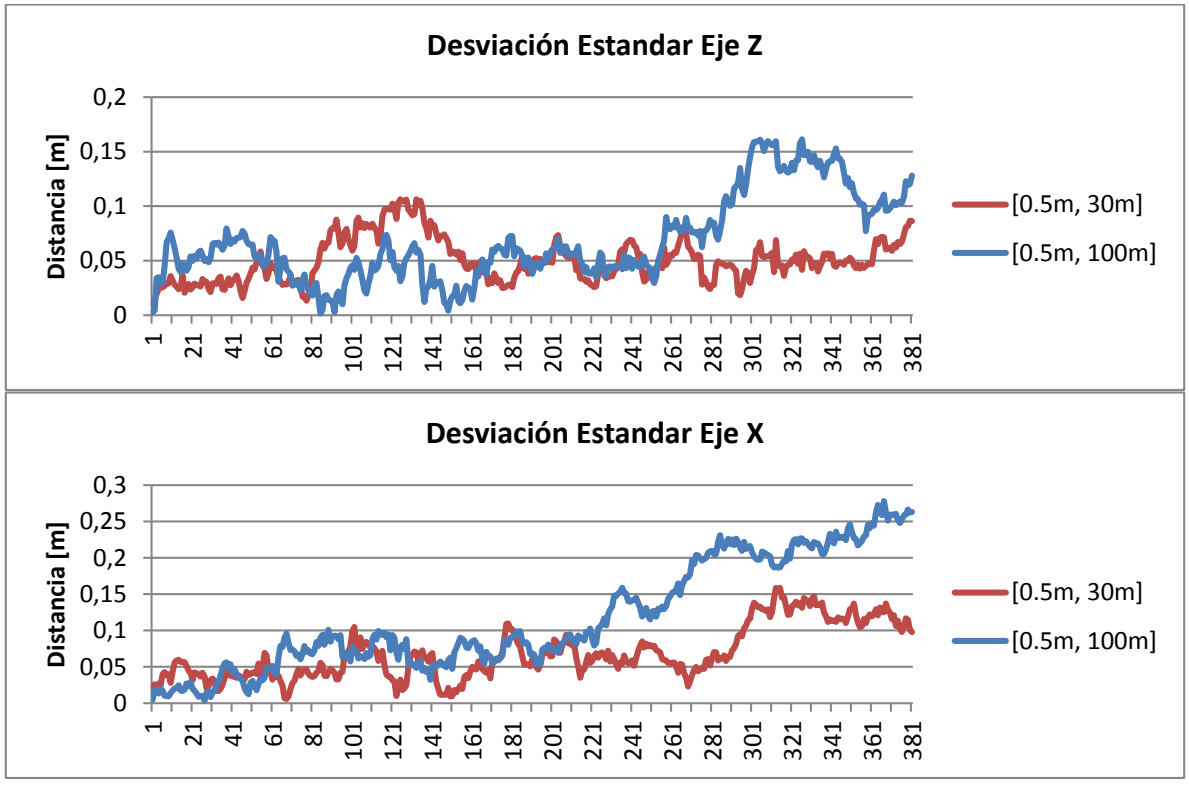


Ilustración 69: Desviación estándar para el caso cercano y el caso total. Tramo iluminado.

4.4 Resultados sin RANSAC

En la Ilustración 70 se observan los resultados de aplicar el algoritmo sin realizar RANSAC, al tramo iluminado. Se observa en el grafico izquierdo que la solución diverge completamente, escapándose de los rangos máximos permitidos $X = -0.5m$ e $Z = 34m$, mencionados en la Tabla 6. Sin embargo al aplicar el filtro de datos para puntos cercanos, se obtiene un resultado que, si bien no está a la par de los resultados obtenidos en la sección de RANSAC, es mucho más satisfactorio, pues esta al menos dentro de los rangos del interior del túnel.

