EVALUACIÓN DE ALGORITMOS PARA LA ESTIMACIÓN DEL DESPLAZAMIENTO EN ROBOTS SERPIENTES.

JOHANNA ALEJANDRA FAJARDO MORA

INGENIERÍA ELECTRÓNICA
FACULTAD DE INGENIERÍA
PONTIFICIA UNIVERSIDAD JAVERIANA
BOGOTÁ D.C.
COLOMBIA
JUNIO 2013

EVALUACIÓN DE ALGORITMOS PARA LA ESTIMACIÓN DEL DESPLAZAMIENTO EN ROBOTS SERPIENTES.

T.G 1231.

AUTORA:

JOHANNA ALEJANDRA FAJARDO MORA

Trabajo de grado presentado como requisito parcial para optar al título de
INGENIERO ELECTRÓNICO

DIRECTOR:

Ing. FRANCISCO CARLOS CALDERÓN BOCANEGRA

INGENIERÍA ELECTRÓNICA
FACULTAD DE INGENIERÍA
PONTIFICIA UNIVERSIDAD JAVERIANA
BOGOTÁ D.C.
COLOMBIA
JUNIO 2013

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA ELECTRÓNICA

RECTOR MAGNÍFICO: P. Joaquín Emilio Sánchez García, S.J.

DECANO ACADÉMICO: Ing. Jorge Luis Sánchez Téllez.

DECANO DEL MEDIO UNIVERSITARIO: P. Sergio Bernal Restrepo, S.J.

DIRECTOR DEL DEPARTAMENTO DE ELECTRÓNICA: Ing. Francisco Viveros Moreno.

DIRECTOR DE CARRERA DE ELECTRÓNICA: Ing. Jairo Alberto Hurtado Londoño

DIRECTOR DEL PROYECTO DE GRADO: Ing. Francisco Carlos Calderón Bocanegra

NOTA DE ADVERTENCIA
Artículo 23 de la Resolución N° 13 de Julio de 1946

"La Universidad no se hace responsable por los conceptos emitidos por sus alumnos en sus trabajos de tesis. Solo velará por qué no se publique nada contrario al dogma y a la moral católica y por qué las tesis no contengan ataques personales contra persona alguna, antes bien se vea en ellas el anhelo de buscar la verdad y la justicia"

DEDICATORIA

A Alejandro y Miguel Ángel, mis hijos.

Edward Bermúdez, mi esposo.

Edilma y Maurico, mis padres.

AGRADECIMIENTOS

Al ingeniero Francisco Carlos Calderón, por su acompañamiento y enseñanzas, durante el desarrollo de este trabajo de grado.

A mis padres, que con su gran comprensión, cuidaron de mi tesoro, mientras terminaba de cumplir una de mis grandes metas.

A mi esposo y amigo, quien me alienta seguir sin desfallecer.

A Alejo y Miguel Ángel, que con su hermosa sonrisa, sus besos y abrazos llenan de energía mi vida, y por el gran sacrificio que hicieron, para que yo culminara mi trabajo de grado.

TABLA DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

INDICE DE ECUACIONES

1. INTRODUCCIÓN

En un robot tipo serpiente, se tiene la dificultad de determinar el desplazamiento total de la plataforma a una estrategia de locomoción "gait" específica, esto debido a que en el acto de desplazarse se involucra todo la estructura mecánica y no solo parte de ella, como ocurre en robots con ruedas o miembros especializados en la locomoción (odometría). Entonces se hace necesario implementar otras técnicas que determinen el desplazamiento del robot sin utilizar sensores inerciales, que son óptimos, para medir rotación en un cuerpo rígido, pero no están diseñados para sensar la posición espacial y que tampoco haga uso de sistemas de posicionamiento global, ya que estos a pesar de dar una medida a acertada de la posición, no tienen un buen funcionamiento en ambientes internos, que es el escenario que se propone, para realizar las pruebas de este trabajo.

Ese trabajo de grado, tiene como objetivo general, evaluar el desempeño de dos algoritmos de emparejamiento temporal de puntos discretos sobre video, aplicados a la medición del desplazamiento de un robot serpiente, que se llevará a cabo mediante el desarrollo de tres objetivos específicos: Crear una base de datos de videos, en un escenario controlado, usando una cámara embarcada en distintas ubicaciones sobre el robot serpiente modular usado por el grupo de investigación SIRP; implementar dos algoritmos de emparejamiento temporal de puntos discretos, seleccionados del estado del arte, a nivel mundial, uno de acuerdo con su rapidez y baja complejidad computacional, y otro de acuerdo a su desempeño; comparar el desempeño de los dos algoritmos implementados, con el de flujo óptico piramidal de Lucas y Kanade en la tarea de estimar el desplazamiento de un robot serpiente.

El método de detección del desplazamiento del robot tipo serpiente, que se propone en este trabajo de grado, es un método por análisis de imágenes mediante la integración temporal de imágenes de una cámara ubicada en el robot tipo serpiente (LOLA), utilizado por el grupo de investigación SIRP, con el fin de evaluar el desplazamiento del robot, mediante tres algoritmos diferentes, y su desempeño, en un escenario controlado, con objetos y texturas también definidas. Para este proyecto no se cuenta con la información de distancia a los puntos de la escena medidos, la distancia recorrida se evaluará en unidades de píxeles.

En una primera fase del proyecto se investigó cuál de los algoritmos que se encuentran en el estado del arte cumplen con las condiciones de operación que se han fijado para este proyecto en cuanto a velocidad y baja complejidad computacional, y desempeño en cuanto a su capacidad de responder a cambios de rotación e iluminación en el video, de acuerdo a las necesidades planteadas en el grupo de investigación SIRP, debido a que velocidad y baja complejidad computacional, no siempre implica buen desempeño y viceversa; de estos se escogieron dos, que posteriormente fueron comparados con el algoritmo más conocido y utilizado para este tipo de aplicaciones, que es el de flujo óptico piramidal de Lucas y Kanade (Bradski, 2008).

Luego se construyó un escenario que cumpliera con las especificaciones de simplicidad y uniformidad para realizar las pruebas correspondientes, primero de ubicación de la cámara en el robot y luego con los algoritmos elegidos.

En una siguiente fase se implementaron los dos algoritmos de emparejamiento temporal de puntos discretos, seleccionados del estado del arte, uno de acuerdo a su rapidez y baja complejidad computacional, y el otro de acuerdo a su alto desempeño, los que se adaptaron a la aplicación establecida

(estimación del desplazamiento del robot tipo serpiente). Luego se realizaron varias tomas de video, que fueron utilizadas para analizar los algoritmos anteriormente elegidos, incluyendo el de referencia (Lucas y Kanade).

En una última fase se comparó el desempeño de los dos algoritmos implementados, con el de flujo óptico piramidal de Lucas y Kanade en la tarea de estimar el desplazamiento de un robot serpiente, debido a su amplio uso en el estado del arte este algoritmo (Lucas y Kanade), puede se consideró como estándar en el emparejamiento de puntos. Del resultado del flujo óptico piramidal se seleccionaron, manualmente, los puntos buenos para crear la base de datos a comparar con los resultados de los otros dos algoritmos.

El desarrollo de este proyecto aporta al sistema de posicionamiento del robot, que se está desarrollando en el grupo de investigación SIRP.

2. MARCO TEÓRICO

En aplicaciones de seguimiento en tiempo real visual, como odometría y realidad aumentada, es muy importante determinar, los puntos de interés que van a determinar el seguimiento de dichos puntos para encontrar por medio de un algoritmo confiable, la relación entre los mismos puntos en fracciones de tiempos determinados, que permitan estimar el desplazamiento del objeto a estudio en este caso el robot tipo serpiente.

Para el desarrollo de este proyecto de grado, se debe entender en una primera parte cómo se desplaza la serpiente de tal forma que se pueda ubicar la cámara en el lugar más apropiado, luego se entrará a revisar el análisis de diferentes algoritmos y las soluciones propuestas en el estado del arte.

2.1. ROBOT SERPIENTE

2.1.1. Descripción general del robot tipo serpiente

El robot a utilizar en este trabajo de grado es un robot modular, tipo serpiente, que consta de 16 servo motores unidos uno tras otro por medio de un par de soportes acoplados entre sí, en una cadena que permite el movimiento articulado del robot. Ha sido diseñado para moverse en terrenos irregulares, ya que uno de sus objetivos principales es servir como herramienta de fácil manejo debido a su bajo peso y su facilidad de carga, a los soldados del ejército colombiano en principio, para su desplazamiento en terrenos que tienen minas antipersonales, lo cual es un riesgo para los soldados y pobladores de las zonas afectadas; su facilidad de locomoción y su estructura física, lo hacen un robot versátil y apto para diferentes tareas en campo. (Melo K., 2011)

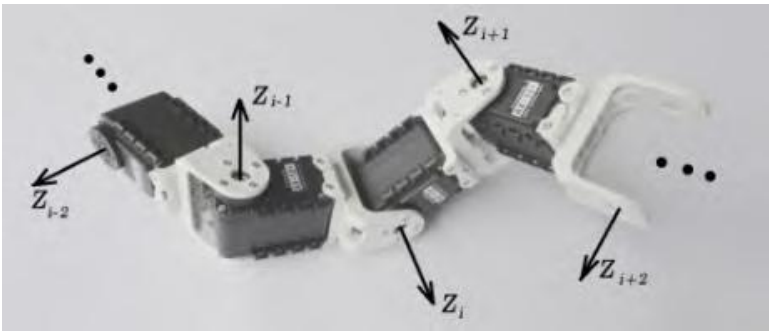


Figura 1. Cuerpo del robot tipo serpiente modular. Cuatro de los 16 módulos implementados en el robot.[1]

El robot ha sido modelado en el grupo de investigación SIRP, con el *Blockset* de *SimMechanics* que pertenece a *Simulink*, en donde se ha evidenciado que los rendimientos son muy cercanos a la realidad. Luego se ha realizado un control predictivo que permite observar el comportamiento del sistema para señales de referencia diferentes, así como las perturbaciones escalonada en la posición de salida. (Cortes F., 2011)

[1] Figura tomada del articulo: Preliminary Studies on Modular Snake Robots applied on De-mining Tasks, IEEE, 2011 (Kamilo Melo L. P., 2011)

2.1.2. Descripción general del funcionamiento y operación del robot

El robot esta construido con 16 modulos Dynamixel AX-12 + servomotores (actuadores) girados entre si 90° y soportes de plastico para los empalmes. Cada articulación puede variar su tamaño con sólo cambiar el soporte de plástico. Algunas características del robot se muestran en la Tabla 1. Para comandar el robot, existe una conexión en serie PC-robot que permite que se puedan enviar los datos de instrccion a cada actuador del robot y permite también enviar información del robot al PC de control. Estas lineas de cable se utilizan tambien para alimentar los servomotores. En algunos casos se utiliza una piel cubierta de espuma que le permite al robot, mejorar el rendimiento al andar debido al incremento de la fricción y a una mayor área de contacto. (Melo K. Paez L., 2012)

CARACTERISTICA	**MEDIDA/NÚMERO**
Número de módulos	16
Longitud por modulo	6,5 cm – 9,5 cm
Longitud total	107 cm – 157 cm
Peso total	1,4 Kg
Torque máximo por actuador	150 Kgf-cm
Velocidad máxima por actuador	50 rpm

Tabla 1. Características del robot modular tipo serpiente.[2]

Uno de los puntos más importantes del funcionamiento del robot es el control de retroalimentación de este, el grupo de investigación SIRP ha trabajado en la implementación y evaluación de un modelo de control predictivo que incluye GPC (Control Predictivo Generalizado) y UMPC (modelo sin restricciones de control predictivo); en cuyos resultados encontraron que el modelo de control distribuido de predicción fue más rápido que un regulador PID clásico, robusto a cambios de parámetros como la longitud del módulo y de la inercia, ideal para su aplicación en diferentes robots modulares. También el grupo de investigación SIRP se ha propuesto trabajar con arreglos de cámaras, sensores como acelerómetros, giroscopios y codificadores, para posibilitar la operación remota del robot (Melo K. Paez L., 2011), para lo cual este proyecto de grado aportará en cuanto al uso del video, como instrumento de medida de la distancia recorrida por el robot modular tipo serpiente.

2.1.3. Desplazamiento

Debido a que el robot serpiente es modular, se pueden aprovechar varias de sus capacidades de locomoción para el desplazamiento en terrenos variados, en el laboratorio se han investigado varias de estas capacidades del robot en la búsqueda de una apropiada forma de desplazamiento. (Kamilo Melo, 2011) El grupo de investigación SIRP, ha encontrado varias formas de desplazamiento que se mencionan a continuación.

2.1.3.1. Rodando en una cadena cerrada.

En este tipo de desplazamiento se conectan los dos extremos de la cadena, con el fin de lograr una cadena cerrada elipsoidal como se observa en la Figura 2. Esta cadena se

[2] Tabla tomada del articulo: Indoor and Outdoor Parametrized Gait execution with Modular Snake Robots, (Kamilo Melo L. P., 2012)

consideró como un sistema inestable, por lo que fue necesario contemplar diferentes aspectos teniendo en cuenta la cinemática del modelo propuesto, la velocidad y los ejes de rotación, entre otras. (Kamilo Melo, Alexandra Velasco and Carlos Parra, 2011)

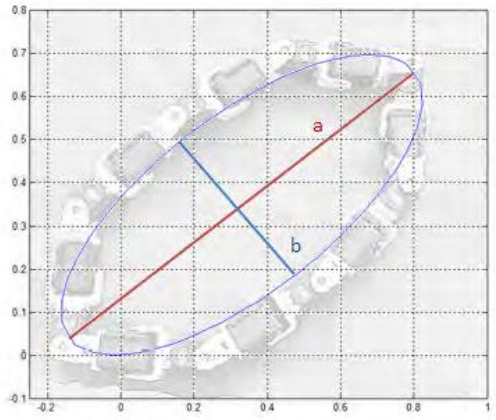


Figura 2. Robot serpiente modular en configuración elipsoidal cerrada.[3]

2.1.3.2. Rodando en la parte exterior de los cilindros.

Debido a los ambientes donde se espera que este dispositivo funciones, con alta vegetación como arbustos y árboles, esta configuración ofrece la posibilidad de embarcar una cámara para que el robot pueda capar una imagen desde arriba de los árboles. Cuando el robot gira sobre sí mismo alrededor de su eje longitudinal, ayudada por la forma curva que se genera por el desplazamiento de fase ya que sus módulos están dispuestos a 90° uno del otro, le brinda estabilidad mecánica al robot en la superficie en donde este se coloca. (Kamilo Melo, Laura Paez, Monica Hernandez, Alexandra Velasco, Francisco Calderon, Carlos Parra, 2011)

2.1.3.3. Forma de hélice para mover los cilindros exteriores

Esta configuración está relacionada con la locomoción en superficies cilíndricas en la que el desplazamiento, presenta una configuración de hélice, en donde el radio del cilindro y la longitud de la cubierta, son los parámetros principales que describen la forma. Mediante simulación, se obtuvieron resultados en cuanto a la estabilidad del sistema versus el número de vueltas alrededor del cilindro o la relación entre el radio y la longitud de la cubierta, como se muestra en la Figura 3. Se estudiaron dos formas de locomoción de este tipo, una usando un movimiento de balanceo en cilindros horizontales y otra en cilindros verticales, estas formas de locomoción son objeto de estudio del grupo de investigación SIRP.

[3] Figura 2 tomada del artículo: Preliminary Studies on Modular Snake Robots applied on De-mining Tasks (Kamilo Melo, Laura Paez, Monica Hernandez, Alexandra Velasco, Francisco Calderon, Carlos Parra, 2011)

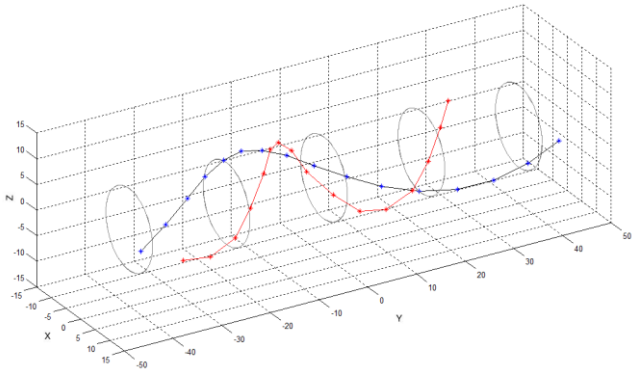


Figura 3. Representación del desplazamiento en hélice con dos configuraciones diferentes[4]

2.1.3.4. Desplazamientos LP (Linear Progression) y SW (Side Widing)

Estos movimientos de la serpiente, son descritos por la ecuación de H.Choset, donde n, es el número de modelos del robot tipo serpiente (Ecuación 1), en donde Off se refiere al desfase inicial que se puede otorgar a la serpiente mediante la interfaz de usuario, A es la amplitud del seno que describe el movimiento de la serpiente; α determina el ancho de la onda seno, que se relaciona con la frecuencia espacial y con la frecuencia temporal del desplazamiento del robot y θ brinda información sobre la velocidad del desplazamiento en grados por segundo del robot tipo serpiente. Tanto el desplazamiento o "gait", Linear Progression, como el Side WInding, pueden ser utilizados en casi cualquier ambiente, tanto externo como interno. (Flórez J., Calderón F., Parra C., 2012)

$$\theta_n(t) = \left\{ \begin{array}{c} Off_e + A_e Sin(\alpha)\ \ n : pares \\ Off_o + A_o Sin(\alpha + \delta)\ \ n : Impares \end{array} \right\}$$
$$\alpha = \frac{d\alpha}{dn}n + \frac{d\alpha}{dt}n$$

Ecuación 1. Ecuación de H. Choset

(a) (b)

Figura 4. Desplazamientos tipo LP (a) y SW (b) del robot.[5]

2.1.3.5. Desplazamiento oscilatorio

Este desplazamiento que es utilizado por los primates para ir de rama en rama en los árboles, es tema de análisis en el grupo de investigación SIRP, trata de encontrar un mínimo de energía utilizada por el primer módulo, en el impulso inicial, para realizar el movimiento de oscilación. Este movimiento le permite al robot convertir la energía potencial en energía

[4] Figura 3, tomada del artículo: Preliminary Studies on Modular Snake Robots applied on De-mining Tasks (Kamilo Melo, Laura Paez, Monica Hernandez, Alexandra Velasco, Francisco Calderon, Carlos Parra, 2011)
[5] Figura 4. Tomada del artículo: Indoor and Outdoor Parametrized Gait execution with Modular Snake Robots (Kamilo Melo, Laura Paez and Carlos Parra, 2012)

cinética, como en un péndulo, de esta forma genera desplazamiento, como se muestra en la Figura 5.

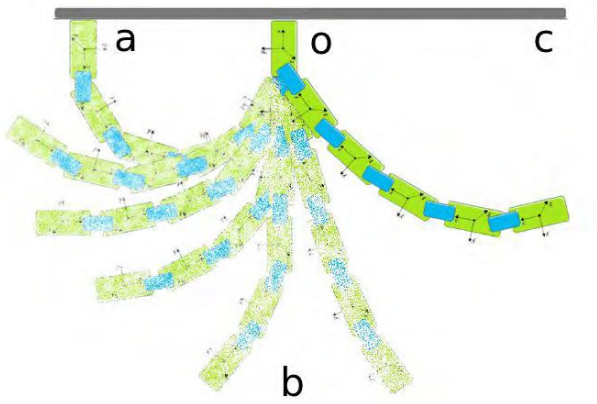


Figura 5. Configuración paso a paso del movimiento generado por el desplazamiento oscilatorio.[6]

2.2. ANALISIS COMPUTACIONAL

2.2.1. Seguimiento visual (Visual tracking)

Seguimiento visual refiere al seguimiento a largo plazo de objetos, en una secuencia de video, donde el objeto se define por su ubicación y su extensión en una trama definida. En cada cuadro que sigue, se debe poder estimar la distancia al objeto desde la cámara y/o la ausencia del objeto en el cuadro. Es importante no sesgar este proceso, ya que los objetos pueden variar e incluso la posición del robot cambiar con respecto al objeto en tanto gira en su desplazamiento y se puede ver comprometido el modelo, estas variables deben ser tenidas en cuenta en el procesamiento del video a la hora de realizar el seguimiento. Existen algunas aplicaciones computacionales que tienen en cuenta estas variables y obtienen resultados buenos a bajos costos computacionales. (Juan Pablo Ramirez Paredes, Raul E. SanchezYanez, Victor Ayala Ramirez, 2011)

Debido a que la visión basada en el seguimiento tiene un alcance limitado, en cuanto a tiempo de procesamiento y errores en el comportamiento debido a la inestabilidad numérica, es necesario plantear métodos de estimación híbridos que logren menor incertidumbre en el procesamiento del video. (Madjid Maidi, Jean-Yves Didier, Fakhreddine Ababsa, Malik Mallem, 2008)

2.2.2. Detectores

Son utilizados para distinguir los puntos en la imagen que se pueden reconocer en dos o más cuadros distintos, que posteriormente serán utilizados por el descriptor para la identificación de

[6] Figura 3, tomada del artículo: Preliminary Studies on Modular Snake Robots applied on De-mining Tasks (Kamilo Melo, Laura Paez, Monica Hernandez, Alexandra Velasco, Francisco Calderon, Carlos Parra, 2011)

características que permitan reconocer las características de la imagen. Entre los más reconocidos están:

2.2.2.1. HARRIS (Corner detector)

Teniendo en cuenta que una esquina es un punto de cambio de brillo en la imagen o un punto de máxima curvatura en ella, el método de Harris propone reconocer las características esenciales de las esquinas y/o bordes de una imagen, para reducir la cantidad de información procesada en la detección de puntos clave a partir de una imagen en escala de grises, de donde se extraen los bordes por medio del cálculo del gradiente de la curvatura, y define un umbral de operación que le permite detectar valores característicos en la imagen y así obtener los puntos y aunque no es un método muy usado, por algunas de sus complicaciones con puntos cercanos, es un método que tiene una muy buena aproximación de puntos de interés. (Zhiyong Ye, Yijian Pe,i Jihong Shi, 2006)

2.2.2.2. SURF (Speeded Up Robust Features)

SURF es un detector similar a SIFT, que es invariante a rotaciones y cambios en la imagen, pero con una complejidad menor, que se basa en la matriz Hessiana debido a su buen comportamiento computacional. Consiste en fijar un radio alrededor de un punto que sea reproducible, además, solo son necesarias seis operaciones para obtener un punto ubicado en una posición x y y, luego realiza una serie de cálculos que posibilitan encontrar los puntos de interés, de acuerdo a sus variaciones, lo que permite que sea más rápido en tiempo de respuesta con muy buenos resultados. (Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, 2006)

2.2.2.3. SIFT (Scale Invariant Feature Transform)

Es un detector que presenta, un excelente comportamiento, debido a que sus características de invariancia por rotación, escala de la imagen y cambios parciales permiten obtener bastantes puntos de interés, que son generados mediante la convolución de la imagen original con una escala variable. Por otra parte si una característica específica está por debajo de la curva del umbral este la descarta, lo que permite una mejor aproximación de puntos y lo hace bastante estable; aunque es más lento que SURF, puede ser una buena opción, debido a que es compatible con los dos sistemas operativos más utilizados Windows y Linux. (Jae-Ho Yun, Rae-Hong Park, 2006)

2.2.2.4. FAST (Features from Accelerated Segment Test)

Los detectores FAST son ampliamente usados debido a sus propiedades computacionales. Toma un parámetro dentro de un umbral entre el pixel central y un anillo al rededor del centro con un radio especifico, luego para obtener una buena cantidad de puntos clave, se establece un umbral bajo. Para obtener funciones FAST en cada nivel se filtran por Harris. (Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski, 2011)

Si existe un conjunto de n píxeles contiguos en el círculo que son todos más brillante que la intensidad de un pixel p candidato a ser una esquina o borde, más un umbral t. y además el número de pixeles n más brillantes es 12 o mayor, se considera que se ha encontrado una esquina, ósea un punto de interés (Figura 6), igualmente sucede si los n pixeles en el círculo son más oscuros que el pixel central p, más el umbral. Si no cumple con este criterio no es una esquina o punto de interés. Es un detector con mucho rendimiento aunque presenta algunos problemas como: no se tienen buenos resultados para n menores que 12, contiene supuestos de la distribución de la apariencia de la imagen, entre otros. (Edward Rosten, 2006)

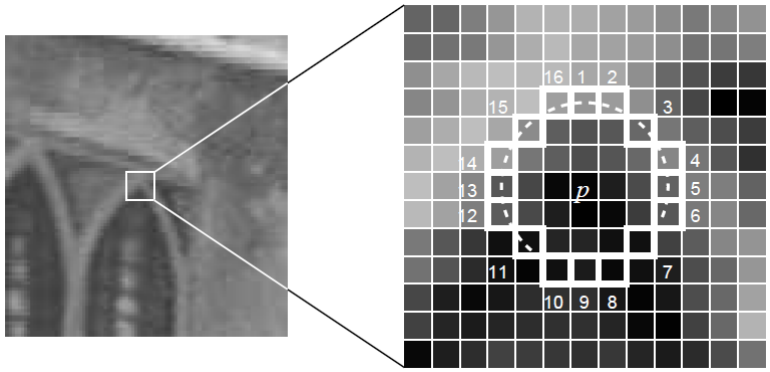


Figura 6. 12 puntos de prueba para la detección de una esquina en un cuadro de imagen.

Los cuadrados destacados son los píxeles utilizados en la detección de la esquina. El pixel en p es el centro de un candidato esquina. El arco indicado por la línea de punteada, pasa a través de los 12 píxeles juntos que son más brillantes que p, comparados con el umbral establecido.[7]

2.2.3. Descriptores

Los descriptores son utilizados para identificar características especiales del objeto que se visualiza, teniendo en cuenta esto se encuentra que la mayoría de descriptores se enfocan en las características locales de los objetos que sirven para identificar más adelante los puntos de interés en la imagen; hay varios descriptores propuestos en el estado del arte y de estos se escogerán para este trabajo de grado dos algoritmos de acuerdo a los objetivos.

A continuación se da una descripción general de algunos descriptores consultados en los trabajos más recientes o más usados, del estado del arte.

2.2.3.1. MeshHOG

A diferencia del método tradicional de identificar las características de la imagen estándar, captura tanto la geometría local, y las propiedades espaciales, para aplicaciones en 3D. (Andrei Zaharescu, Edmond Boyer, Radu Horaud, 2012)

2.2.3.2. SURF (Speeded Up Robust Features)

SURF es como su nombre lo dice un descriptor, veloz y robusto de características que en vez de seguir las imágenes completas, busca y compara la ventana que solo contiene el objeto de interés. Este enfoque de ventana, le da la capacidad de rastrear los puntos de interés de manera más eficiente, haciéndolo más robusto ante el ruido y ante cambios de posición incluso oclusiones, además permite desacoplar las características de la traslación y rotación de la imagen, se han realizado varios trabajos de investigación que demuestran que este algoritmo es bastante estable en entonos saturados, que contienen bastantes objetos con bordes suaves, definidos y con diferente iluminación. (Tuan La Anh, Jae-Bok Song, 2012)

2.2.3.3. SIFT (Scale Invariant Feature Transform)

SIFT es el descriptor más conocido y utilizado en los últimos años para realizar tareas de seguimiento, logra la invariancia a cambios en la escala y rotación al tener un marco de referencia con una escala principal, y además calcula la rotación a partir de la imagen, se

[7] Figura 6 extraida del artículo: Machine learning for high-speed corner detection (Edward Rosten, 2006)

basa en el gradiente de histogramas locales que se encuentran en una cuadricula alrededor del punto definido. La propuesta inicial ha sido modificada varias veces para lograr aproximaciones más eficientes a los cálculos. (Steffen Gauglitz, Tobias Höllerer, Matthew Turk, 2011)

2.2.3.4. SaddleSURF

Este es un descriptor que presenta una versión modificada del SURF, debido a que descriptores como SURF y SIFT se basa en puntos extremos es decir máximos y mínimos locales, esta alternativa propone utilizar puntos de silla de la imagen, que ayudan en la identificación de características, "aquí distancia euclidiana del descriptor se utiliza para encontrar las coincidencias correctas". (Sajith Kecheril S., Arathi Issac, C. Shunmuga Velayutham, 2012)

2.2.3.5. ORB

ORB es un descriptor binario basado en BRIEF, que es invariante a la rotación, e inmune al ruido, es mas rápido que el descriptor SIFT en su desempeño y ha sido probado en varios experimentos de investigación incluidos en telefonos inteligentes, demostrando bajo costo computacional. (Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski, 2011)

2.2.3.6. FREAK (Fast Retina Keypoint)

En estos días, el despliegue de algoritmos de visión en los teléfonos inteligentes y dispositivos embebidos con poca memoria y complejidad de cálculo, donde el objetivo es hacer más rápido descriptores para calcular, más compacto mientras permanece sólido a escala, rotación y el ruido.

FREAK es un descriptor basado en el sistema de visión humano, específicamente en la retina, que realiza eficientemente una comparación de intensidades en la imagen sobre una muestra específica, en un patrón de retina (Figura 7), que empieza por buscar los primeros 16 bytes que representa el grueso de la información, si la distancia es menor que un umbral establecido, se continua con los siguientes bytes para analizar la información de forma más detallada, como resultado se obtienen comparaciones en cascada que reducen el tiempo de procesamiento y lo hace robusto a cambios en la iluminación y puntos de vista. (A. Alahi, R. Ortiz, and P. Vandergheynst., 2012)

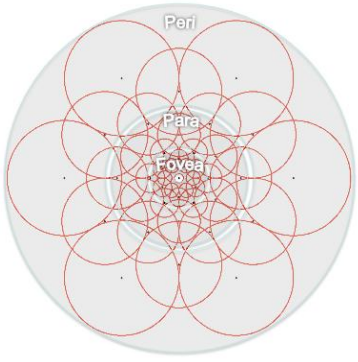


Figura 7. Patrón de muestreo FREAK.

Similar a la retina del ojo humano donde la distribución de las células ganglionares con sus correspondientes campos receptivos. Cada círculo representa un campo receptivo donde la imagen se suaviza con su correspondiente núcleo gaussiano.[8]

[8] Figura 7 extraída del articulo: FREAK: Fast Retina Keypoint (A. Alahi, R. Ortiz, and P. Vandergheynst., 2012)

Este algoritmo ha demostrado ser no solo más rápido sino más robusto que otros descriptores, como SIFT y SURF (A. Alahi, R. Ortiz, and P. Vandergheynst., 2012), como se muestra en la Tabla 2, lo cual lo vuelve una muy buena opción a desarrollar en este trabajo de grado.

Tiempo por punto de interes	SIFT	SURF	BRISK	FREAK
Descripción en [ms]	2,5	1,4	0,031	0,018
Tiempo de coincidencia en [ns]	1014	566	36	25

Tabla 2. Comparación de tiempos de respuesta entre el descriptor FREAK y otros similares.

El tiempo de computación en las imágenes es de 800x600, donde aproximadamente 1.500 puntos significativos son detectados por imagen. Los tiempos de cálculo corresponden a la descripción y tiempos de coincidencia de todos los puntos de interés.[9]

2.2.3.7. BRISK

Es un descriptor binario, invariante en rotación, que utiliza un número determinado de puntos en una muestra específica, en el cual cada punto se relaciona con muchas parejas que se dividen en grupos de corta y larga distancia. Utiliza el subgrupo de larga distancia, para estimar la dirección del punto, mientras el subgrupo de corta distancia, se utiliza para construir el descriptor binario después de la rotación de la muestra. (A. Alahi, R. Ortiz, and P. Vandergheynst., 2012)

2.2.3.8. BRIEF

Es un descriptor reciente, que utiliza pruebas simples de comparaciones binarias, entre los pixeles de un cuadro de una imagen, funciona muy similar al descriptor SIFT, sin embargo es sensible a la rotación. (Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski, 2011)

2.2.4. Puntos de interés

Debido a que no todos los puntos de interes son igualmente importantes, es clave reconocer aquellos que representan razgos que no se deformen con respecto a la posición y que sean fijos, como los bordes en especial las esquinas y regiones de alta intensidad (Henrik Aanæs, Anders Lindbjerg Dahl, Kim Steenstrup Pedersen, 2012), los diferentes descriptores mencionados en la sección 2.2.3. utilizan filtros como Harris para realizar el reconocimiento de esquinas, que resultan bastante acertados para la aplicación propuesta en este trabajo de grado. Lo importante de estos puntos, es que deben ser casi únicos para que sean reconocibles en fotogramas posteriores, lo que significa, que deben contener caracteristicas únicas de la imagen (Figura 8). (Bradski Gary, Kaehler Adrian., 2008)

[9] Tabla 2, extraída del articulo: FREAK: Fast Retina Keypoint (A. Alahi, R. Ortiz, and P. Vandergheynst., 2012)

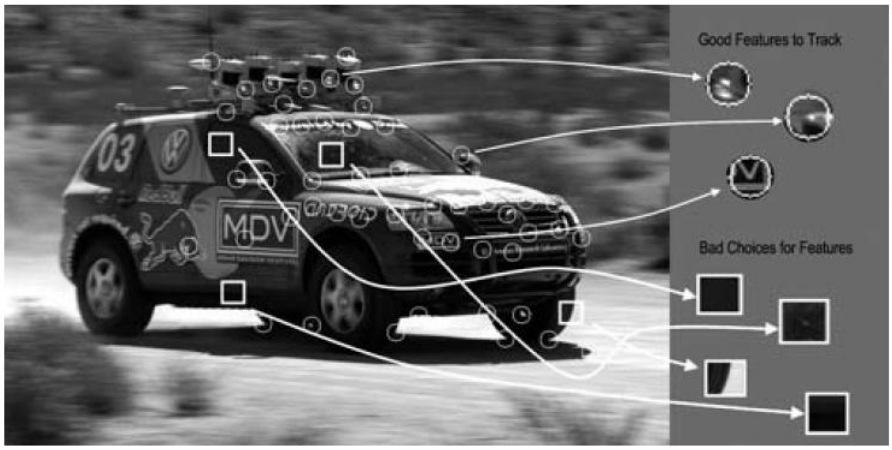


Figura 8. Ejemplo de definición de puntos en la imagen o fotograma.

Los puntos en los círculos son buenos puntos para el seguimiento, mientras que los de los cuadrados, aunque definen los bordes, son malas decisiones.[10]

Tambien para realizar la medición de distacia, es importante establecer una escala a partir de los puntos de interés, que sean lineales o no lineales, basados en los puntos de interes detectados que pueden ser analizados con modelos gaussianos de segundo orden, lo cual permite una calibración acertada en cuanto a la escala determinada, que pueda disminuir el error por deformación de la imagen, debido al dezplazamiento y/o cambio de trayectoria. (Lindeberg, 2012)

2.2.5. Emparejamiento o correspondencia de puntos

Dado que se requiere una evaluacion de puntos en tiempos rápidos, cercanos al tiempo real, es importante identificar las variables que influyen en el proceso, determinar la distancia recorrrida por el robot conforme al desplazamiento en la malla de puntos establecidos previamente, requiere, además de la definición de los puntos posibles, reconocer los cambios que se pueden presentar en el ambiente, para una vez identificados, descartar los puntos que se puedan ver más afectados por estos cambios y que tengan una incidencia importante en el resultado final. (Kalal Zdenek, 2012)

2.2.6. Análisis de texturas y tramas

Debido a que se deben comparar los puntos previamente elegidos, es importante poder controlar el ambiente en el que se desplaza el objeto, ya que la secuencia de videos, con las texturas planas del entorno, permite diferenciar cambios en la geometría, en las condiciones de iluminación y en los niveles de desenfoque de movimiento, que pueden ser útiles a la hora de construir la base de datos, ya que estas distorsiones deben ser tenidas en cuenta a la hora de evaluar los algoritmos seleccionados, de manera que resulte más confiable, y permita verificar de manera más asertiva las ventajas y falencias de cada algoritmo elegido (Gauglitz Steffen, 2011).

[10] Figura 8 tomada del libro Learning OpenCV, Cap 10. Pag 317, (Bradski Gary, Kaehler Adrian., 2008)

2.2.7. Ruido

2.2.7.1. Ruido Blanco Gaussiano:

Entendido como el ruido proporcionado por el ambiente y la cámara inherente al sistema, como las sombras, los cambios de iluminación, entre otros, que tiene una distribución similar a la gaussiana y de gran incidencia en este trabajo de grado, ya que se encuentra presente en todo momento, lo que implica de los algoritmos utilizados deben ser capaces de responder adecuadamente ante la presencia de este ruido.

2.2.8. Variables a tener en cuenta

Aunque el objeto de este trabajo de grado no es la medición del desplazamiento del robot tipo serpiente en tiempo real, uno de los algoritmos elegidos si tiene relación con la velocidad computacional de este comparada con el rendimiento y veracidad en la medida, lo que implica que al menos un algoritmo debe ser robusto en cuanto al ruido producido por el ambiente, y la rotación de la imagen. Además se deben tener en cuenta, ciertos factores que pueden afectar la medida, de esta manera se debe construir un escenario que contemple estas variables y que permita el control de la mayoría de ellas.

- ***Iluminación:*** aunque el escenario construido contará con iluminación constante, es posible que la percepción de esta varíe, en tanto la cámara se aleja o acerca a la imagen de prueba, por esto los algoritmos deben ser capaces de soportar dichos cambios que no serán muy notorios.

- ***Deformación de la imagen:*** esta se debe a la percepción de la cámara de la imagen, ya que al cambiar de posición la serpiente y por tanto la cámara embarcada en ella, la imagen se ve diferente, es por esto que la selección de puntos debe ser cuidadosa en tanto a aquellos que cambian su forma de acuerdo a la rotación de la imagen, esto se logra por medio de algoritmos robustos que son capaces de distinguir estos cambios y asociarlos a los puntos de interés seleccionados como FREAK, FAST y por supuesto Lukas y Kanade

- ***Ubicación de la cámara:*** La ubicación de la cámara en el robot serpiente, es clave para el desarrollo exitoso de este trabajo, ya que influye sustancialmente en la deformación de la imagen con respecto al movimiento del robot, por ende se deben realizar varias pruebas en donde se encuentre la mejor ubicación de la cámara en el robot, para cumplir el objetivo.

2.3. LUKAS Y KANADE

2.3.1. Flujo Óptico

El flujo óptico es el proceso por el cual se puede evaluar el movimiento de dos fotogramas, sin ningún conocimiento previo sobre el contenido de las imágenes; es así, como se puede asociar algún tipo de velocidad con cada pixel en la imagen y/o de manera equivalente cierto desplazamiento que representa la distancia en que un pixel se ha movido entre el los dos fotogramas. Este desarrollo que se refiere comúnmente a una óptica densa, asocia una velocidad con cada pixel determinado en la imagen (Figura 9).

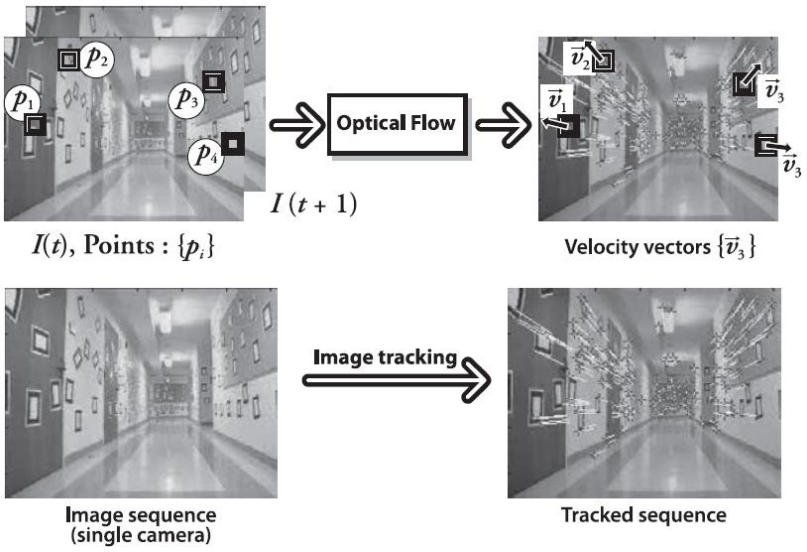


Figura 9. Flujo óptico.

Pixeles elegidos (parte superior izquierda): se realiza un seguimiento en el tiempo y su movimiento se convierte en vectores de velocidad (superior derecha); una sola imagen del pasillo (parte inferior izquierda), y los vectores de flujo (inferior derecha) como se mueve la cámara en el pasillo.[11]

Debido a que no todos los puntos de la imagen son buenos para seguir cuadro a cuadro, ya que aunque sean bordes su contenido puede ser poco diferente del resto, lo que dificulta el reconocimiento en los fotogramas, algunos algoritmos escogen subconjuntos de puntos que tienen ciertas características deseables como esquinas, lo que hace de estos algoritmos posibilidades más robustas y confiables, con bajo costo computacional.

2.3.2. ¿Cómo funciona el método de Lukas y Kanade?