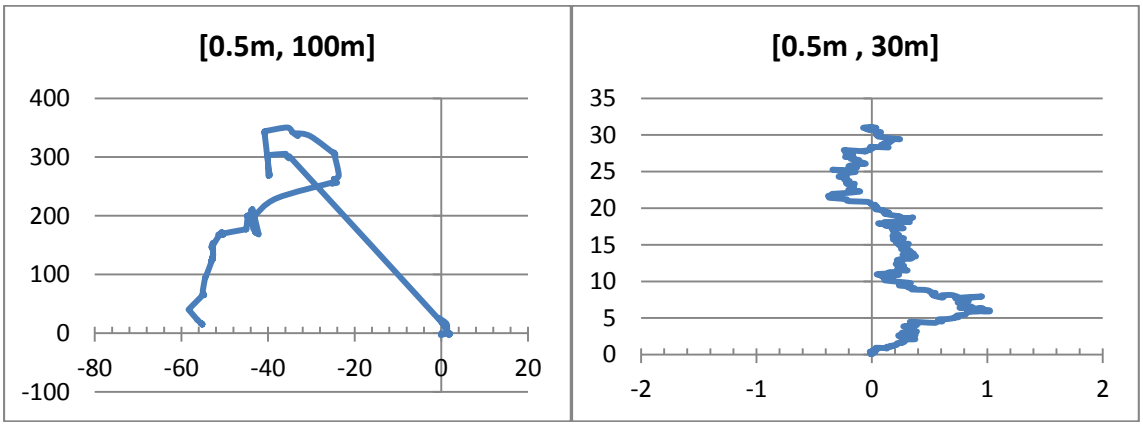


Ilustración 70: Resultados de aplicar el algoritmo sin realizar RANSAC, para datos filtrados por distancia, y sin filtrar. Tramo iluminado.

Se concluye que para el caso de un túnel, los datos recolectados correspondientes a puntos lejanos son en su mayoría *outliers*.

Capítulo 5: Conclusiones y Recomendaciones

El principal objetivo de este trabajo era el de implementar el sistema de odometría visual para ayudar al robot a estimar su posición en un entorno estático utilizando la cámara estereoscópica. Este objetivo se cumple completamente, sin embargo cabe notar que la velocidad del algoritmo fue más lenta de lo esperado debido a limitaciones que están ligadas tanto a la implementación en si del software, como también al hardware del robot.

Se logró hacer una triangulación en las imágenes de la cámara que logró añadir información de profundidad a las imágenes capturadas por esta. La calidad de la información en el eje Z fue de excelente calidad, como queda demostrado en los resultados de la sección 4.3.

Se logró la implementación de un sistema extensible que permita intercambiar partes del algoritmo para probar distintos métodos en el futuro. Este es fácil de configurar para distintos tipos de cámara y podría eventualmente ser utilizado, con algunas modificaciones, en distintos tipos de robots.

Si bien se utilizó ROS para extraer la información previamente grabada por las cámaras, no se logró implementar el algoritmo para leer las imágenes directamente de la cámara dentro de ROS, pues no se pudo lograr ejecución en tiempo real. Sin embargo la implementación actual facilito la prueba con distintos métodos de cálculo y distintos valores de parámetros, cosa que para el caso de tiempo real habría sido un proceso mucho más lento.

Los resultados obtenidos en la Sección 4.3.1 y 4.3.3 permiten concluir que el aumento del número de iteraciones N del algoritmo de RANSAC contribuye a la estabilidad de la solución, y por lo tanto a un mejor resultado. Con respecto al parámetro de distancia d, que establece el límite de la zona de puntos buenos y malos, se encontró que posee una gran influencia en el parámetro ϵ, la contaminación de los datos. Esto no solo hace que crezca N con la disminución de d y desestabilice la solución, sino que también disminuirá el número *inliers* a tal punto que podría arruinar los resultados completamente si el valor es demasiado bajo.