El algoritmo de Lucas-Kanade (LK) es una importante técnica para producir resultados densos que funciona con información local de una pequeña ventana que rodea los puntos de interés, en un principio esto se convirtió en un problema debido a que era difícil el seguimiento de estos puntos, cuando había grandes movimientos. Fue por eso que se desarrolló el algoritmo de flujo óptico piramidal de Lukas y Kanade que permite "seguir" las ventanas en forma de pirámide, desde el nivel más alto, y bajando los niveles uno a uno, de esta forma, los grandes movimientos son captados por las ventanas locales.

El algoritmo de Lukas y Kanade se basa en tres supuestos:

1. ***Constancia de brillo:*** se supone que el brillo de un pixel no cambia debido a que se realiza un seguimiento imagen a imagen. Como se muestra en las siguientes ecuaciones:

$$f(x, t) \equiv I(x(t), t) = I(x(t + dt), t + dt)$$

Ecuación 2. LK Constancia de brillo[12]

[11] Figura 9 imagen tomada del libro Learning OpenCV, Cap 10. Pag 322, (Bradski Gary, Kaehler Adrian., 2008)
[12] Ecuación 2 tomadas del libro Learning OpenCV, Cap 10. Pag 325, (Bradski Gary, Kaehler Adrian., 2008)

$$\frac{\partial f(x)}{\partial t} = 0$$

Ecuación 3. LK Constancia de brillo[13]

2. ***Persistencia temporal o "pequeños movimientos":*** la imagen se mueve lentamente en el tiempo, lo que significa que los incrementos temporales son rápidos en relación, a la variación en el movimiento de la imagen, es decir que el punto de interés no se sale del cuadro repentinamente.

   Esto puede ser analizado en un primer instante en una dimensión, teniendo en cuenta la Ecuación 3, del primer supuesto y sustituyendo f(x(t),t) por I(x(t),t) y aplicando derivadas parciales, se obtiene:

$$\underbrace{\left.\frac{\partial I}{\partial x}\right|_t}_{I_x} \underbrace{\left(\frac{\partial x}{\partial t}\right)}_{\mathbf{v}} + \underbrace{\left.\frac{\partial I}{\partial t}\right|_{x(t)}}_{I_t} = 0$$

Ecuación 4. Sustitución en la Ecuación 2 de brillo, por I (imagen); derivada parcial.[14]

Donde Ix se refiere a la imagen actual, It a la imagen a través del tiempo y v a la velocidad de flujo óptico en una dirección, de esta forma se obtiene que:

$$\mathbf{v} = -\frac{I_t}{I_x}$$

Ecuación 5. Velocidad de flujo óptico en 1D[15]

Aunque aparentemente funciona en 1D, en dos dimensiones ya no se obtiene el resultado esperado, debido a que la ecuación se transformaría en una ecuación con dos variables desconocidas, lo que permitiría solo encontrar una solución para un movimiento normal a la recta que describe la Ecuación 6 en 2D para un pixel, lo cual resulta un problema.

$$I_x u + I_y v + I_t = 0$$

Ecuación 6. Generalización para 2D de la Ecuación 5.
Donde u es la velocidad en el eje x y v es la velocidad en el eje y.[16]

Es por eso que se recurre a la tercera suposición del algoritmo de Lukas y Kanade.

3. ***Coherencia espacial:*** se refiere a que los puntos vecinos en un fotograma que pertenecen a la misma superficie, tienen un movimiento similar en el plano de la imagen. De esta manera se puede asumir que el movimiento de los pixeles que rodean a un pixel central, puede ser descrito por el siguiente sistema de ecuaciones:

[13, 14, 15, 16] Ecuaciones 3, 4, 5 y 6 tomadas del libro Learning OpenCV, Cap 10. Pag 327, 328, (Bradski Gary, Kaehler Adrian., 2008)

$$\underbrace{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix}}_{\substack{A \\ 25\times 2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\substack{d \\ 2\times 1}} = -\underbrace{\begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}}_{\substack{b \\ 2\times 1}}$$

Ecuación 7. Sistema de ecuaciones para evaluar una ventana de 5x5 valores brillantes.
Este sistema de ecuaciones se puede hacer con los valores que se deseen, que rodeen el pixel central.[17]

$$\underbrace{(A^{\mathrm{T}}A)}_{2\times 2}\underbrace{d}_{2\times 1} = \underbrace{A^{\mathrm{T}}b}_{2\times 2}$$

Ecuación 8. Ecuación 7 minimizada para encontrar una solución.[18]

$$\begin{bmatrix} u \\ v \end{bmatrix} = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}b$$

Ecuación 9. Solución del sistema de ecuaciones propuesto por LK[19]

De esta manera se obtiene una aproximación bastante acertada al movimiento entre ventanas, y para que estas suposiciones tengan validez, el análisis en forma de pirámide, brinda mayor seguridad y reduce la incertidumbre que se genera en la medida, al tomar como inicio la ventana superior de la pirámide y descender hacia la base que es más ancha a cierta velocidad establecida previamente (Figura 10).

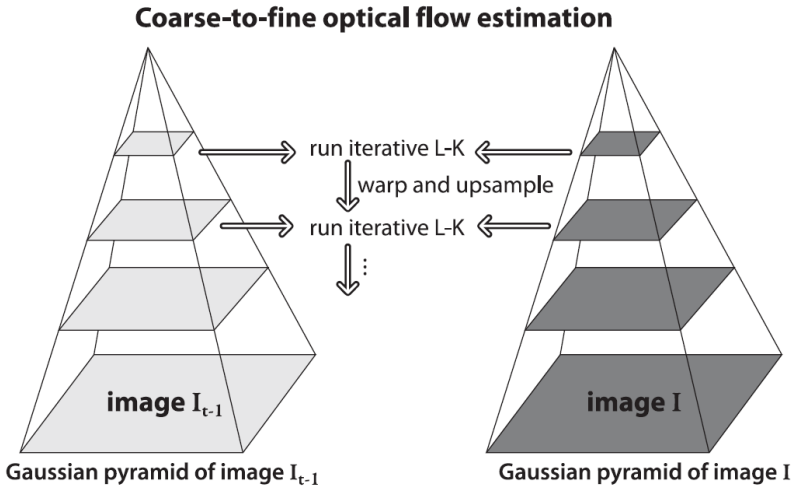


Figura 10. Flujo óptico piramidal de Lukas y Kanade.
Procesar primero el flujo óptico en la parte superior de la pirámide, reduce los problemas causados por la violación de los supuestos del movimiento pequeño y coherente, el movimiento estimación del nivel anterior se toma como el punto de partida para la estimación de movimiento en la capa de abajo.[20]

[17, 18, 19] Ecuaciones 7, 8 y 9 tomadas del libro Learning OpenCV, Cap 10. Pag 328, (Bradski Gary, Kaehler Adrian., 2008)
[20] Figura 10 tomada del libro Learning OpenCV, Cap 10. Pag 330, (Bradski Gary, Kaehler Adrian., 2008)

3. ESPECIFICACIONES

Para la realización de este proyecto de grado se requiere en primer lugar, de un escenario controlado con paredes monocromáticas y superficies lisas, para el desplazamiento del robot tipo serpiente, por otro lado en una de las paredes, que denominaremos el plano de referencia, se colocaron varias imágenes y/o objetos fijos, para la realización de las pruebas de video con el robot y así realizar las bases correspondientes para la posterior comparación.

Durante la construcción del escenario se eligieron, del estado del arte, los dos algoritmos a utilizar, que cumplían con los requisitos planteados en los objetivos, que junto al algoritmo de flujo óptico piramidal de Lukas y Kanade, sirvieron para realizar las pruebas y posterior comparación.

El sistema tiene como entrada, el video seleccionado para luego procesarlo de la siguiente forma:

- Se añadió al algoritmo principal, una primera parte, que convierte los videos en fotogramas o cuadros que no son almacenados en la memoria del computador, los que permite que aunque los algoritmos sean robustos, el uso de los recursos del computador, no sea mayor al necesario.

- Las imágenes son procesadas en primer lugar por el detector de puntos *GoodFeaturesToTrackDetector (GFTD),* que no solo ubica los puntos mediante Harris en la imagen, sino que seleccionas aquellos "más importantes", que fueron los que permitieron realizar un seguimiento posterior.

- Cada algoritmo elegido tiene un descriptor que aplicado al video, sirvió para determinar la ubicación de los puntos importantes con los que se trabajó en los fotogramas y son ellos los que determinan en gran parte las características en el procesamiento que se buscan en este proyecto.

- Seguido, se realizó una correspondencia entre los puntos según los momentos seleccionados entre los pasados y actuales, por medio del algoritmo ya implementado en opencv, llamado *BruteForceMatcher,* que compara cada punto de una imagen con todos los puntos de la otra, obteniendo resultados favorables.

- Teniendo en cuenta la correspondencia se estimó el desplazamiento de acuerdo a la variación de los pixeles en la imagen, utilizando Matlab, en el que se estimó, la media, la mediana y la varianza, para obtener los resultados de la distancia recorrida por la serpiente para cada caso, en pixeles.

- Como se muestra en la Figura 11. Mediante la Ecuación 10, que relaciona la distancia al objeto con los pixeles y al ángulo de visión de la imagen con respecto a la cámara. Para finalmente realizar una comparación manual de los resultados obtenidos en el procesamiento de los videos, con los datos reales de desplazamiento de la serpiente, en medidas de longitud, para futuras comparaciones en implementaciones de otro tipo de algoritmos con los videos suministrados como base de datos.

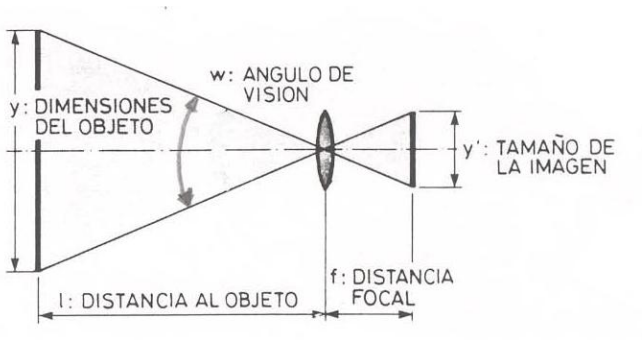


Figura 11. Relación entre la distancia focal y el tamaño de la imagen

$$\tan w = y'/f$$

Ecuación 10. Relación entre la distancia y pixeles.
w: ángulo de visión, f: distancia focal, y´: tamaño de la imagen

- Por último se evaluó el desempeño de cada algoritmo con respecto a una medición manual, con el fin de verificar que la medición anterior se aproxime a la real y/o evaluar la eficiencia de cada algoritmo según el desplazamiento del robot tipo serpiente.

Dado que la salida esperada del sistema consiste en la medición del desplazamiento de la imagen en píxeles, se tomaron varios parámetros para el análisis de resultados enunciados a continuación:

- ***Tiempo de procesamiento***: Entendido como el tiempo que demora cada algoritmo en generar un resultado verídico, ya que para este trabajo de grado, no se tiene como prioridad los resultados en tiempo real, los algoritmos escogidos no solo deberán ser rápidos en procesar la información, sino que deberán acercarse de manera valida al resultado esperado.

- ***Cercanía de los resultados a los obtenidos en el procesamiento del video con el algoritmo de Lukas y Kanade:*** dado que el algoritmo de Lukas y Kanade, es un algoritmo confiable en la estimación de desplazamiento, será este la referencia inicial de buen funcionamiento de los demás algoritmos elegidos, lo que permitirá tener alguna certeza de la efectividad de los algoritmos escogidos en la medición de desplazamiento del robot tipo serpiente.

- ***Cercanía de los resultados obtenidos con la medición manual:*** debido a que las pruebas de este trabajo de grado se realizarán, en un escenario controlado, es posible medir de manera física la distancia que recorre el robot, por lo tanto este será un parámetro a tener en cuenta en la evaluación de desempeño de los algoritmos, en los que se podrá observar su correspondencia con la realidad y lo esperado.

3.1. DIAGRAMAS EN BLOQUES

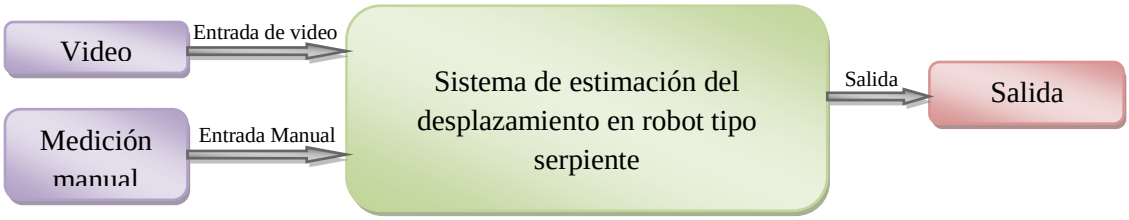


Figura 12. Diagrama de bloques general del sistema

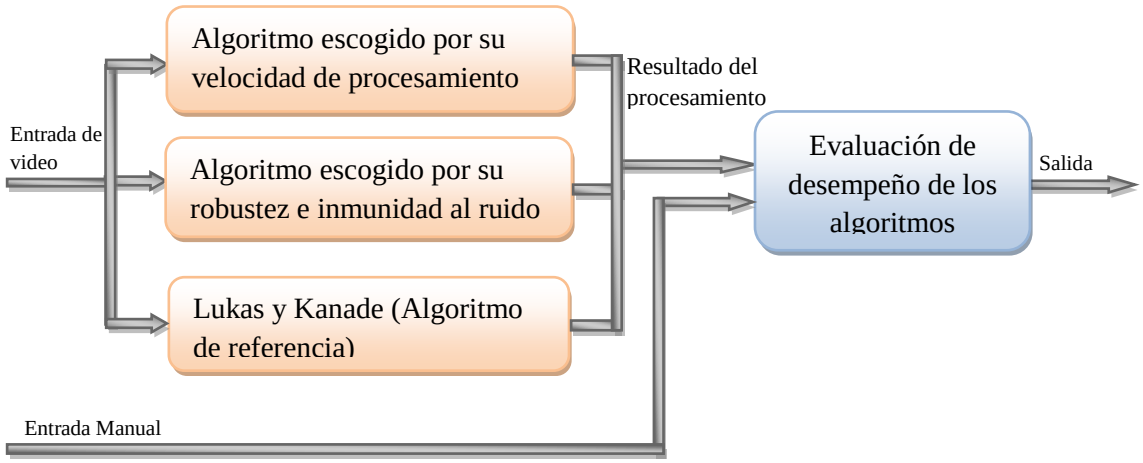


Figura 13. Diagrama de bloques interno del sistema

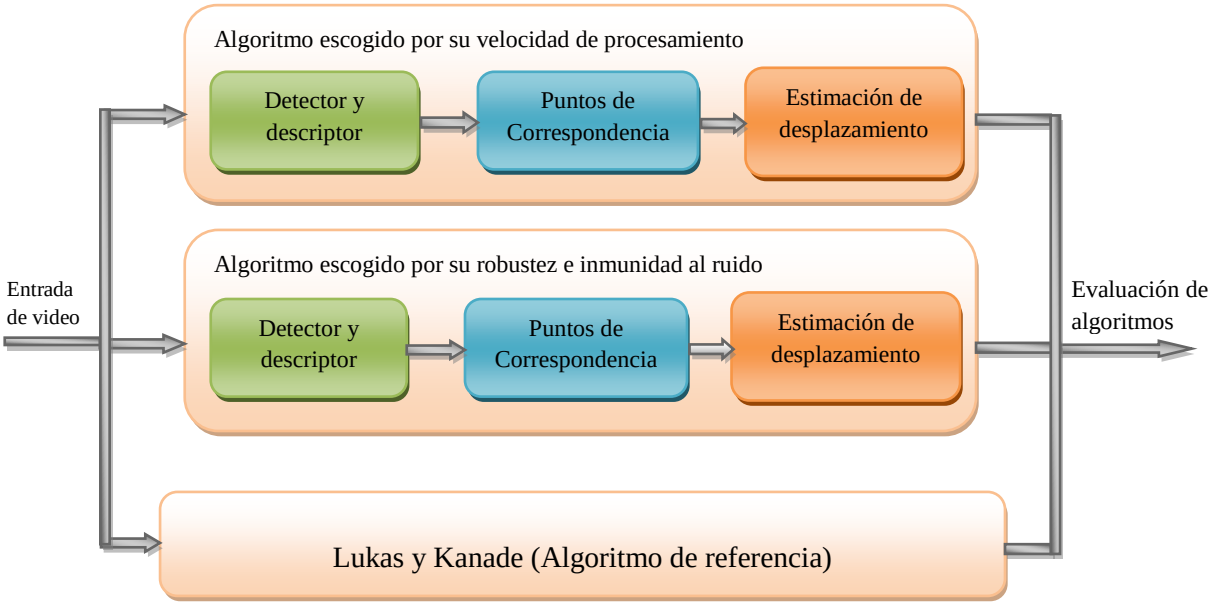


Figura 14. Diagrama de bloques del procesamiento del video

3.2. DIAGRAMAS DE FLUJO

3.2.1. Lukas y Kanade

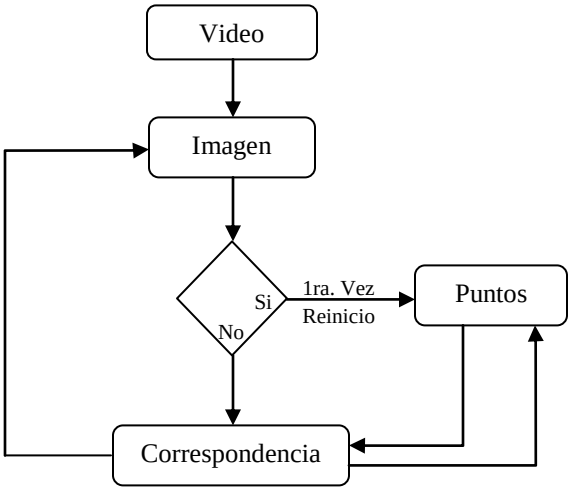


Figura 15. Diagrama de flujo, algoritmo de Lukas y Kanade

3.2.2. Algoritmos escogidos

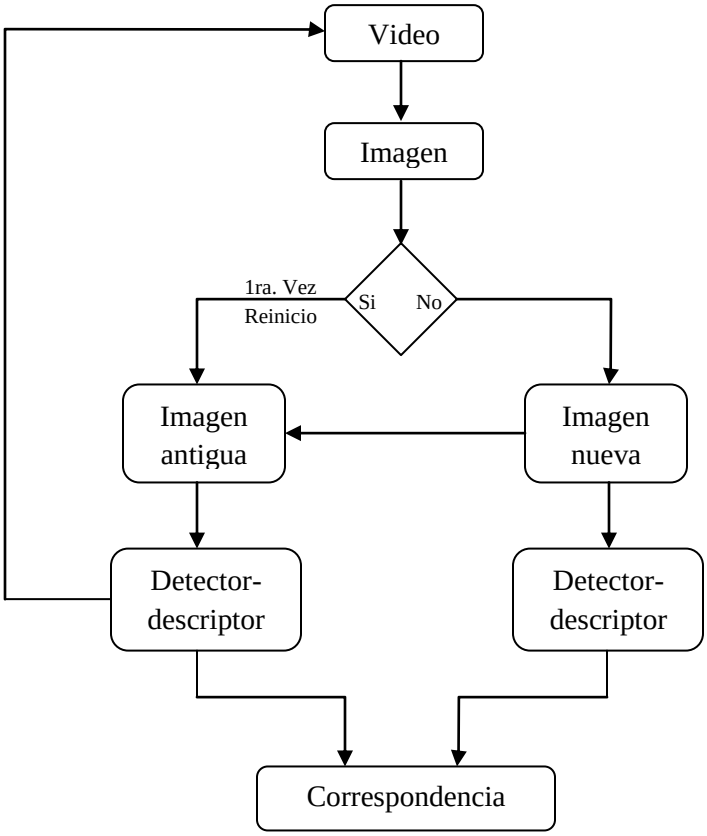


Figura 16. Diagrama de flujo de los algoritmos elegidos

4. DESARROLLOS

4.1. ESCENARIO CONSTRUIDO

Para la realización de este trabajo de grado, se necesitó adecuar un espacio que fuera en su mayoría monocromático, y que los objetos que estuvieran en él tuvieran tanto bordes definidos como suaves; por eso, se adecuo un espacio en el laboratorio de robótica, en el que se realizaron las pruebas correspondientes. (Figura 17)

Figura 17. Escenario de pruebas.

En el escenario de pruebas, se pueden observar objetos con figuras definidas en 2D y objetos en 3D, que permiten tener una mejor apreciación de que tan robusto, puede ser cada algoritmo, en tanto la iluminación y por ende las sombras cambian, pero los objetos se mantienes estáticos.

4.2. ESCOGENCIA DE UBICACIÓN DE LA CÁMARA EN EL ROBOT Y CREACIÓN DE LA BASE DE DATOS DE LOS VIDEOS

Paralelo a la escogencia de los algoritmos a utilizar en este trabajo de grado, se realizaron las pruebas para escoger la ubicación más conveniente de la cámara en el robot, así como la forma de desplazamiento de este, debido a que dicha ubicación, permitirá obtener resultados más confiables, de acuerdo a los algoritmos elegidos.

Los desplazamientos del robot elegido para este trabajo de grado fueron*: linear progression y Side Winding*, debido a su baja complejidad y mayor posibilidad de conservar un movimiento en línea recta a cierta distancia del objetivo. Teniendo en cuenta lo anterior, se escogieron dos ubicaciones posibles para la cámara y se tomaron tres videos por movimiento y ubicación de la cámara, a tres velocidades diferentes en el escenario controlado.

Para la realización de estos videos se utilizaron: el software de captura de video *GUVCViewer*, en el que se calibraron los parámetros de captura de la cámara utilizada de forma manual, en donde la exposición absoluta, dependió de la iluminación en el momento de las pruebas y quedó fija, y el foco absoluto, quedó fijo en infinito, para obtener la mejor

imagen posible, a la resolución 640X480, como se muestra en la Figura 19; el robot serpiente, cuenta con una interfaz disponible (Figura 18), que permite fijar los valores de amplitud del seno y el ángulo, que forma la serpiente en el movimiento y la velocidad estimada en dTheta/dt, entre otros parámetros (valores de la Ecuación 1,), en algunos movimientos prediseñados del robot, que fueron ajustados, para los dos movimientos como se muestra en la Tabla 3 y la Tabla 4, el único parámetro que se varió, durante las pruebas, fue la velocidad en cada tipo de desplazamiento y con cada ubicación de la cámara en la serpiente entre, los valores 18, 36 y 54 rad/s.

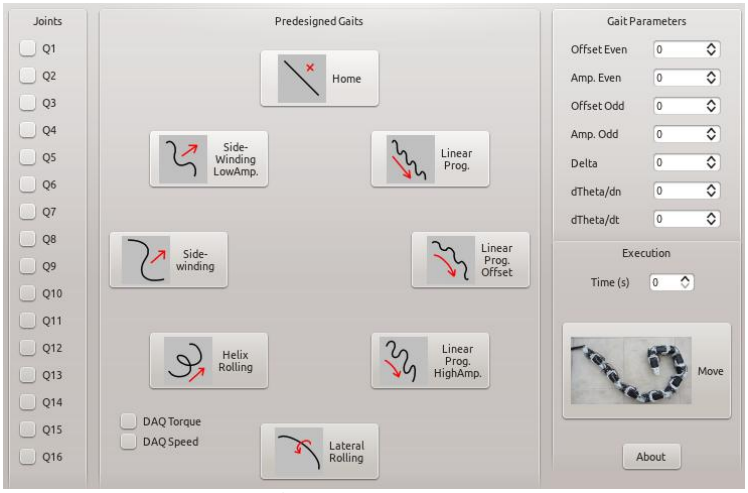


Figura 18. Interfaz de usuario, del robot tipo serpiente

Figura 19. Parámetros utilizados, en el control manual de la toma de video, en las pruebas realizadas.

PARÁMETROS	VALOR
Offset even	0
Amp. Even	10
Offset odd	0
Amp. Odd	0
Delta	0
dtheta/dn	120
dtheta/dt	-18 ó -34 ó -54
Tiempo de ejecución [s]	10

Tabla 3. Parámetros de movimiento del robot tipo serpiente en el movimiento predeterminado: Linear Progression

PARÁMETROS	VALOR
Offset even	0
Amp. Even	10
Offset odd	0
Amp. Odd	6
Delta	45
dtheta/dn	30
dtheta/dt	-18 ó -34 ó -54
Tiempo de ejecución [s]	10

Tabla 4. Parámetros de movimiento del robot tipo serpiente en el movimiento predeterminado: *Side Winding*

4.2.1. Movimientos *Linear Progression y Side Winding*, con la cámara ubicada al frente

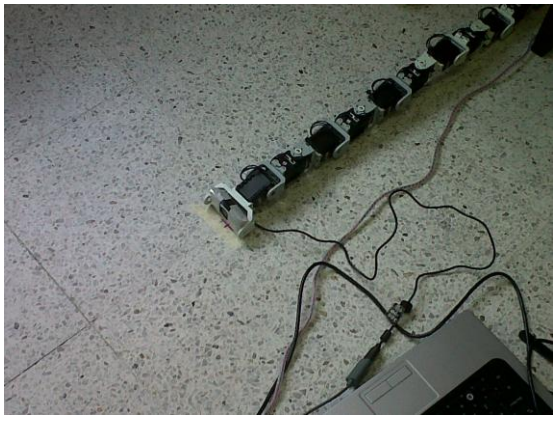


Figura 20. Movimiento Linear Progression (LP), utilizado para realizar las bases de videos.

Debido a la ubicación de la cámara, para el movimiento *Linear Progression* (LP), se observan cambios bruscos, sobretodo en la mayor velocidad (54 dThta/dt), lo que supone, mayor dificultad a la hora de estimar distancia. Por otro lado el movimiento del

robot debe ser frontal al escenario, para lograr una mejor captura y posterior análisis de los videos correspondientes, en este caso su desplazamiento es en línea recta, en dirección al escenario.

Al contrario del movimiento anterior, para *Side Winding*, con esta ubicación de la cámara, el robot serpiente, debe desplazarse de lado, para poder capturar en video el escenario, es decir en vez de acercarse o alejarse al escenario, su recorrido es paralelo a él; aunque en la práctica, el robot se desvía, produciendo un movimiento en semicírculo.

4.2.2. Movimiento *Linear Progression*, Cámara ubicada lateral

Figura 21. Ubicación lateral de la cámara en el robot, durante el movimiento de *Linear Progression*.

Se realizaron varias tomas con la cámara colocada en el lado de la serpiente que apunta al escenario, para determinar en qué posición se podía obtener un mejor resultado; para esto, el robot se desplaza paralelo al escenario, logrando capturar en los videos, gran parte los objetos colocados allí.

4.2.3. Movimiento *Side Winding*, Cámara ubicada lateral

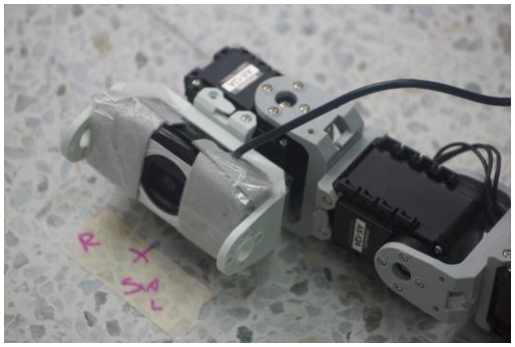


Figura 22. Cámara embarcada en el robot, de forma lateral, para el movimiento *Side Winding*

Cuando el robot se desplaza de esta manera, y la cámara se ubica en los lados del cuerpo del robot, el movimiento de este, debe ser frontal al escenario, en donde se puede apreciar, que se aleja o se acerca al escenario aunque su cuerpo está paralelo a él.

4.2.4. Videos adicionales en un escenario no controlado

En estos 20 videos se puede observar, como se comporta la serpiente, en un ambiente no controlado, variando su dTheta/dt, de 5 en 5 a partir de 10 y finalizando en 55.[21] Los

[21] Videos realizados por Johanna María Flores, integrante del grupo SIRP de investigación de la Pontificia Universidad Javeriana

parámetros para estas pruebas se muestran en la Tabla 5 y Tabla 6. Parámetros fijados para la toma de los videos en un ambiente no controlado, para un desplazamiento Side Winding.

Tabla 5. Parámetros fijados para la toma de los videos en un ambiente no controlado, para un desplazamiento *linear Progression*.

PARÁMETROS	VALOR
Offset even	0
Amp. Even	40
Offset odd	0
Amp. Odd	10
Delta	45
dtheta/dn	-45
dtheta/dt	Varía
Tiempo de ejecución [s]	36

Tabla 6. Parámetros fijados para la toma de los videos en un ambiente no controlado, para un desplazamiento *Side Winding*.

A estos videos también se les calculo el desplazamiento con los algoritmos elegidos, en los que se evidencio mayor lentitud en el procesamiento de los videos obtenidos.

4.3. ESCOGENCIA DE ALGORITMOS

Para escoger los algoritmos descriptores, se realizó una búsqueda en el estado del arte y posteriores pruebas con dos fotogramas iguales, con el fin de comparar el funcionamiento de los diferentes detectores y descriptores; luego se escogieron dos algoritmos, que además de cumplir con los requisitos de ser robustos ante el ruido y de rápida ejecución, estuvieran referenciados en las librerías de Opencv y que pudieran ser utilizados sin ningún inconveniente de derechos de autor.

4.3.1. FREAK (Fast Retina Keypoint)

Se escogió en primera instancia el algoritmo descriptor FREAK, por su capacidad para operar con ruido y además porque según el estado del arte, supera en velocidad y resultados a otros descriptores como BRISK, ORB e incluso SIRF. Además FREAK se encuentra en las librerías de Opencv, en donde se muestra claramente, el proceso de carga de imágenes, pasa estas imágenes a escala de grises y realiza el procesamiento a través del detector, descriptor y finalmente la correspondencia de puntos.

El detector utilizado en FREAK, puede variar conforme el usuario lo programe, por tal razón se realizaron, pruebas con tres detectores, conocidos y se obtuvieron los siguientes resultados:

Figura 23. Resultado de correspondencia del algoritmo descriptor FREAK, utilizando el detector FAST

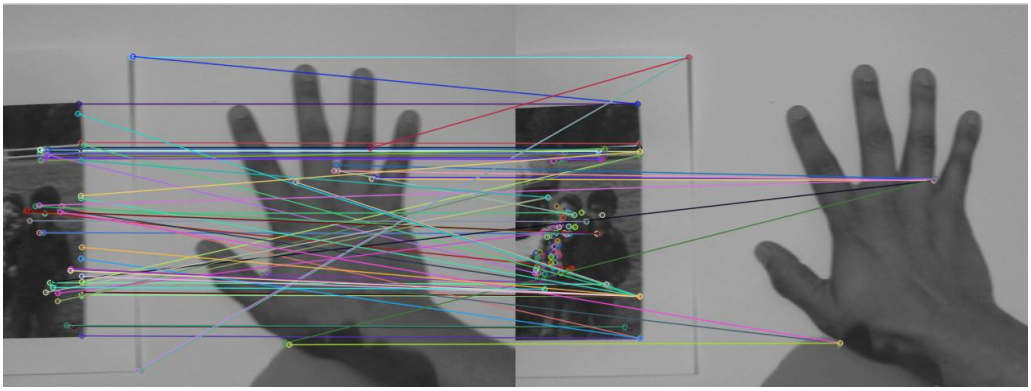


Figura 24. Resultado de correspondencia del algoritmo descriptor FREAK, utilizando el detector GFTD

Figura 25. Resultado de correspondencia del algoritmo descriptor FREAK, utilizando el detector SURF

	FAST	GFTD	SURF
Tiempo de detector [s]	0.00785627	0.348106	0.861627
Tiempo de descriptor[s]	2.5549e-5	0.15618	0.258734
Tiempo de correspondencia [s]	0.00142363	0.00154594	0.00495643

Tabla 7. Tiempos de respuesta del algoritmo FREAK, utilizando los detectores FAST, GFTD y SURF

Finalmente y debido a las pruebas anteriores, en la que se evidencia el desempeño, no solo en correspondencia de puntos sino en velocidad, se decidió utilizar para este trabajo de grado el algoritmo descriptor FREAK, con el detector SURF, ya que en comparación con el GFTD (*Good features to track detector*), muestra buen comportamiento y su respuesta en cuanto a velocidad es aceptable para este trabajo de grado.Tabla 7

El algoritmo que se utilizó puede verse en el anexo 1.

4.3.2. DETECTOR_DESCRIPTOR_MATCHER

El siguiente algoritmo elegido, se encontró en los recursos de Opencv, en donde a través del algoritmo *Detector_Descriptor_Matcher.cpp*, se permite al usuario configurar, cada opción para utilizar los detectores, descriptores y el tipo de correspondencia que se desea, por tal razón, se consideró bastante útil para, encontrar el algoritmo robusto que se requería en este trabajo de grado, teniendo en cuenta que, este algoritmo realiza el emparejamiento de punto de forma aleatoria, solo se utilizó para evaluar y escoger el descriptor que se utilizó posteriormente.

Teniendo en cuenta la opción de este algoritmo para controlar el detector y el descriptor a utilizar para el procesamiento de las imágenes, se realizaron pruebas con varias combinaciones detector-descriptor, para elegir la que mejor respuesta obtuviera, de acuerdo a los requisitos anteriormente expuestos; los resultados fueron los siguientes, teniendo en cuenta que el par de imágenes de las siguientes figuras, se seleccionó ya que presenta regiones con bordes definidos, y regiones más suaves.

- **Detector FAST, Varia Descriptor**

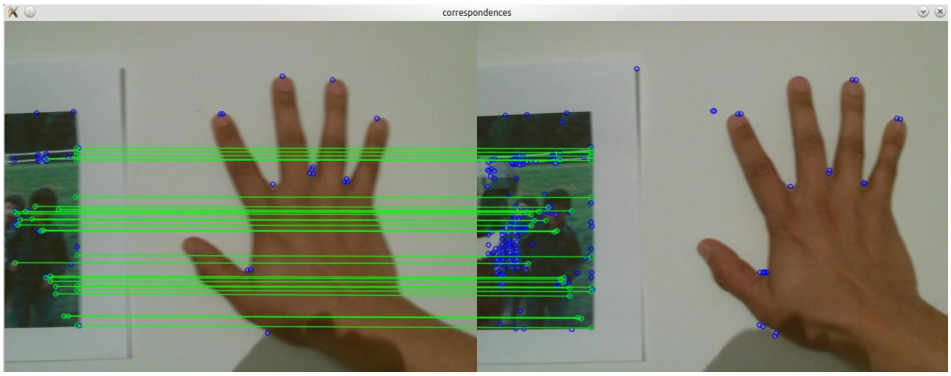


Figura 26. Prueba del algoritmo *Detector_descriptor_matcher*, utilizando detector FAST y descriptor BRISK.

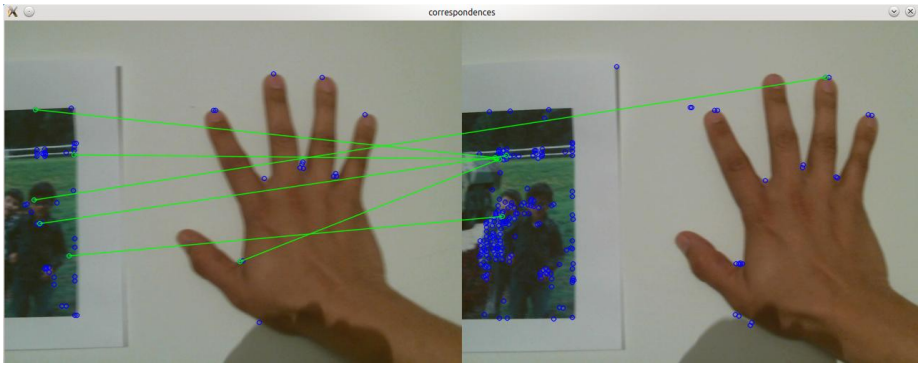


Figura 27. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector FAST y descriptor FREAK.

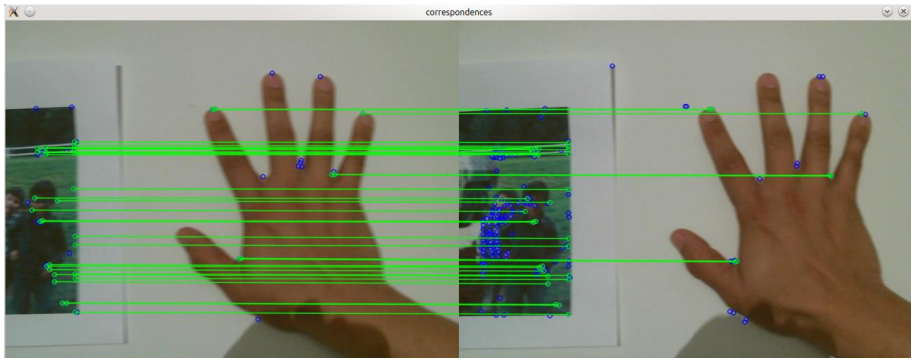


Figura 28. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector FAST y descriptor ORB.

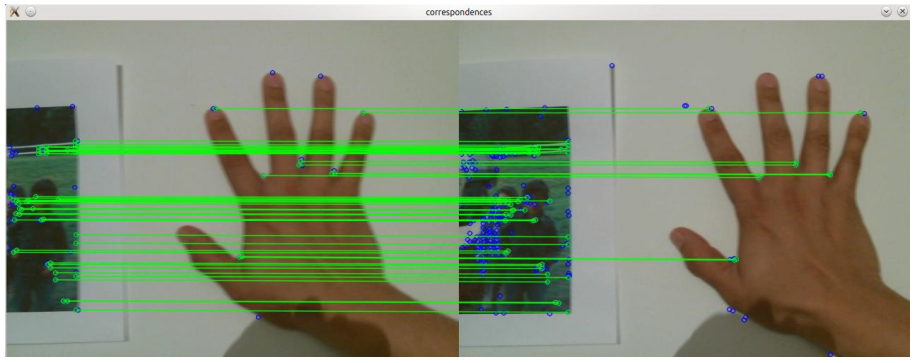


Figura 29. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector FAST y descriptor SIFT.

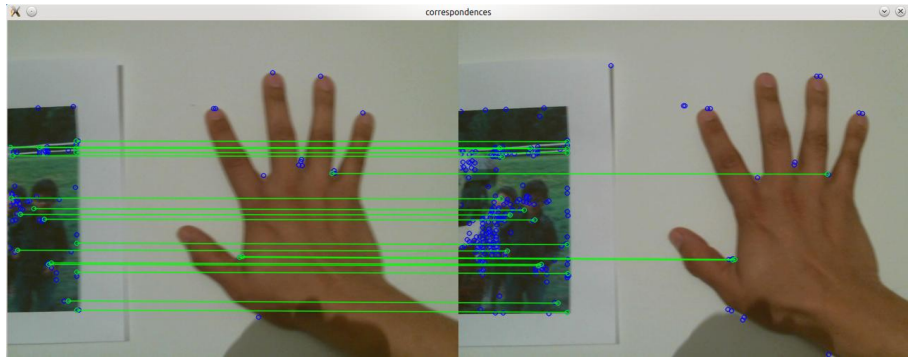


Figura 30. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector FAST y descriptor SURF.

	BRISK	FREAK	ORB	SIFT	SURF
Tiempo de detector [s]	0.01771936	0.01934176	0.01845214	0.01817654	0.01703554
Tiempo de descriptor[s]	0.01952059	0.1779165	0.1103313	0.78159	0.1027906
Tiempo de correspondencia [s]	0.0372281	0.432312	0.0239088	0.0319456	0.0579341

Tabla 8. Comparación de tiempos de respuesta detector FAST, variando el descriptor.

- **Detector SIFT, varia descriptor**

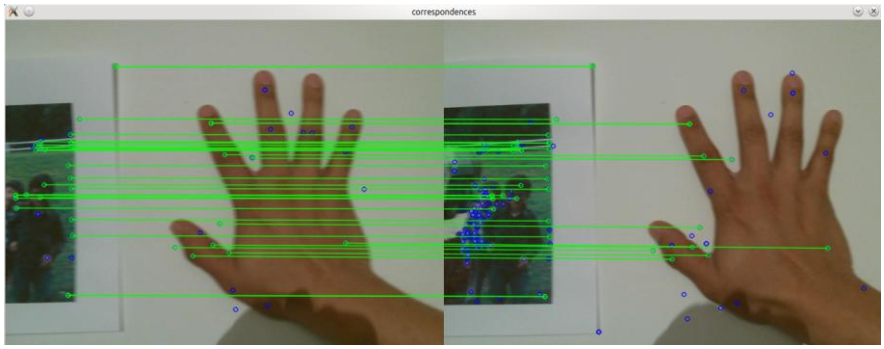


Figura 31. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SIFT y descriptor BRISK.