Con respecto a la triangulación de características se encontró que en este proceso aparecen una gran cantidad de puntos cuya coordenada Z hacían parece que se encontraran detrás de la cámara. Estos puntos contaminan los datos y contribuyen a un mayor ϵ, porcentaje de *outliers*, lo que incrementa el número de iteraciones necesarias para tener resultados confiables, y por lo tanto aumenta el tiempo de cálculo de la etapa de movimiento. Utilizando el hecho de que la cámara es paralela y las líneas epipolares están alineadas con el eje Y, se pueden eliminar la mayor parte de los puntos contaminados, ya que provienen de un *matching* de características incorrectas.

Un aspecto del sistema que contribuye a un mayor tiempo de cómputo innecesario tiene que ver con el módulo de triangulación. El sistema busca puntos y los triangula cada vez que

llega una nueva imagen. Luego usa los puntos triangulados de dos imágenes 3D sucesivas y hace *matching* entre estos. Sin embargo muchos puntos que fueron triangulados no calzan entre ambas imágenes 3D a causa de que el algoritmo de detección de características arrojo resultados distintos para cada imagen. Para incrementar el tiempo de cómputo, se propone como mejora realizar primero el *matching* para ambas imágenes, realizar la intersección del conjunto de puntos, y luego computar la triangulación solo para los puntos que coexisten.

En la Sección 4.3.4se compararon los resultados con distintos filtros de profundidad aplicados a los puntos triangulados. Se concluye que el filtrado de puntos lejanos contribuye a una solución más estable para distintas instancias de ejecución del algoritmo, incluso si se omite RANSAC. Sin embargo, los resultados obtenidos son solo es válidos para los parámetros utilizados en los experimentos de este trabajo, es decir, un robot andando en una trayectoria aproximadamente recta en un túnel y una cámara inclinada hacia abajo. Para generalizar este resultado sería necesario realizar más experimentos en distintos tipos de ambientes, con distintos tipo de trayectorias y configuraciones físicas de la cámara.

Los resultados obtenidos para la estimación de movimiento eran considerablemente distintos para la sección oscura y clara del túnel, incluso para un alto número de iteraciones en el algoritmo de RANSAC. La desviación estándar de los resultados en el caso RANSAC presentaba una tendencia clara a la baja con el aumento de iteraciones N siempre que el parámetro d sea lo suficientemente pequeño, lo que sugiere que los resultados se hacen más estables. Sin embargo este comportamiento no se presenta para la zona oscura del túnel, donde la desviación estándar no sufre mayores cambios con N.

Como se vio en la Sección 4.3.1, existen problemas de cálculo cuando el robot se queda completamente quieto, desviando la solución y dando resultados menos acertados. Este problema se esconde dentro del hecho de que la desviación al punto final medido es pequeño, y lo enmascara dentro de lo que sería una solución buena. Este comportamiento se hace más notorio en la sección oscura del túnel, donde la calidad y cantidad de las muestras es pobre dado los problemas de iluminación. Sin embargo este efecto se ve disminuido en las secciones 4.3.2 y 4.3.3donde se incrementa la cantidad y calidad de las características detectadas utilizando ecualización de histograma, o bien disminuyendo el parámetro de distancia d de RANSAC para encontrar una solución más acertada. Se concluye entonces que una buena solución no solo se compone de la distancia final al punto objetivo, sino que también en toda su trayectoria, la cual, en este trabajo, solo puede ser medida a través de estos espacios en donde el robot se queda quieto.

Como se puede apreciar en las secciones de resultados, en los gráficos de comparación de medias (como por ejemplo, la Ilustración 46), la calidad de la solución para el eje Z es muy superior a la calidad de la solución en los otros ejes. Esto es debido a varios factores. En primer lugar que la profundidad es calculada en la fase de triangulación, donde el

emparejamiento de características se hace solo en las líneas epipolares y por lo tanto se eliminan la mayoría de los errores de *matching* posibles. En segundo lugar las imágenes son sacadas al mismo tiempo con las mismas condiciones de iluminación y la misma distorsión dada por el movimiento, lo que da lugar a una mejora en la calidad de las características y por lo tanto en su emparejamiento. Ninguno de estos factores se cumple al comparar características de imágenes en distintos intervalos de tiempo. Con esto se concluye que la calidad de las soluciones en los otros ejes es función de la calidad y cantidad de las características detectadas en las imágenes.

Se concluye que la calidad de iluminación que posee la imagen es un factor importante para el desempeño del algoritmo, esto debido a que el algoritmo de detección busca puntos en las zonas de alto contraste, como se menciona en la Sección 2.2. El efecto de este fenómeno queda explicitado en los resultados obtenidos en la Sección 4.3.2, donde se incorpora un proceso de mejoramiento de contraste, lo que permite obtener una mejor estimación. También es posible mejorar este aspecto haciendo mejoras en el hardware del robot, por ejemplo: el uso de luces para iluminar el camino podrían aportar la luz necesaria para un funcionamiento adecuado del algoritmo, ventaja que se encuentra disponible en un auto común; una mejora en la cámara de video, con respecto a la sensibilidad a la luz que posea el sensor de imagen, podría mejorar la iluminación global de la fotografía. También, para un experimento futuro, se propone intentar tomar capturas con mayor tiempo de exposición, que si bien generaría imágenes borrosas, sería interesante conocer si la luminosidad extra aporta positivamente.

Con respecto al tiempo de ejecución del algoritmo RANSAC, la maquina utilizada rondaba los 2 segundos de tiempo en ejecutar todo el algoritmo. Este tiempo es demasiado grande para ser considerado en una aplicación de tiempo real. Sin embargo es posible acelerar el desempeño del algoritmo explotando la capacidad multinúcleo del procesador, puesto que se observó que no hubo diferencias en tiempo de ejecución al correr 1 instancia del algoritmo con respecto a correr 3 instancias. Si consideramos que hay 24 núcleos disponibles en esta máquina en particular, se podría lograr ejecutar el algoritmo a una velocidad aproximada de 12 Hz. Sin embargo, y a modo de trabajo futuro, el sistema debe ser optimizado para procesamiento en paralelo, distribuyendo en hilos de procesamiento diferentes las imágenes de la cámara, a medida que van llegando. Hay que considerar, para esta última solución, que existirá un tiempo de retardo que será al menos de 2 segundos, por lo que la información de posición no está inmediatamente disponible.

Bibliografía

[1] Chris Harris and Mike Stephens, *A combined corner and edge detector*.: In Proc. of Fourth Alvey Vision Conference, 1988.

[2] OpenCV. (2015, Mar.) OpenCV. [Online]. http://docs.opencv.org/ 3.0-beta/_images/harris_result.jpg

[3] D.Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*.: International Journal of Computer Vision, 60, 2, pp. 91-110, 2004.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, *SURF: Speeded Up Robust Features*., 2006.

[5] OpenCV. (2015, Mar.) OpenCV. [Online]. http://docs.opencv.org/3.0-beta/ _images/surf_boxfilter.jpg

[6] UTexas. (2015, Mar.) Texas University. [Online]. http://www.cs.utexas.edu/~grauman/ courses/spring2011/slides/lecture16_stereo1.pdf

[7] S Umeyama,.: IEEE Trans. Pattern Anal. Machine Inte, 1991, pp. 376-380.

[8] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer*, 2nd ed.: Cambridge University Press, ISBN: 0521540518, 2004.

[9] C. Olson, L. Matthies, M. Schoppers, and Maimone Maimone., *Robust stereo ego-motion for long distance navigation*.: IEEE, 2000.

[10] Yang Cheng, Mark W. Maimone, and Larry Matthies., *Visual odometry on the mars exploration rovers*.: IEEE Robotics and Automation magazine, 2005.

[11] K.S. Arun, T.S. Huang, and S.D. Blostein, *Least-squares fitting of two 3-d point sets*.: IEEE Trans. Pattern Anal. Mach. Intell., 1987.

[12] Bernd Kitt, Frank Moosmann, and Christoph Stiller, *Moving on to dynamic environments: Visual odometry using feature classification*.: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.

[13] Heiko Hirschmüller, Peter R. Innocent, and Jon M. Garibaldi.,.: In Proceedings of the 7th International Confe- rence on Control, Automation, Robotics and Vision, 2002, pp.