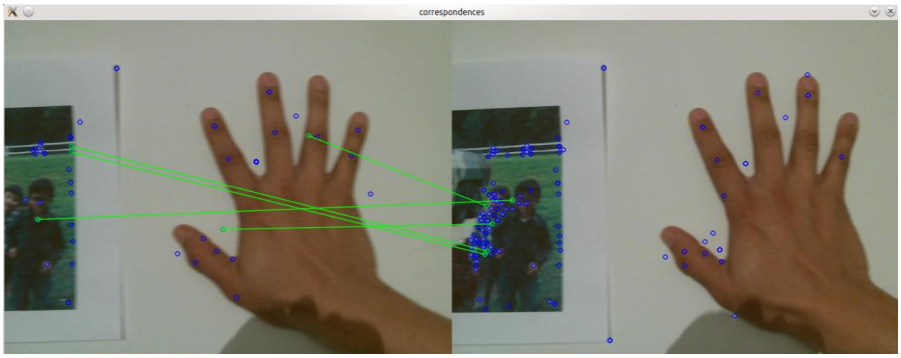


Figura 32. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SIFT y descriptor FREAK.

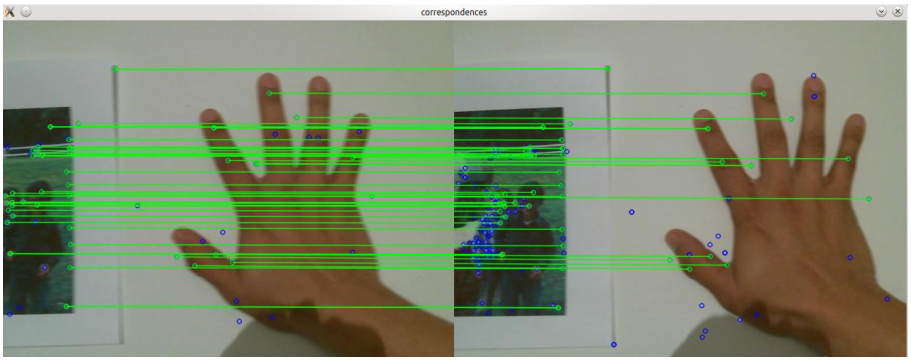


Figura 33. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SIFT y descriptor SIFT.

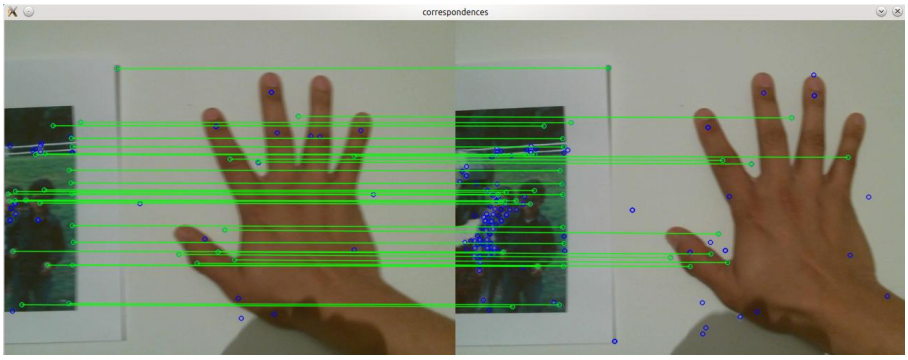


Figura 34. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SIFT y descriptor SURF.

	BRISK	FREAK	ORB	SIFT	SURF
Tiempo de detector [s]	2.22962	2.23652	-	2.23925	2.24116
Tiempo de descriptor[s]	0.01748532	0.1678096	-	2.18577	0.1534685
Tiempo de correspondencia [s]	0.0198505	0.351579	-	0.0259196	0.0224718

Tabla 9. Comparación de tiempos de respuesta detector SIFT, variando el descriptor.

- **Detector SURF, varia descriptor**

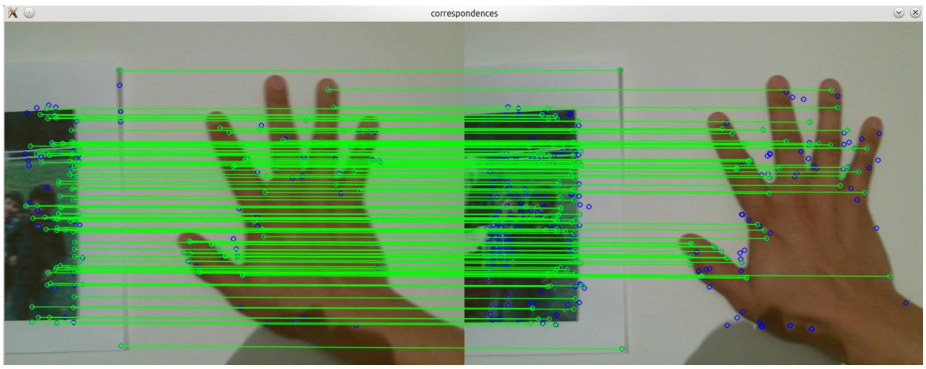


Figura 35. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SURF y descriptor BRISK.

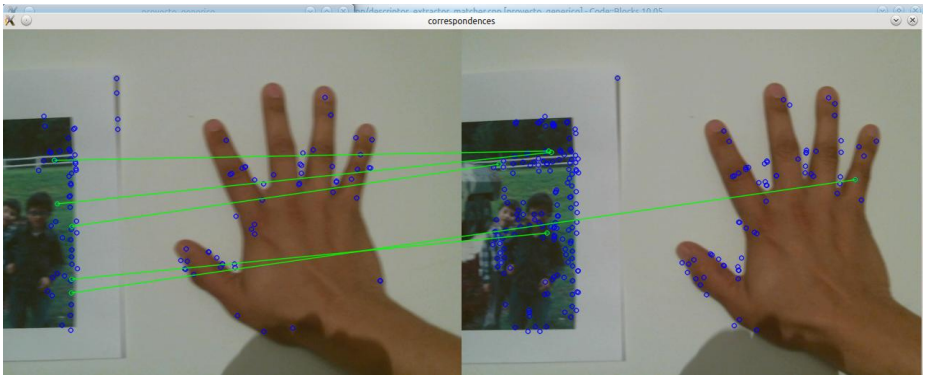


Figura 36. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SURF y descriptor FREAK.

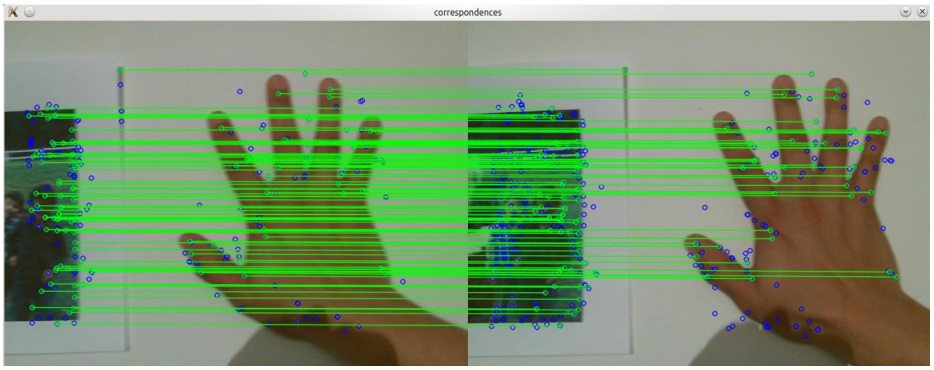


Figura 37. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SURF y descriptor ORB.

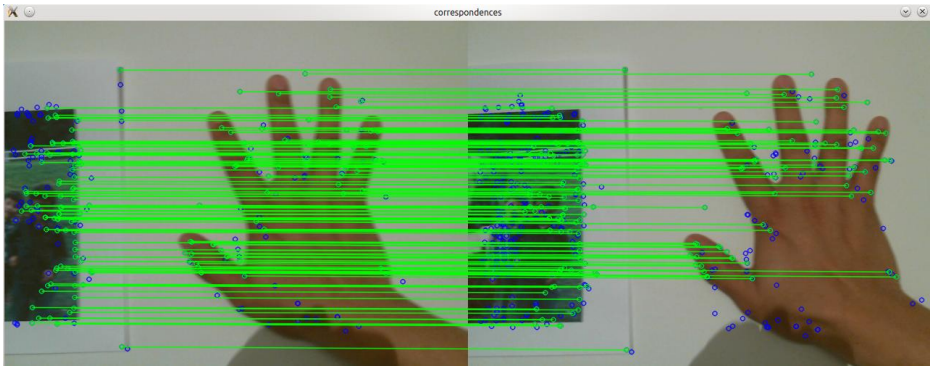


Figura 38. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SURF y descriptor SIFT.

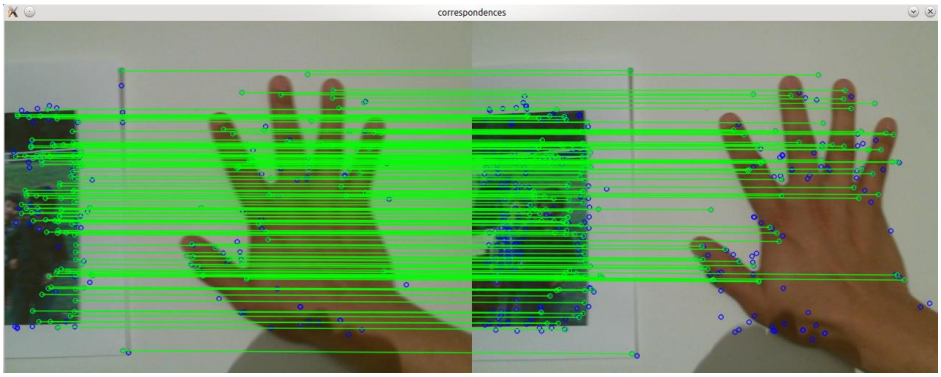


Figura 39. Prueba del algoritmo Detector_descriptor_matcher, utilizando detector SURF y descriptor SURF.

	BRISK	**FREAK**	**ORB**	**SIFT**	**SURF**
Tiempo de detector [s]	0.652871	0.712697	0.72093	0.694836	0.705548
Tiempo de descriptor[s]	0.0279614	0.1785513	0.1150178	6.54142	1.127971
Tiempo de correspondencia [s]	0.0613943	0.437997	0.0586747	0.10394	0.0787963

Tabla 10. Comparación de tiempos de respuesta detector SURF, variando el descriptor.

Teniendo en cuenta las pruebas anteriores, se decidió trabajar este algoritmo con el descriptor BRISK , ya que muestra, mayor correspondencia de puntos y menos errores, cuando se observan las imágenes, logra obtener resultados acertados, con un número de puntos en la imagen, apropiados para los cálculos posteriores aunque los tiempos de ejecución son más lentos.

4.4. DESARROLLO COMPLEMENTARIO DE ALGORITMOS

Para poder trabajar con los descriptores elegidos, fue necesario añadir a estos en una primera parte, un algoritmo de captura de video y posteriormente convertir este video en fotogramas o cuadros de imágenes que pudieran ser procesados, con los algoritmos existentes. La generación de estos fotogramas, debía poder controlarse, para el adecuado funcionamiento de los algoritmos escogidos. Y se realizó mediante una función sencilla en C++,

4.4.1. Algoritmo de captura de video y conversión a fotogramas

```
    // Load Images
    CvCapture * pCapture = 0;
    int i;
    char filename[50];
    // Inicializar video de captura
    pCapture = cvCaptureFromAVI( "capture_18_lpl.avi" );
    if( !pCapture )
    {
            fprintf(stderr, "Ocufile:///home/alejandra/Video/LP/capture-11.avirrio una falla al momento
            de iniciar la captura de video.\n");
            return -1;
    }


    IplImage *imgA  = cvQueryFrame( pCapture );
    IplImage *imgB;
    if( !imgA ){
            fprintf(stderr, "Ocurrio una falla al obtener un cuadro de iuntitledmagen\n");
    }

imgB=cvCloneImage(imgA);
    for(int p=0; p<TAM_MAXIMO; p++)
    {

            imgA= cvQueryFrame( pCapture );


            if( !imgB ) {
                    std::cout << " --(!) Error reading image 2" << std::endl;
                    break;
            }
```

4.4.2. Resultado del algoritmo de captura de video y conversión a fotogramas

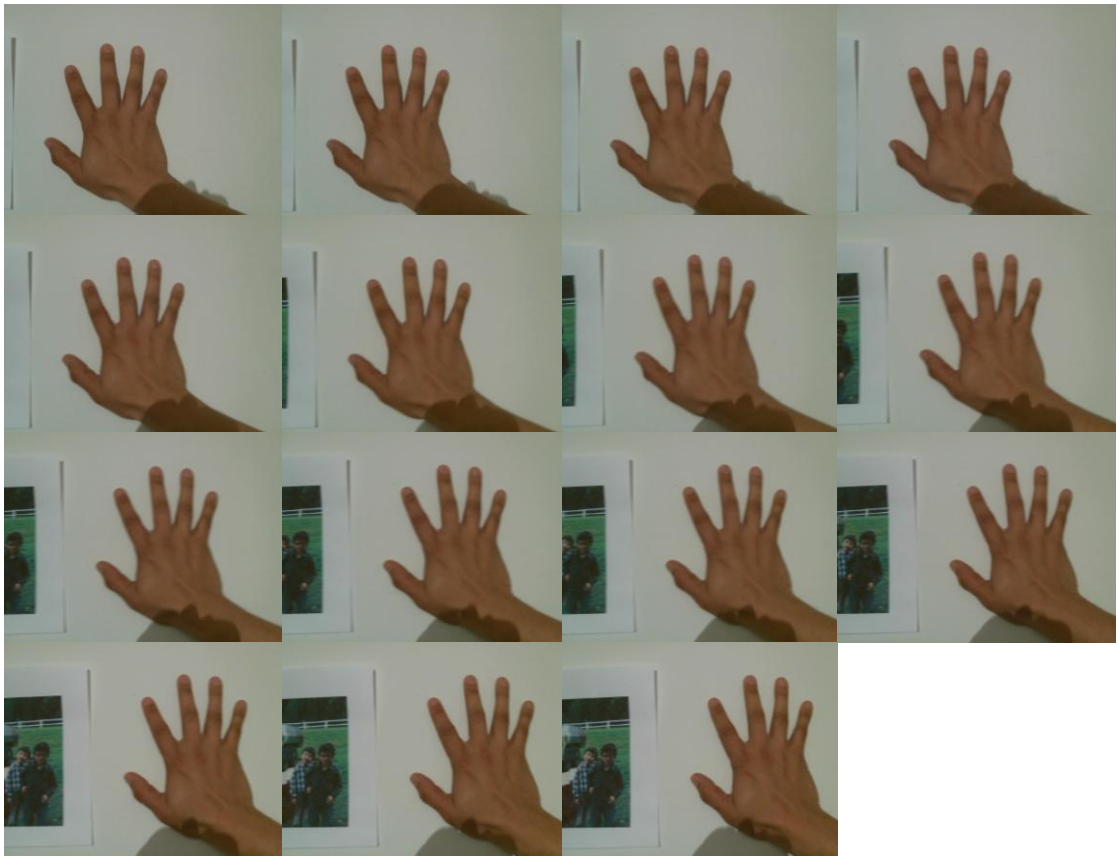


Figura 40. Secuencia de fotogramas o imágenes tomadas del video.

Luego de realizar la captura del video y de guardar temporalmente los fotogramas se realiza el procesamiento de detección, descriptor y correspondencia, explicados anteriormente, mediante un sencillo llamado de las funciones ya existentes en las librerías de Opencv, en un ciclo que puede capturar hasta nueve mil fotogramas por video, según sea el tiempo de duración de cada video.

Posteriormente y fuera del ciclo anterior, se estimó la distancia euclidiana (Ecuación 11), entre cada punto a lo largo de cada video, y se exportó esta información a un archivo .csv, para su posterior procesamiento en Matlab. También se calcularon las distancias recorridas en y y en x, por cada par de puntos, mediante la resta de los puntos (x2 – x1) y (y2-y1), respectivamente.

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ecuación 11. Ecuación para estimar la distancia euclidiana entre dos puntos

Para controlar la rapidez de cada algoritmo, se generaron también archivos .csv, que contienen todos los tiempos, que se demoraron los algoritmos en procesar la información, desde que inicia a detectar hasta que termina en la correspondencia de puntos, los resultados se encuentran en los anexos del 3 al 8, en los que se puede observar el tiempo en cada proceso, para hallar el tiempo total, se realizó, la suma de estos tiempos y el promedio, para

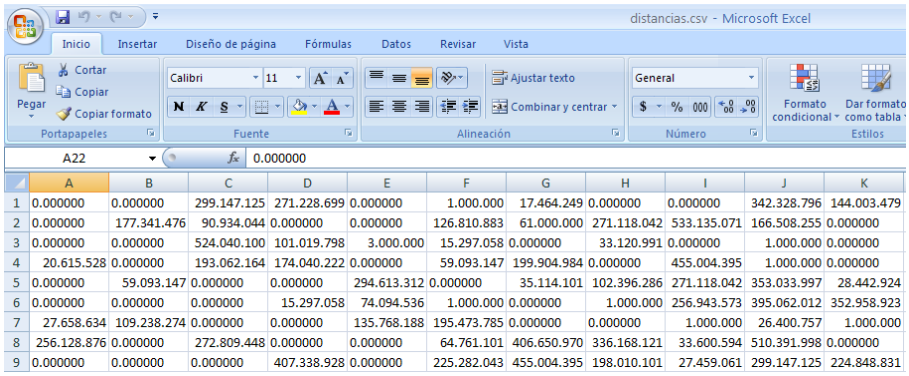
determinar cuánto tiempo se demora cada algoritmo en realizar cada paso (detector, descriptor, correspondencia).