1099–1104.

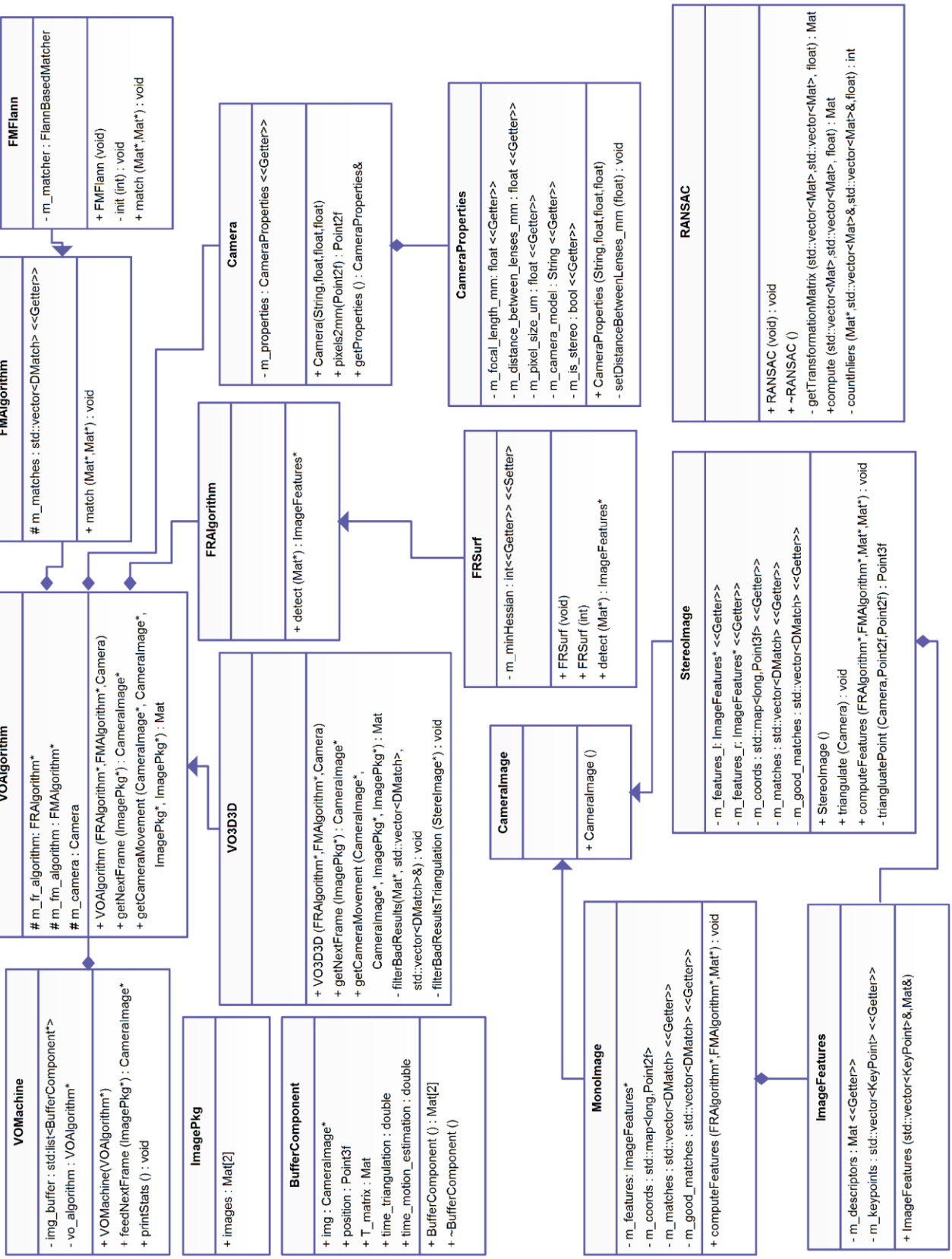
[14] M. A. Fischler and R. C. Bolles, *RANSAC sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*.: Commun. ACM, vol. 24, no. 6, pp. 381–395, 1981.