4.4.3. Algoritmo de estimación de distancia recorrida en pixeles

```
for( size_t m = 0; m < matches.size(); m++)
{
        int i1 = matches[m].queryIdx;
        int i2 = matches[m].trainIdx;

        const KeyPoint &kp1 = keypointsA[i1], &kp2 = keypointsB[i2];


        float distanciax;
        float distanciay;

        restax[m]=(kp2.pt.x - kp1.pt.x);
        restay[m]=(kp2.pt.y - kp1.pt.y);
        distanciax=restax[m];
        distanciay=restay[m];

        fprintf (pFile5, "%f;",distanciax);
        fprintf (pFile6, "%f;",distanciay);

        float auxiliar;

        distancia[m]=sqrt(pow((kp1.pt.y - kp2.pt.y),2)+pow((kp1.pt.x - kp2.pt.x),2));
        auxiliar=distancia[m];

         fprintf (pFile1, "%f;",auxiliar);

}
```

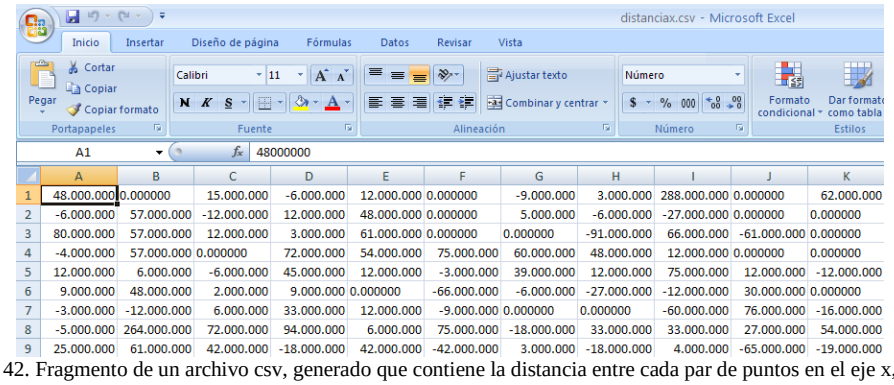
4.4.4. Resultado, se obtenido del algoritmo de estimación de distancia recorrida

Los resultados se pueden ver completos en los anexos 3 al 8, a continuación se muestra un fragmento de los archivos generados por el algoritmo, en que se puede observar el desplazamiento en pixeles de cada punto, frente a su ubicación en el cuadro anterior, de los videos tomados (Ver anexos); esta estimación de la distancia se realizó por el método euclidiano de la Ecuación 11, y por resta de puntos como se ve a continuación. (Figura 41, Figura 42, Figura 43)



	A	B	C	D	E	F	G	H	I	J	K
1	0.000000	0.000000	299.147.125	271.228.699	0.000000	1.000.000	17.464.249	0.000000	0.000000	342.328.796	144.003.479
2	0.000000	177.341.476	90.934.044	0.000000	0.000000	126.810.883	61.000.000	271.118.042	533.135.071	166.508.255	0.000000
3	0.000000	0.000000	524.040.100	101.019.798	3.000.000	15.297.058	0.000000	33.120.991	0.000000	1.000.000	0.000000
4	20.615.528	0.000000	193.062.164	174.040.222	0.000000	59.093.147	199.904.984	0.000000	455.004.395	1.000.000	0.000000
5	0.000000	59.093.147	0.000000	0.000000	294.613.312	0.000000	35.114.101	102.396.286	271.118.042	353.033.997	28.442.924
6	0.000000	0.000000	0.000000	15.297.058	74.094.536	1.000.000	0.000000	1.000.000	256.943.573	395.062.012	352.958.923
7	27.658.634	109.238.274	0.000000	0.000000	135.768.188	195.473.785	0.000000	0.000000	1.000.000	26.400.757	1.000.000
8	256.128.876	0.000000	272.809.448	0.000000	0.000000	64.761.101	406.650.970	336.168.121	33.600.594	510.391.998	0.000000
9	0.000000	0.000000	0.000000	407.338.928	0.000000	225.282.043	455.004.395	198.010.101	27.459.061	299.147.125	224.848.831

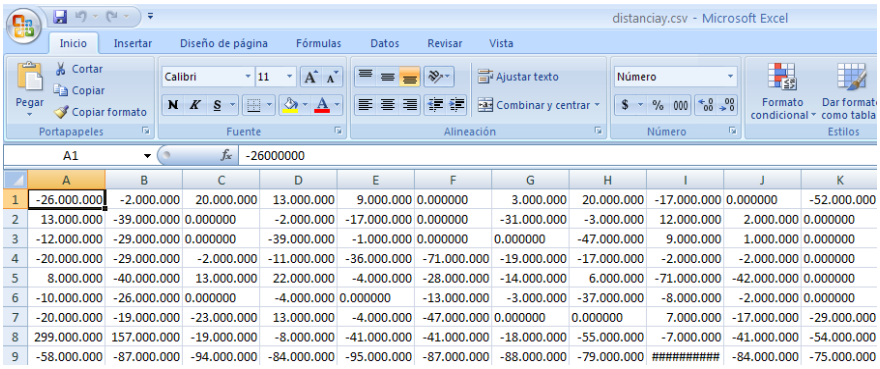
Figura 41. Fragmento de un archivo csv, generado que contiene la distancia euclidiana entre cada par de puntos, encontrados por fotograma.



distanciax.csv - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K
1	48.000.000	0.000000	15.000.000	-6.000.000	12.000.000	0.000000	-9.000.000	3.000.000	288.000.000	0.000000	62.000.000
2	-6.000.000	57.000.000	-12.000.000	12.000.000	48.000.000	0.000000	5.000.000	-6.000.000	-27.000.000	0.000000	0.000000
3	80.000.000	57.000.000	12.000.000	3.000.000	61.000.000	0.000000	0.000000	-91.000.000	66.000.000	-61.000.000	0.000000
4	-4.000.000	57.000.000	0.000000	72.000.000	54.000.000	75.000.000	60.000.000	48.000.000	12.000.000	0.000000	0.000000
5	12.000.000	6.000.000	-6.000.000	45.000.000	12.000.000	-3.000.000	39.000.000	12.000.000	75.000.000	12.000.000	-12.000.000
6	9.000.000	48.000.000	2.000.000	9.000.000	0.000000	-66.000.000	-6.000.000	-27.000.000	-12.000.000	30.000.000	0.000000
7	-3.000.000	-12.000.000	6.000.000	33.000.000	12.000.000	-9.000.000	0.000000	0.000000	-60.000.000	76.000.000	-16.000.000
8	-5.000.000	264.000.000	72.000.000	94.000.000	6.000.000	75.000.000	-18.000.000	33.000.000	33.000.000	27.000.000	54.000.000
9	25.000.000	61.000.000	42.000.000	-18.000.000	42.000.000	-42.000.000	3.000.000	-18.000.000	4.000.000	-65.000.000	-19.000.000

Figura 42. Fragmento de un archivo csv, generado que contiene la distancia entre cada par de puntos en el eje x, con su respectiva dirección, encontrados por fotograma.



distanciay.csv - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K
1	-26.000.000	-2.000.000	20.000.000	13.000.000	9.000.000	0.000000	3.000.000	20.000.000	-17.000.000	0.000000	-52.000.000
2	13.000.000	-39.000.000	0.000000	-2.000.000	-17.000.000	0.000000	-31.000.000	-3.000.000	12.000.000	2.000.000	0.000000
3	-12.000.000	-29.000.000	0.000000	-39.000.000	-1.000.000	0.000000	0.000000	-47.000.000	9.000.000	1.000.000	0.000000
4	-20.000.000	-29.000.000	-2.000.000	-11.000.000	-36.000.000	-71.000.000	-19.000.000	-17.000.000	-2.000.000	-2.000.000	0.000000
5	8.000.000	-40.000.000	13.000.000	22.000.000	-4.000.000	-28.000.000	-14.000.000	6.000.000	-71.000.000	-42.000.000	0.000000
6	-10.000.000	-26.000.000	0.000000	-4.000.000	0.000000	-13.000.000	-3.000.000	-37.000.000	-8.000.000	-2.000.000	0.000000
7	-20.000.000	-19.000.000	-23.000.000	13.000.000	-4.000.000	-47.000.000	0.000000	0.000000	7.000.000	-17.000.000	-29.000.000
8	299.000.000	157.000.000	-19.000.000	-8.000.000	-41.000.000	-41.000.000	-18.000.000	-55.000.000	-7.000.000	-41.000.000	-54.000.000
9	-58.000.000	-87.000.000	-94.000.000	-84.000.000	-95.000.000	-87.000.000	-88.000.000	-79.000.000	##########	-84.000.000	-75.000.000

Figura 43. Fragmento de un archivo csv, generado que contiene la distancia entre cada par de puntos en el eje y, con su respectiva dirección, encontrados por fotograma.

La siguiente tabla, muestra los tiempos de ejecución aproximado de los algoritmos FREAK y BRISK, en el procesamiento de cada video, comparado con los el tiempo de ejecución aproximado del algoritmo de Lukas y Kanade.[22]

	TIEMPO DE EJECUCIÓN EN (S)		
	FREAK	BRISK	LUKAS Y KANADE
LP FRONTAL 18	171,242	533,032	4,464539
LP FRONTAL 36	169,77	527,369	4,297852
LP FRONTAL 54	170,988	529,051	4,367546
SW FRONTAL 18	172,386	541,346	4,521676
SW FRONTAL 36	144,499	454,55	4,521676
SW FRONTAL 54	137,746	429,508	4,521676
LP LATERAL 18	156,159	489,552	4,521676
LP LATERAL 36	138,395	432,364	4,521676
LP LATERAL 54	141,466	441,758	4,521676
SW LATERAL 18	157,039	474,946	4,521676
SW LATERAL 36	149,885	469,46	4,521676
SW LATERAL 54	111,354	433,708	4,521676
TIEMPO PROMEDIO DE EJECUCIÓN (s)	151,7440833	479,7203333	4,485418417

Tabla 11. Tabla comparativa de los tiempos de ejecución de los algoritmos trabajados

[22] Ver anexos 3 -8: pruebas en escenario controlado, y en escenario sin control

4.4.5. Estimación de la distancia total recorrida por la serpiente, desarrollado en Matlab.

Para obtener el desplazamiento total realizado por la serpiente, se utilizaron las funciones mean( ), var( ) y median( ), en Matlab, que permitían por medio de la media, la mediana y la varianza, obtener el desplazamiento total del robot, a partir de los archivos generados en CSV, de manera gráfica, de tal forma que se pudieran comparar los resultados obtenidos por cada algoritmo con el de referencia de Lukas y Kanade.

4.4.6. Resultado final de la estimación de la distancia euclidiana recorrida en pixeles, por la serpiente, tomando los datos previamente generados

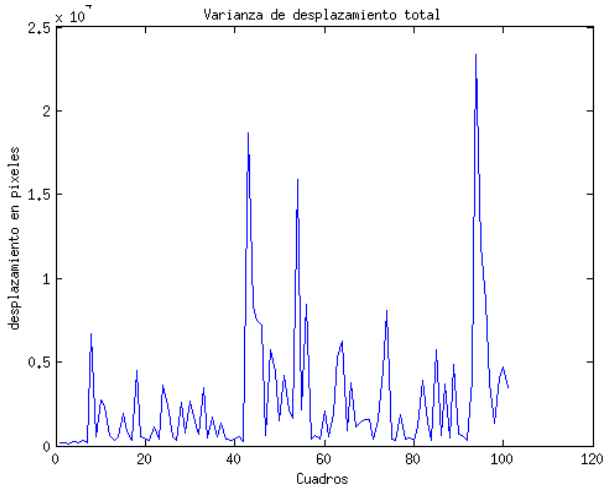


Figura 44. Grafica de la variancia, del video LP18 con descriptor BRISK

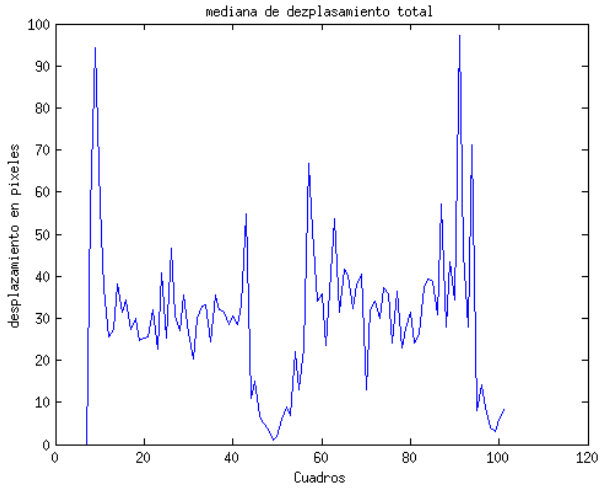


Figura 45. Grafica de la mediana, del video LP18 con descriptor BRISK

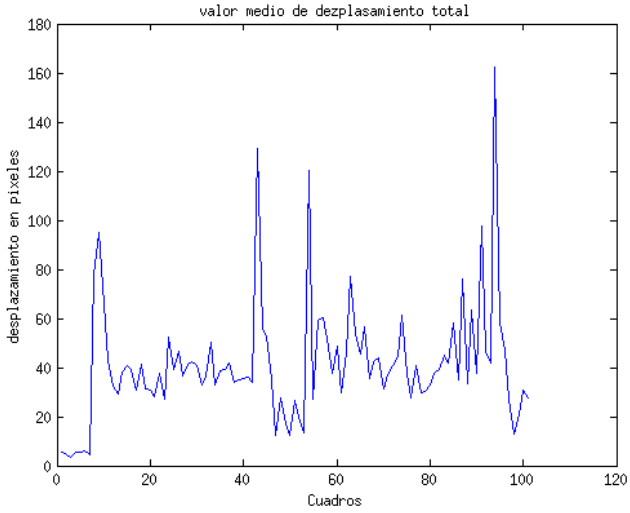


Figura 46. Grafica del valor medio, del video LP18 con descriptor BRISK

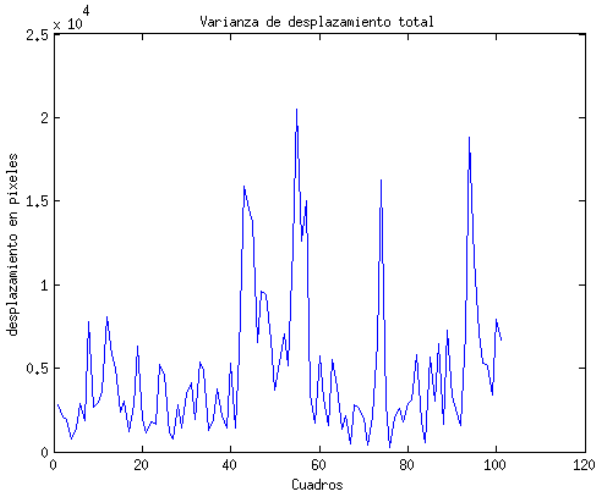


Figura 47. Grafica de la variancia, del video LP18 con descriptor FREAK

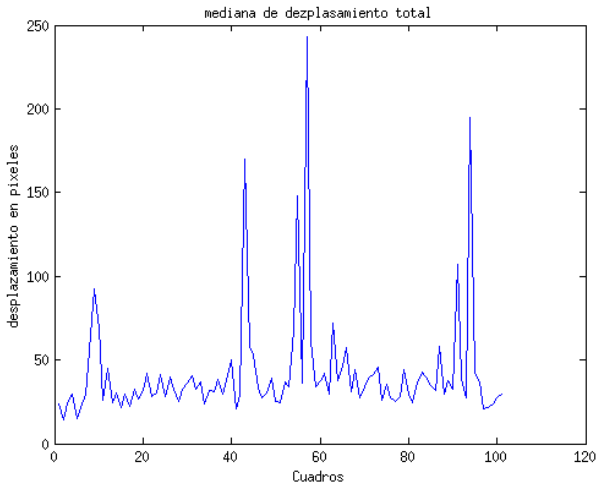


Figura 48. Grafica de la mediana, del video LP18 con descriptor FREAK

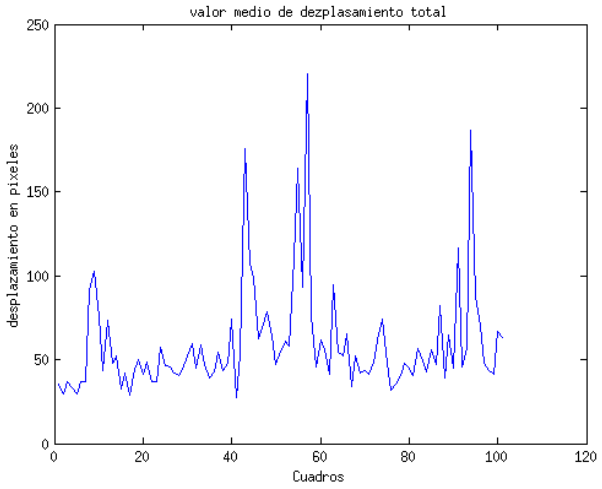


Figura 49. Grafica del valor medio, del video LP18 con descriptor FREAK

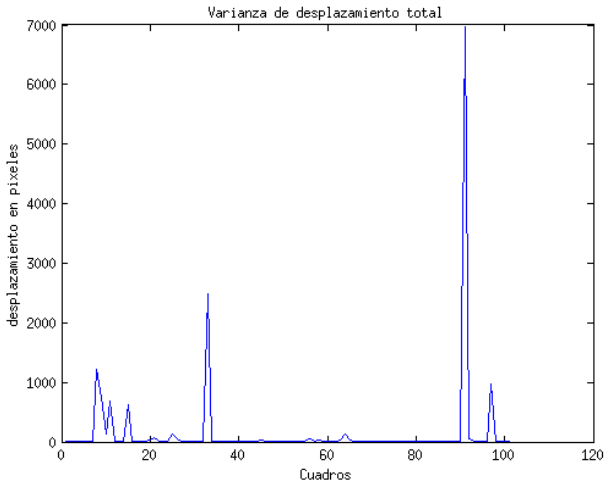


Figura 50. Grafica de la variancia, del video LP18 con el algoritmo de referencia de LK

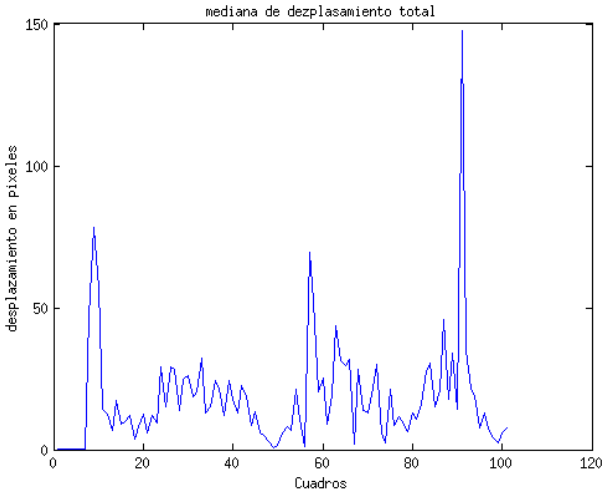


Figura 51. Grafica de la mediana, del video LP18 con el algoritmo de referencia de LK

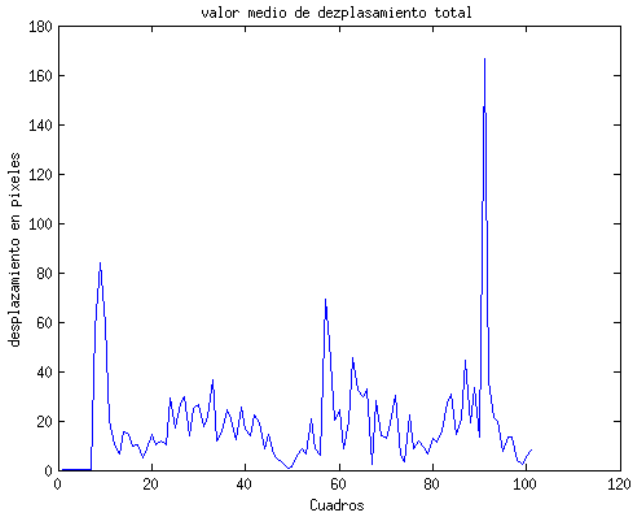


Figura 52. Grafica del valor medio, del video LP18 con el algoritmo de referencia de LK

Las gráficas representan la aproximación de la distancia euclidiana en pixeles, en los 100 cuadros por video, que se realizaron para este trabajo de grado. Las demás gráficas pueden observarse en el anexo 10

4.5. ESTIMACIÓN DE DEZPLAZAMIENTO

4.4.1. Estimación gráfica de la media, la mediana y la varianza, comparando los tres algoritmos.

La comparación de la estimación gráfica del desplazamiento se realizó mediante gráficas comparativas de la media, la mediana y la varianza, de la siguiente forma, teniendo en cuenta que la gráfica de color azul, representa al algoritmo FREAK, la verde s BRISK y la roja, es el algoritmo de referencia de flujo óptico piramidal de Lukas y Kanade (LK):

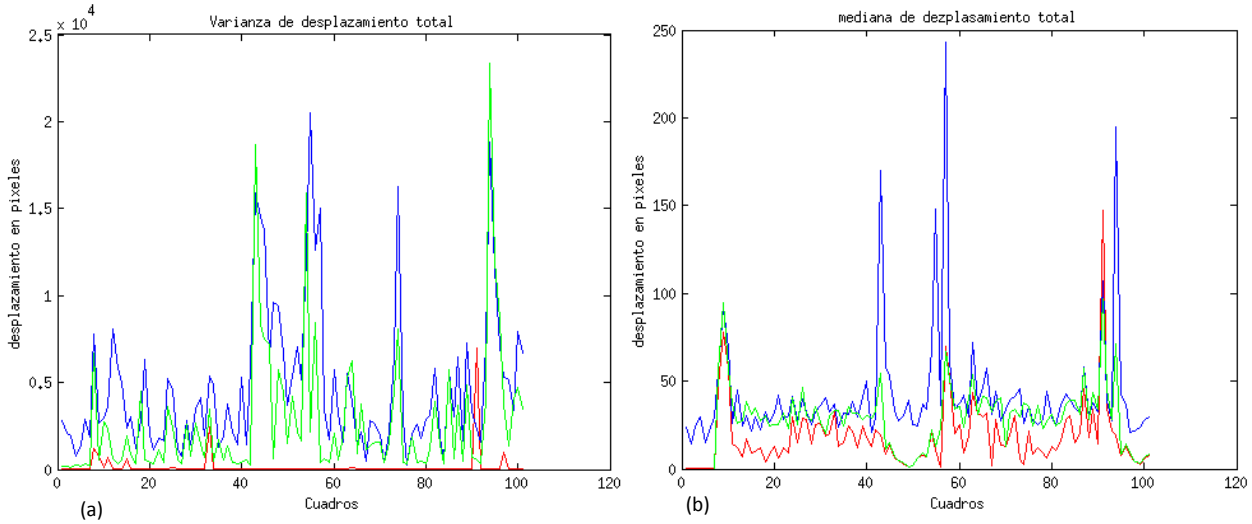


Figura 53. Gráficas comparativas, de la variancia(a), la mediana (b), del video LP18 con BRISK, FREAK y LK

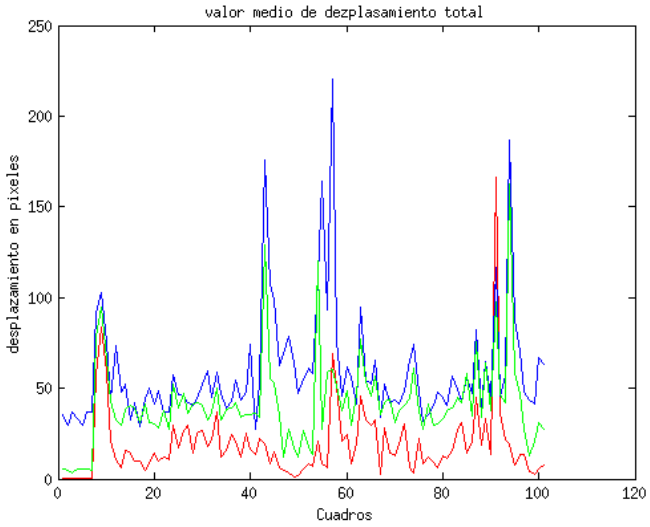


Figura 54. Gráficas comparativa del valor medio, del video LP18 con BRISK, FREAK y LK

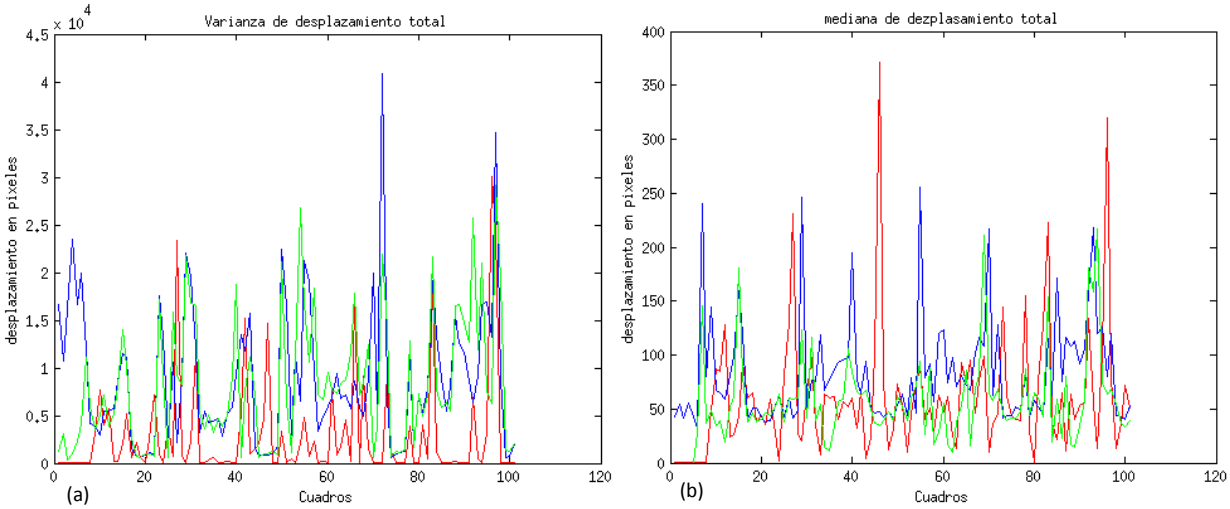


Figura 55. Gráficas comparativas, de la variancia(a), la mediana (b), del video SW36 con BRISK, FREAK y LK

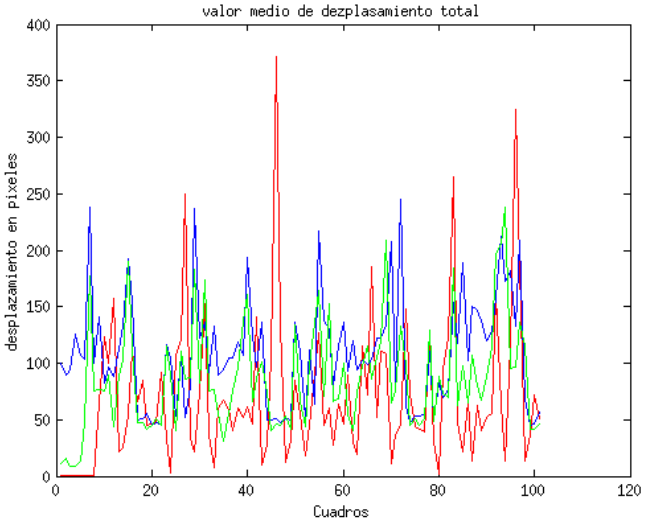


Figura 56. Gráfica comparativa del valor medio, del video SW36 con BRISK, FREAK y LK

En la Figura 53, se observa el comportamiento de los tres algoritmos, en el movimiento (gait) Linear Progression, con la cámara en el frente del robot y en la Figura 55, se observan las gráficas correspondientes al video con el movimiento Side Winding, con la cámara igualmente en frente, las demás gráficas de comparación se pueden apreciar en el anexo 11.

4.4.2. Estimación manual del desplazamiento en pixeles.

La estimación del desplazamiento de forma manual, fue utilizada para comparar más adelante, los resultados obtenidos con los tres algoritmos y así, poder analizar qué tan cercano es el resultado al desplazamiento en pixeles; para ello fue necesario utilizar el software de edición de video *GIMP Image editor*, que tiene la posibilidad de establecer un sistema coordenado, en la esquina superior izquierda, y determinando el punto en la imagen a medir, se realizó la verificación de la ubicación de este punto en el inicio del video y al final de él, como se muestra a continuación. (Los cuadros con los respectivos puntos escogidos se encuentran en los anexos, en la carpeta Pruebas manuales)

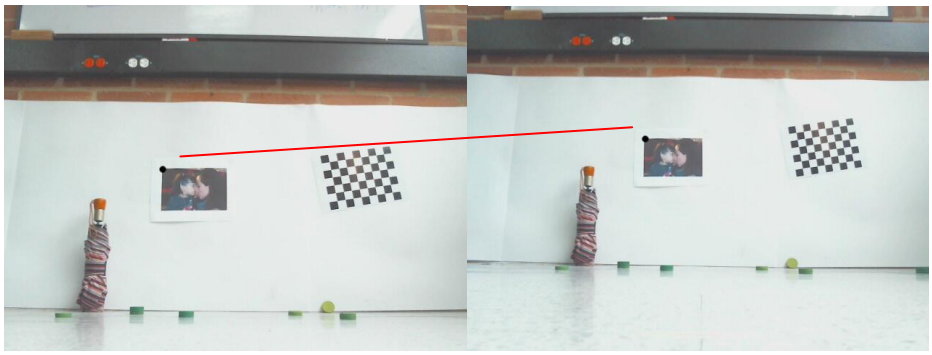


Figura 57. Ejemplo de cálculo manual del desplazamiento entre el fotograma inicial y el final, del video LP18, con la cámara ubicada al frente

4.4.3. Comparación de los datos encontrados manualmente y por aproximación de los resultados obtenidos con los algoritmos trabajados.

Para comparar que tan próximos pueden ser los resultados de los algoritmos BRISK y FREAK con el de referencia de Lukas y Kanade y con la comparación del cálculo manual, se encontraron en primera parte el promedio del desplazamiento en las coordenada x y y, de los tres algoritmos trabajados, estos datos fueron sacados de los archivos que se generan al procesar los videos con cada algoritmo; luego por medio de la Ecuación 11. Ecuación para estimar la distancia euclidiana entre dos puntos, se determina la distancia recorrida por el robot serpiente en pixeles, en cada video de la base de datos, y se relaciona en la siguiente tabla con los datos obtenido manualmente.

	Distancia recorrida en pixeles FREAK	Distancia recorrida en pixeles BRISK	Distancia recorrida en pixeles LK
LP FRONTAL 18	19850.34553	12893.91752	5349.73424
LP FRONTAL 36	26332.05343	19923.91687	10978.8663
LP FRONTAL 54	34974.78425	32966.27934	18541.1158
SW FRONTAL 18	25112.98612	15742.94559	4852.19668
SW FRONTAL 36	36275.47785	28118.17408	9736.24256
SW FRONTAL 54	32306.59437	34383.48492	15156.5581
LP LATERAL 18	31809.33999	13278.68301	2534.60089
LP LATERAL 36	35870.29466	23804.38333	3721.04894
LP LATERAL 54	33955.45225	26606.89112	4972.82568
SW LATERAL 18	33387.66571	32138.87807	6702.03011
SW LATERAL 36	47257.35056	44541.43576	12964.9952
SW LATERAL 54	37954.38245	35449.91388	15374.1135

Tabla 12. Comparación de la distancia recorrida en pixeles con los algoritmos FREAK BRISK y LK

Como se observa en la tabla 12**¡Error! No se encuentra el origen de la referencia.**, los algoritmos FREAK y BRISK, son bastante aproximados entre sí, aunque difieren aproximadamente en el doble de los resultados del algoritmo de referencia de flujo óptico piramidal de Lukas y Kanade.

5. ANALISIS DE RESULTADOS

En este capítulo se presenta una descripción de los resultados experimentales obtenidos al procesar cada video con los algoritmos descriptores FREAK, BRISK y el algoritmo de referencia de flujo óptico piramidal de Lukas y Kanade.

5.1. Detección de puntos

La detección de puntos en este trabajo de grado se realizó por medio de un detector conocido en el estado del arte, con la función de las librerías de OpenCv, *"GoodFeaturesToTackDetection detector ( )"*, se realizó este proceso después de una comparación con otros detectores como SURF y FAST, encontrando, como se muestra en la Tabla 7, que tiene mejor tiempo de respuesta (en promedio 0,3481 s) que SURF (en promedio 0,861627 s), pero es más lento que FAST (en promedio 0,00785 s), lo que lo hace una mejor opción en cuanto a rapidez se refiere.

Aunque el detector FAST, es realmente rápido, se encontró que no es eficiente cuando la imagen del video se encuentra con figuras suavizadas, sombras y cambios de iluminación, lo que puede afectar considerablemente, su capacidad de detectar y estimar los puntos importantes de la imagen, como se ve en la figura 23; en cambio SURF, es un excelente algoritmo que utiliza u método aleatorio, para encontrar los puntos importantes, lo que lo hace robusto ante los cambios en el escenario mencionados anteriormente, por esta razón su tiempo de ejecución puede ser mayor de lo esperado, como se muestra en la figura 24; finalmente GFTD (*Good Features to Tack Detector*), se basa en el método de detección de puntos de Harris, rechazando aquellos que se encuentran muy cercanos de acuerdo al umbral establecido, lo que lo hace muy bueno para la detección de puntos, con un tiempo de ejecución menor que SURF.

Se comprobó que GFTD como detector, cumple satisfactoriamente su función como detector, obteniendo los puntos importantes de la imagen para su posterior procesamiento, tomando un

tiempo de aproximado de 91 s, por cada video de 10 s de duración, mientras el también muy buen detector SURF toma un tiempo aproximado de ejecución de 245 s, por cada video de 10 segundos. Por esta razón GFTD, fue elegido para ser el detector utilizado en el procesamiento de todos los videos, recopilados en la base de datos.

5.2. Descriptores y correspondencia de puntos

Después de evaluar el comportamiento de la mayoría de descriptores encontrados en el estado del arte, se encontró que los dos descriptores que podrían aportar en cuanto a mayor velocidad de ejecución y robustez eran FREAK y BRISK, algoritmos con los que se realizaron las pruebas correspondientes y se encontró que:

- Aunque FREAK fue descrito como superior a otros descriptores en el marco teórico, en la práctica, se vio altamente superado por descriptores como BRISK y ORB, lo cual sugiere que FREAK no es tan flexible con los detectores de puntos que utiliza previo a la descripción, en comparación con los demás mencionados en el marco teórico. Además presenta mayor varianza, en el procesamiento debido a que, es un algoritmo sensible al ruido como sombras y movimientos fuertes.

  A pesar de lo anterior, presenta resultados muy cercanos al del algoritmo de referencia de flujo óptico piramidal de Lukas y Kanade, lo que demuestra, que con el detector de puntos apropiado, es una buena opción en cuanto al tiempo de ejecución y es asertivo, en cuanto al cálculo del desplazamiento del robot en pixeles, en la Figura 58, se ve como su varianza es mayor en el movimiento (gait) *Side Sinding* en relación con los otros algoritmos, a diferencia de la varianza en el movimiento Linear Progression, en el que está muy aproximado a los demás. Esta comparación se realizó con los demás videos, tanto con la media, la mediana y la varianza (gráficas en el anexo 11), lo que mostró que FREAK, se aleja más del algoritmo de referencia, pero está lo suficientemente cerca, para probar que es bueno en tanto a la aproximación del desplazamiento del robot serpiente, sobre todo cuando la cámara se encuentra en frente.

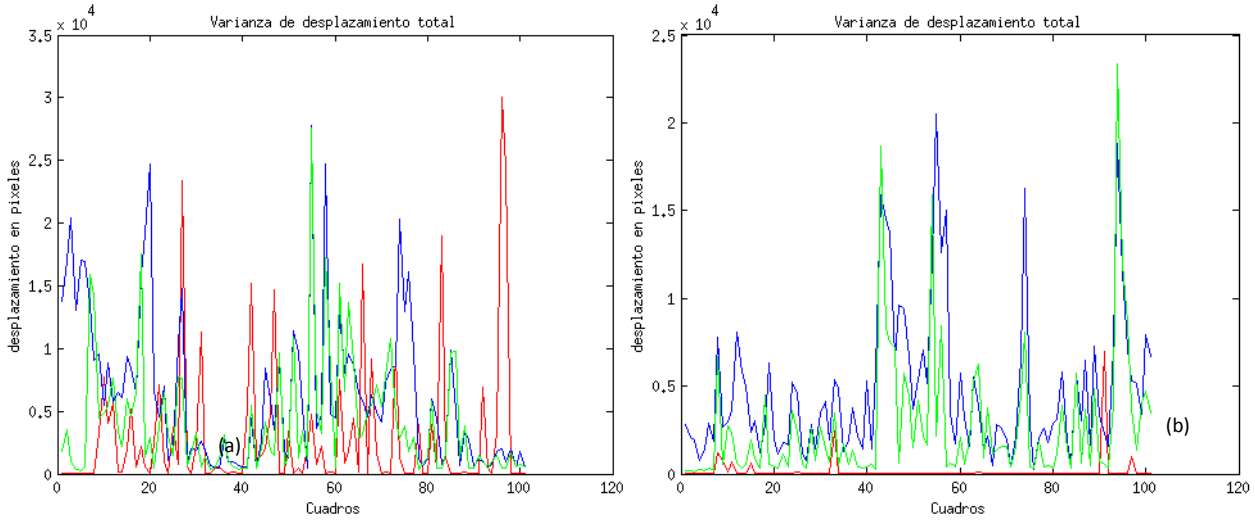


Figura 58. Gráfica comparativa de la varianza en los tres algoritmos (FREAK color azul, BRISK color verde y LK color rojo), en los videos LP18 (a) y SW18

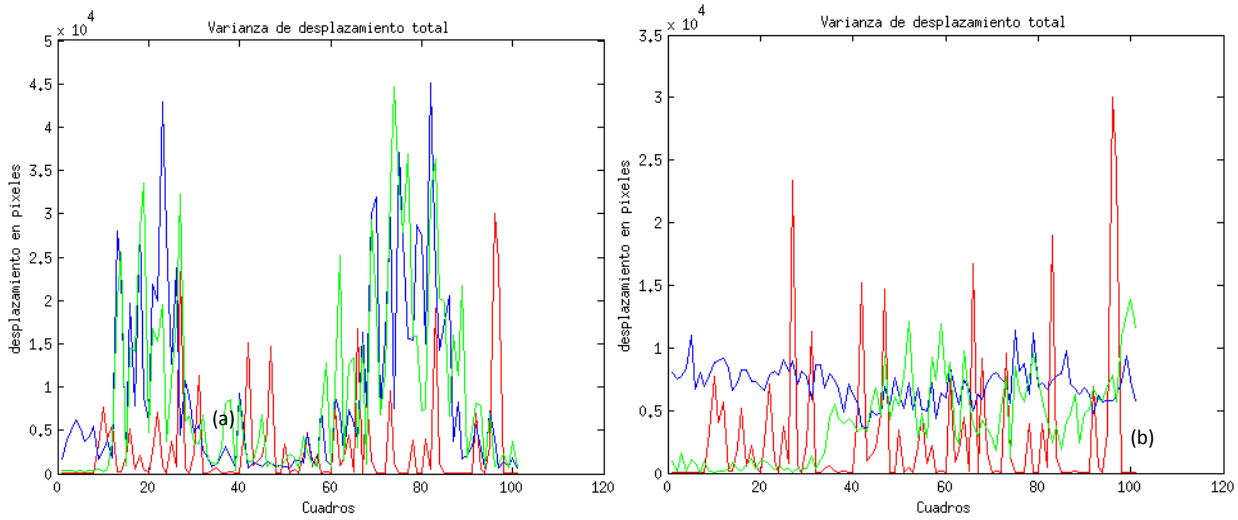


Figura 59. Gráfica comparativa de la varianza en los tres algoritmos (FREAK color azul, BRISK color verde y LK color rojo), en los videos LPL18 (a) y SWL18

- Cuando se utiliza el algoritmo, con el descriptor BRISK, el tiempo de ejecución es mayor, pero los resultados son más cercanos a los generados por el algoritmo de referencia, también se puede observar, que en las velocidades lentas, presenta menor variación en la mediana, con respecto a FREAK. (ver anexo 11)

  Durante la ejecución de los algoritmos se evidenciaba cómo BRISK presentaba menos errores que FREAK, en la hora de hacer la correspondencia, esto se evidencia en la Figura 60 y en la Figura 61, donde se compara en el mismo cuadro de imagen la correspondencia dibujados en los fotogramas, lo que demuestra una vez más que como descriptor BRISK es más robusto ante el ruido, generado por el cambio de iluminación, la pérdida de los objetos por el desplazamiento del robot, y sombras.