[15] Jarekt. (2015, Mar.) Wikipedia. [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/3/34/Equalized_Histogram.svg/300px-Equalized_Histogram.svg.png

[16] Phillip Capper. (2015, Mar.) Wikipedia. [Online]. https://upload.wikimedia.org/wikipedia/commons/thumb/b/bd/Equalized_Hawkes_Bay_NZ.jpg/300px-Equalized_Hawkes_Bay_NZ.jpg

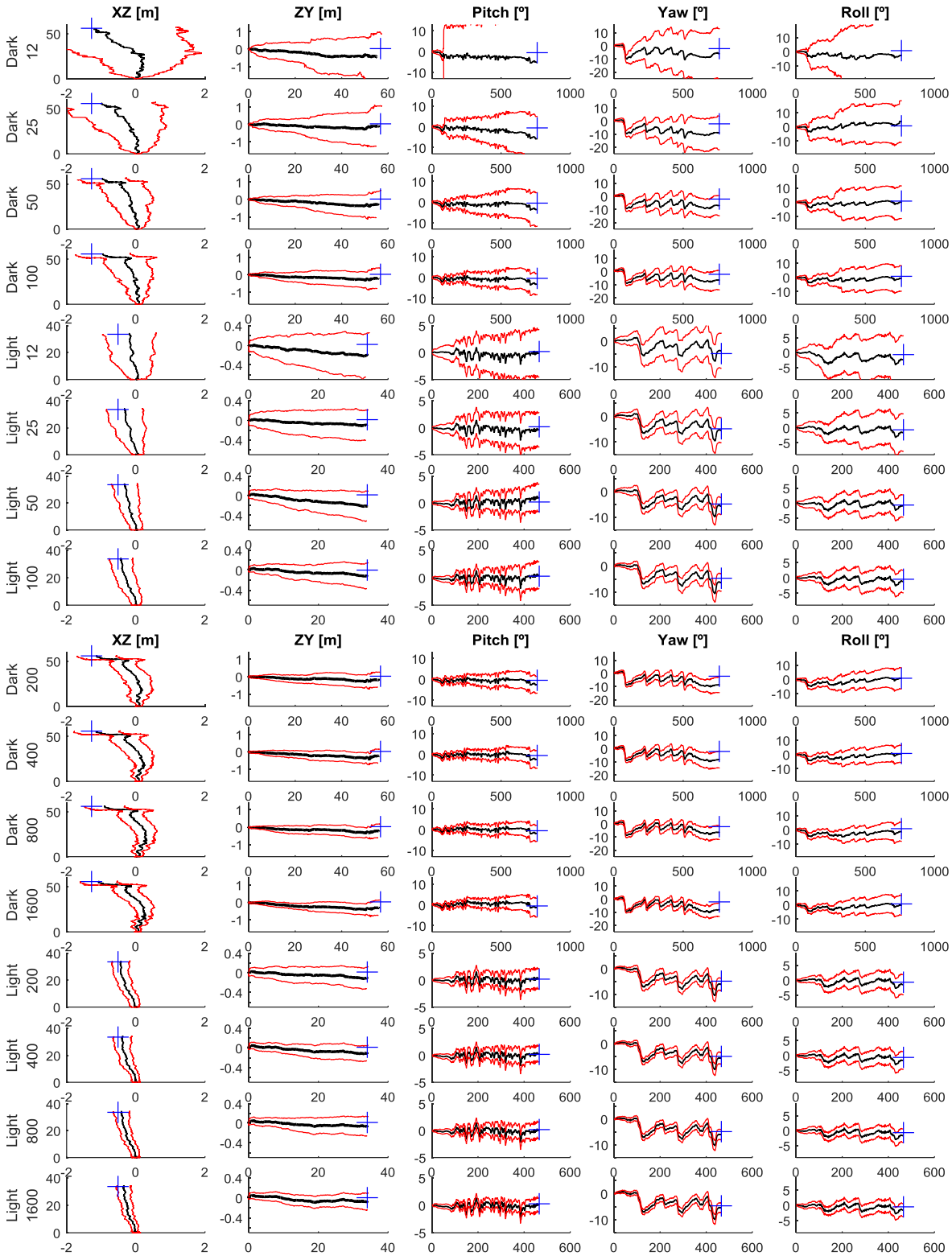
[17] Point Grey. (2015, Mar.) Point Grey. [Online]. http://www.ptgrey.com/content/images/thumbs/0002478_bumblebee-xb3-13-mp-color-firewire-1394b-60mm-sony-icx445_772.jpeg