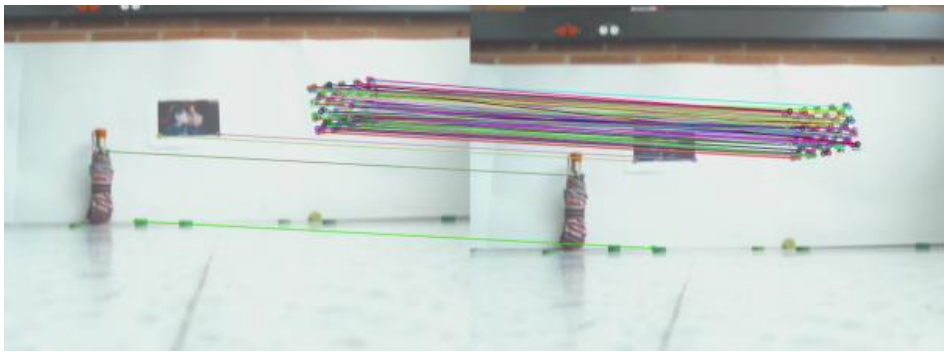


Figura 60. Funcionamiento gráfico del algoritmo BRISK

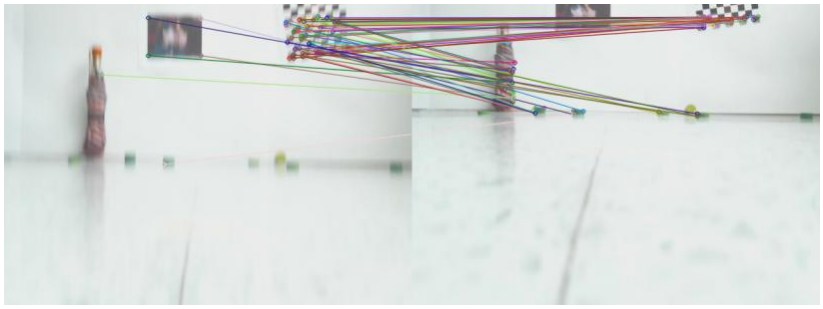


Figura 61. Funcionamiento gráfico del algoritmo FREAK

- En los videos realizados en escenarios no controlados, se muestra que BRISK, no es una algoritmo que pueda responder de manera adecuada a las grandes variaciones que presentan, pese a que su varianza no sobrepasa los 0,5 pixeles no tiene alguna respuesta aceptable en relación al algoritmo de referencia (Figura 62), con FREAK, se observó, que en el movimiento *Linear Progression*, tiene una respuesta y *Side winding*, tiene un comportamiento mayor al del algoritmo de referencia en cuanto a varianza se refiere (Figura 63), (LK varianza entre 0 y 0,5 pixeles, FREAK varianza entre 1,5 y 2 pixeles, en gait SW)

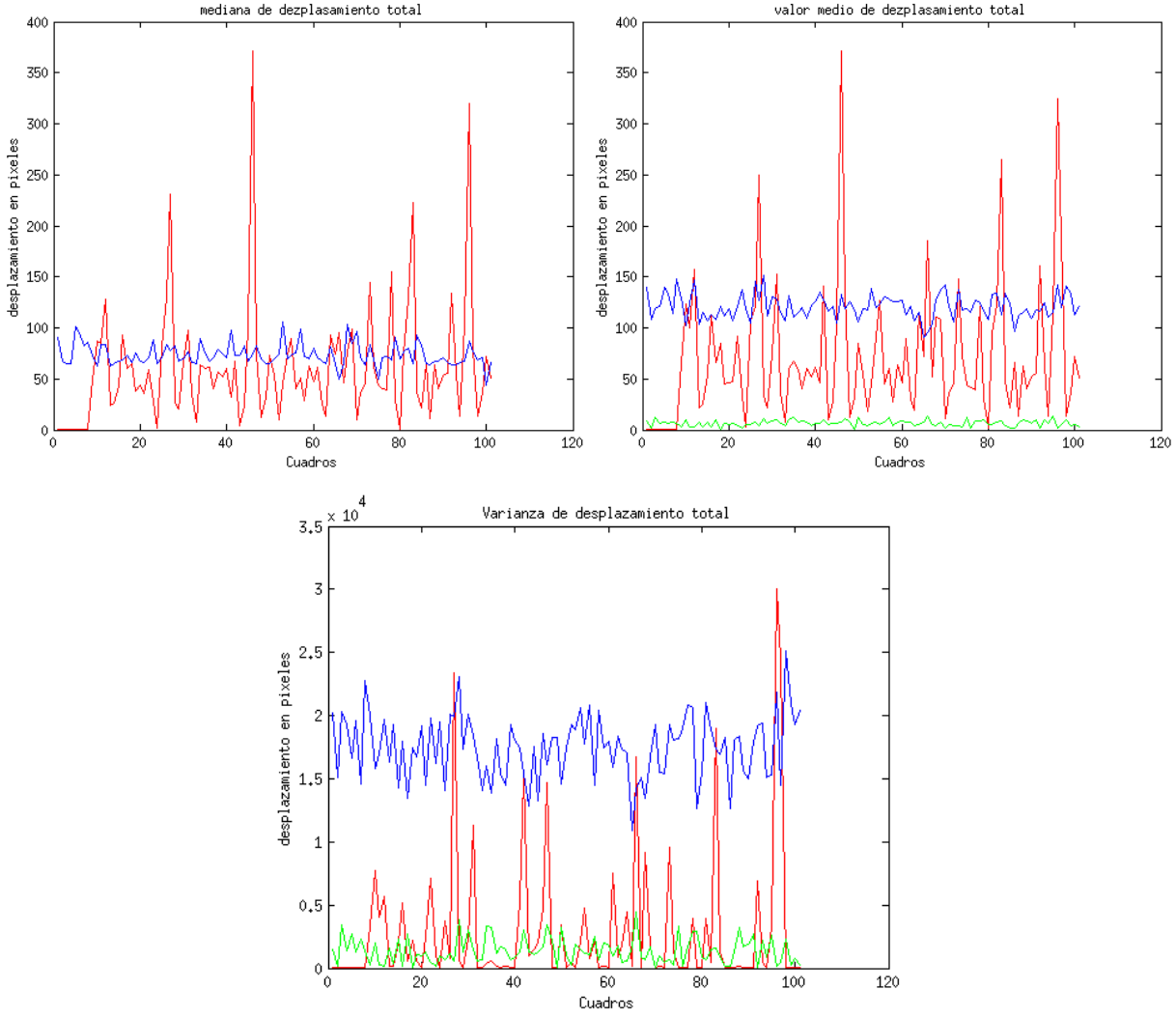


Figura 62. Gráficas de la mediana, el valor medio y la varianza, del video SWNC, procesado con los algoritmos.

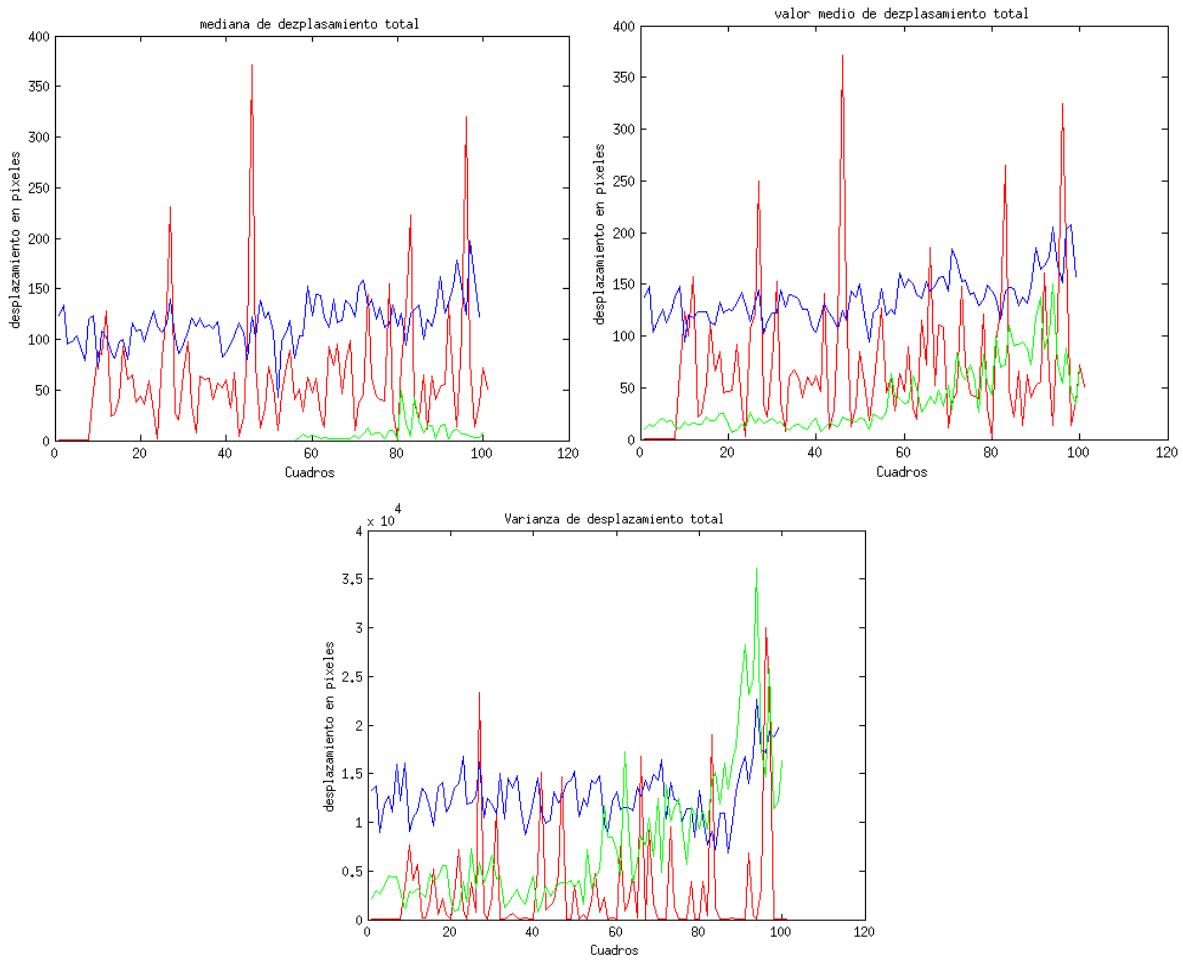


Figura 63. Gráficas de la mediana, el valor medio y la varianza, del video LPNC, procesado con los algoritmos.

5.3. Ubicación de la cámara

Un punto muy importante a tener en cuenta, para el trabajo con los descriptores mencionados, es la ubicación de la cámara en el robot, según el movimiento o desplazamiento que se desee.

Como se ve en el anexo 10, cuando la cámara está ubicada de manera frontal en el robot, la respuesta en el movimiento LP (Linear Progression), presenta menor varianza, que al colocarla al costado, esto se debe a que durante la captura del video, con la cámara puesta de forma frontal, se pierden menos objetos de vista, que con la cámara lateral, permitiendo que los algoritmos presenten resultados más próximos. De manera contraria sucede cuando la cámara está ubicada en el frente mientras la serpiente se mueve de forma SW (Side winding), se presenta menor variación en los datos, si está ubicada al costado.

Cuando la cámara se encuentra al costado, en los dos desplazamientos de la serpiente se observa una varianza considerablemente mayor, que en la ubicación frontal, incluyendo con el algoritmo de referencia de Lukas y Kanade, lo cual muestra que la mejor opción para la estimación del desplazamiento del robot utilizando los algoritmos propuestos es ubicar la cámara en el frente del robot. (Ver anexos 10 y 11)

6. CONCLUSIONES

Para concluir este trabajo de grado, es importante aclarar que las distancias encontradas son aproximaciones tomadas, de los datos generados por los algoritmos FREAK, BRISK y de flujo óptico piramidal de Lukas y kanade (LK), los archivos generados se pueden encontrar en los anexos correspondientes y contienen suficientes datos para hablar de una aproximación certera.

Para el desarrollo de este trabajo, se creó una base de datos de videos, tomados con la cámara embarcada en el robot, en dos ubicaciones (frontal y lateral), y con dos movimientos específicos de la serpiente, *(Linear Progression y Side Winding*), que pueden ser utilizados, para más desarrollos por el grupo de investigación SIRP, de la Pontificia Universidad Javeriana, y que además, permitieron comprobar que: la mejor ubicación para la estimación en pixeles, del desplazamiento del robot serpiente, es frontal, debido a que la imagen captada por la cámara, tiene mayor visión del escenario, pierde menos puntos clave con el movimiento del robot y por lo tanto so obtienen mejores resultados para cumplir con la tarea de estimación de distancia; además cuando está ubicada de forma lateral, la varianza entre puntos aumenta significativamente, incluso al procesar los videos con el algoritmo de flujo óptico piramidal de Lukas y Kanade, lo que se evidencia en el anexo 11.

Uno de los puntos más importantes a tener en cuenta, en un próximo trabajo relacionado con este, es que sin duda, se debe optimizar el algoritmo en cuanto a la detección de puntos, ya que por cada pareja de imágenes generada en los algoritmos de prueba, se repetía el paso completo de detección de puntos, aumentando el error en la medida y el tiempo de ejecución, el cual es mayor que el del algoritmo de Lukas y Kanade; se sugiere contemplar una forma de trabajar con los puntos en vecindades pequeñas por cambio de imágenes y no de lectura completa iniciando el proceso de ceros, como está planteado en este momento.

A pesar de que el tiempo de ejecución es considerablemente alto en comparación con el algoritmo de referencia, FREAK cumple con ocupar menos tiempo que BRISK, en el procesamiento de los videos, lo que es acorde con lo predicho inicialmente, además aunque pareciera diferir más que BRISK, en los resultados, pero al revisar la escala en que se encuentran las gráficas, se puede observar que no es considerablemente diferente y que por lo tanto puede servir como base para la estimación del desplazamiento de un robot tipo serpiente;

No se puede concluir de manera certera acerca de la robustez de los algoritmos, ya que se encontraron, diferencias considerables entre los resultados obtenidos por el algoritmo de flujo óptico piramidal de Lukas y Kanade y los resultados obtenidos por los algoritmos de prueba FREAK y BRISK.

Durante el desarrollo de este trabajo de grado, se evidenciaron cambios muy bruscos en el movimiento de la serpiente, que pueden afectar los resultados deseados, es por esto que se sugiere, para próximos trabajos en el área, colocar una protección a los servo motores, de espuma, que suavicen el movimiento del robot y lo protejan de los golpes naturales generados por los diferentes movimientos (gaits) que este puede tener. Por otra parte se recomienda, a nivel de software implementar un algoritmo “compensador” del movimiento que permita “suavizar” el video para lograr una estimación con menor varianza, o complementar el algoritmo sugerido, con los filtros necesarios para reducir los cambios bruscos y de esa manera obtener resultados más precisos, que permitan mediante la conversión que se sugiere en la Ecuación 10. Relación entre la distancia y pixeles., obtener una medida en unidades de longitud muy cercana a la realidad, que permita

continuar con los desarrollos y ampliar las aplicaciones propuestas, en el grupo SIRP, para el uso del robot tipo serpiente.

Trabajos futuros: Este trabajo de grado, plantea una solución para estimar el desplazamiento del robot tipo serpiente, utilizado por el grupo SIRP, de la Pontificia Universidad Javeriana, que puede ser utilizado, en el desarrollo de la visión del robot, que permita a este, un movimiento en exteriores, que contribuya en efecto, a uno de sus objetivos principales, como es el desplazamiento en terrenos hostiles, que permitan proteger al operario de las minas antipersonales. Por otra parte los resultados obtenidos, pueden ser utilizados, en futuras investigaciones, en el estudio de propuestas diferentes de algoritmos, al de flujo óptico piramidal de Lukas y Kanade. En un posterior trabajo se puede también, utilizar este trabajo como base, para la detección del desplazamiento del robot tipo serpiente en unidades de longitud (metros).

7. BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN

Flórez J., Calderón F., Parra C. (2012). Servo load analysis for the classification of surface. *2012 XVII SYMPOSIUM OF IMAGE, SIGNAL PROCESSING, AND ARTIFICIAL VISION (STSIVA)*.

A. Alahi, R. Ortiz, and P. Vandergheynst. (2012). FREAK: Fast Retina Keypoint. *IEEE Conference on Computer Vision and Pattern Recognition*.

Andrei Zaharescu, Edmond Boyer, Radu Horaud. (4 de mayo de 2012). Keypoints and Local Descriptors of Scalar Functions on 2D. *Springer Science+Business Media*.

Bradski Gary, Kaehler Adrian. (2008). Learning OpenCV. En K. A. Bradski Gary, *Learning OpenCV* (Vol. I, págs. 316- 341 cap 10). O'Reili Media, Inc.

Domínguez, J. M. (2009). "ESTIMACIÓN DE LA DISTANCIA RECORRIDA POR UN ROBOT MÓVIL MEDIANTE LA UTILIZACIÓN DE DESCRIPTORES SURF". Madrid. Recuperado el 2012

Edward Rosten, T. D. (mayo de 2006). Machine learning for high-speed corner detection. *European Conference on Computer Vision*.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski. (2011). ORB: an efficient alternative to SIFT or SURF. *ICCV* , 2564-2571.

F. Cortes, D. Linares, D. Patino, K. Melo. (2011). A Distributed Model Predictive Control (D-MPC). *IEEE*.

Gauglitz Steffen, Höllerer Tobias, Turk Matthew. (31 de March de 2011). Evaluation of Interest Point Detectors and Feature Descriptors. *Springer Science+Business Media*.

Henrik Aanæs, Anders Lindbjerg Dahl, Kim Steenstrup Pedersen. (Mayo de 2012). Interesting Interest Points. *International Journal of Computer Vision, 97*, 18-35.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. (2006). SURF: Speeded Up Robust Features. *Springer-Verlag Berlin Heidelberg* , 404–417.

Jae-Ho Yun, Rae-Hong Park. (2006). Self-Calibration with Two Views Using the Scale-Invariant Feature Transform. *Springer-Verlag Berlin Heidelberg* , 589 – 598.

Jianbo Shi, C. T. (1994). Good Features to Track. *IEEE conference on computer vision and pattern recogngtion (CVPR94) Seattle*.

Juan Pablo Ramirez Paredes, Raul E. SanchezYanez, Victor Ayala Ramirez. (6 de junio de 2011). A fuzzy inference approach to template-based visual tracking. *Springer-Verlag*.

Kalal Zdenek, Mikolajczyl Krystian, Matas Jiri. (July de 2012). *Tracking - Learning - Detection.* Recuperado el 16 de Feb de 2013, de IEEE Trnsactions on Pattern analysis on Machine Intelligence: http://www.computer.org/csdl/trans/tp/2012/07/ttp2012071409-abs.html

Kamilo Melo, Alexandra Velasco and Carlos Parra. (2011). Motion Analysis of an Ellipsoidal Kinematic Closed. *IEEE*.

Kamilo Melo, Laura Paez and Carlos Parra. (2012). Indoor and Outdoor Parametrized Gait execution with Modular Snake. *2012 IEEE International Conference on Robotics and Automation*.

Kamilo Melo, Laura Paez, Monica Hernandez, Alexandra Velasco, Francisco Calderon, Carlos Parra. (2011). Preliminary Studies on Modular Snake Robots. *IEEE*.

Lindeberg, T. (september de 2012). Scale Selection Properties of Generalized Scale-Space Interest Point Detectors. *Journal of Mathematical Imaging and Vision*.

Madjid Maidi, Jean-Yves Didier, Fakhreddine Ababsa, Malik Mallem. (1 de Octubre de 2008). A performance study for camera pose estimation using visual marker based tracking. *Springer- Verlag, 21*, 365-376.

Sajith Kecheril S., Arathi Issac, C. Shunmuga Velayutham. (2012). SaddleSURF: A Saddle Based Interest Point Detector. *Mathematical Modelling and Scientific Computation; Communications in Computer and Information Science, 283*, 413-420.

Stefan Leutenegger, Margarita Chli and Roland Y. Siegwart, Autonomous Systems Lab, ETH Z¨urich. (2011). BRISK: Binary Robust Invariant Scalable Keypoints. *IEEE International Conference on Computer Vision*.

Steffen Gauglitz, Tobias Höllerer, Matthew Turk. (7 de marzo de 2011). Evaluation of Interest Point Detectors and Feature Descriptors. *Springer Science+Business Media,*.

Tuan La Anh, Jae-Bok Song. (marzo de 2012). Robotic grasping based on efficient tracking and visual servoing using local feature descriptors. *International Journal of Precision Engineering and Manufacturing, 13*, 387-393.

Zhiyong Ye, Yijian Pe,i Jihong Shi. (2006). An Improved Algorithm for Harris Corner Detection. *Springer*.

8. ANEXOS

ANEXO 1: Algoritmo completo, utilizado en las pruebas (PDF)

ANEXO 2: Archivo con los fotogramas utilizados en la prueba de captura y de elección de algoritmos.

Anexo en CD, Ruta: /Pruebas tesis/Pruebas con fotogramas/

ANEXO 3: Prueba de video en un escenario controlado, del movimiento Linear Progression, con la cámara a frente.

Pruebas de videos de la serpiente en tres velocidades diferentes y resultados exportados por los algoritmos utilizados, con las características mencionadas. CD

ANEXO 4: Prueba de video en un escenario controlado, del movimiento Side Winding, con la cámara a frente.

Pruebas de videos de la serpiente en tres velocidades diferentes y resultados exportados por los algoritmos utilizados, con las características mencionadas. CD

ANEXO 5: Prueba de video en un escenario controlado, del movimiento Linear Progression, con la cámara lateral.

Pruebas de videos de la serpiente en tres velocidades diferentes y resultados exportados por los algoritmos utilizados, con las características mencionadas. CD

ANEXO 6: Prueba de video en un escenario controlado, del movimiento Side Winding, con la cámara lateral.

Pruebas de videos de la serpiente en tres velocidades diferentes y resultados exportados por los algoritmos utilizados, con las características mencionadas. CD

ANEXO 7: Prueba de video en un escenario sin control, del movimiento Linear Progression, con la cámara a frente.

Pruebas de videos de la serpiente en tres velocidades diferentes y resultados exportados por los algoritmos utilizados, con las características mencionadas. Videos pertenecientes a la base de datos, del grupo de investigación SIRP. CD

ANEXO 8: Prueba de video en un escenario sin control, del movimiento Side Winding, con la cámara a frente.

Pruebas de videos de la serpiente en tres velocidades diferentes y resultados exportados por los algoritmos utilizados, con las características mencionadas. Videos pertenecientes a la base de datos, del grupo de investigación SIRP. CD

ANEXO 9: Pruebas manuales

Pruebas realizadas para estimar el desplazamiento en pixeles del robot de forma manual, cada cuadro muestra el punto elegido para dicha tarea. CD

ANEXO 10: Graficas de valor medio, mediana y varianza para la estimación del desplazamiento en pixeles del robot serpiente generadas por el algoritmo de MatLab.

ANEXO 11: Gráficas comparativas del valor medio, la varianza y la mediana, de los videos tomados, con los algoritmos BRISK, FREAK y LK.

ANEXO 12: Algoritmo de Lukas y Kanade Utilizado para la realización de este trabajo de grado (PDF)