Anexo A. Diagrama UML del Sistema

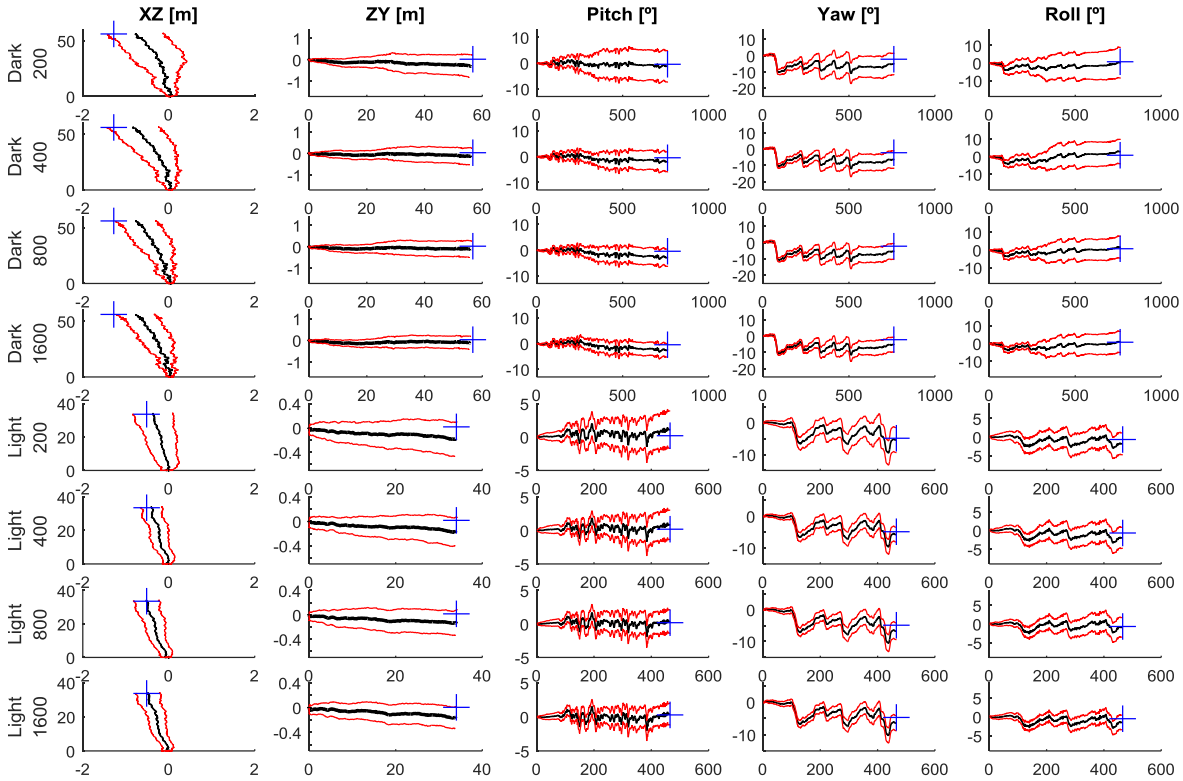


Anexo B. Resumen de Gráficos de Resultados

Los siguientes gráficos fueron obtenidos con el parámetro de RANSAC $d = 0.07$.



El siguiente gráfico utiliza el mismo parámetro $d = 0.07$, sin embargo se le realizo un pre procesamiento de ecualización de histograma.



El siguiente grafico muestra la diferencia en el parámetro d.