



UNIVERSIDAD CATÓLICA DEL NORTE
FACULTAD DE INGENIERÍA Y CIENCIAS GEOLÓGICAS
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Seguimiento de rostros mediante métodos directos utilizando imágenes RGB-D sintéticas.

Luis Mario Ramírez Solís
Profesor guía:
PhD. Juan Bekios Calfa

Antofagasta, Chile
Septiembre, 2017

ÍNDICE GENERAL

ÍNDICE DE FIGURAS	viii
NOMENCLATURA	ix
GLOSARIO	x
RESUMEN	xi
I. Introducción	1
1.1. Planteamiento del problema	1
1.2. Descripción formal del problema	2
1.3. Escenario de aplicación	2
1.4. Descripción de la solución propuesta	3
1.5. Motivación	4
1.6. Hipótesis	6
1.7. Objetivos	6
1.7.1. Principales contribuciones	6

1.8. Estructura de la tesis	7
II. Estado del arte	8
2.1. Introducción	8
2.1.1. Point tracking	9
2.1.2. Silhouette tracking	9
2.1.3. Template tracking	11
2.1.4. Aprendizaje	12
2.2. RGB-D tracking	13
III. Fundamentación Teórica	14
3.1. Introducción al tracking	14
3.2. Componentes de un seguidor de objetos	16
3.2.1. Representación del Objeto	16
3.2.1.1. Representación del objeto por su forma	16
3.2.1.2. Representación del objeto por su apariencia	18
3.2.2. Selección de atributos a seguir	19
3.2.3. Detección del objeto	20
3.2.4. Seguimiento del objeto	21
3.2.5. Modelo de Movimiento	23
3.2.6. Métricas de similitud	28

3.2.7. Nomenclatura matemática para el seguimiento de objetos	30
3.2.8. Gradiente	31
3.2.9. Hessian	32
3.2.10. Estrategia de búsqueda	32
3.2.10.1. Método de Newton	33
3.3. Algoritmo aditivo de Lucas-Kanade(LK)	36
3.4. Composicional Inverso (CI) de Baker-Matthews	39
3.5. Algoritmo de aprendizaje de Jurie-Dhome	41
3.5.1. Cálculo de Jacobiana por hiperplanos	42
3.5.2. Fase de entrenamiento	44
3.5.3. Fase de seguimiento	46
IV. Herramientas de implementación	48
4.1. Blender	48
4.1.1. Definición de Keypoints	50
4.2. Base de datos	51
V. Resultados experimentales	53
5.1. Imágenes 2D	54
5.1.1. Aditivo	55
5.1.2. Composicional inverso	56

5.1.3. Aprendizaje	59
5.1.4. Comparación imágenes de textura	61
5.2. Imágenes RGB-D	61
5.2.1. Aditivo	61
5.2.2. Composicional inverso	63
5.2.3. Aprendizaje	66
5.2.4. Comparación de imágenes de profundidad	66
VI. Conclusiones y trabajos futuros	70
6.1. Conclusiones	70
6.2. Trabajos futuros	72
Apéndice A. Expansión de Taylor del algoritmo de Lukas-Kanade	73
Apéndice B. Publicaciones	74
BIBLIOGRAFÍA	83

ÍNDICE DE FIGURAS

	Página
1.1. Escenario experimental	3
3.1. Estructura de un seguidor de objetos	15
3.2. Representaciones de un objeto	17
3.3. Clasificación de los métodos de seguimiento de objetos	22
3.4. Ejemplo de relación de puntos.	22
3.5. Transformación de traslación.	23
3.6. Transformación de escalado.	24
3.7. Transformación de rotacion.	24
3.8. Transformación euclidiana.	25
3.9. Transformación de similitud.	25
3.10. Transformación afín.	26
3.11. Transformación proyectiva.	27
3.12. Resumen de transformaciones geométricas	27

3.13. Tipos de transformaciones geométricas	28
3.14. Proceso descriptivo de un seguidor de objetos	31
3.15. Ajuste de una función cuadrática al valle de un punto mínimo de una función.	34
3.16. Resumen alto nivel del algoritmo Lukas-Kanade	37
3.17. Diagrama del proceso del algoritmo de Lucas-Kanade	40
3.18. Diagrama del proceso del composicional inverso	43
3.19. Proceso de perturbación del algoritmo de aprendizaje	45
3.20. Fase de entrenamiento Jurie-Dhome	46
3.21. Fase de seguimiento Jurie-Dhome	47
4.1. Camara utilizada para generar modelo 3D.	48
4.2. Modelo utilizado para la generación de imágenes sintéticas.	49
4.3. Nodos de composición	49
4.4. Gráfica de métricas	50
4.5. Método de definición de keypoints.	51
4.6. Representación del <i>ground truth</i>	52
4.7. Modelos que conforman la base de datos creada	52
5.1. Secuencia de imágenes de textura	54
5.2. Secuencia de imágenes de profundidad generadas	55
5.3. Gráfica de métricas del algoritmo aditivo	57
5.4. Gráfica de métricas del composicional inverso	58

5.5. Gráfica de métricas del algoritmo de aprendizaje	60
5.6. Gráfica comparativa de métricas imágenes de textura	62
5.7. Gráfica de métricas del algoritmo aditivo en imágenes de profundidad	64
5.8. Gráfica de métricas del composicional inverso en imágenes de profundidad	65
5.9. Gráfica de métricas del algoritmo de aprendizaje en imágenes de profundidad	67
5.10. Gráfica comparativa de métricas imágenes de profundidad	69

NOMENCLATURA

CMYK (Cyan, Magenta, Yellow and Key)

DTT (Distance Transform Templates)

RGB (Red, Green, Blue)

RGB-D (Red, Green, Blue - Depth)

SD (Steepest Descent)

TBD (Tracking By Detection)

GLOSARIO

FERNs: Estructuras no jerárquicas

Frame: Cuadro o imagen perteneciente a una secuencia de vídeo.

Greedy: Paradigma algorítmico que sigue la heurística de escoger siempre el mínimo local óptimo.

Kernel: Referente en este trabajo al área que se busca en la imagen.

Spline:

RESUMEN

Actualmente, el campo de detección de objetos en imágenes es un área con mucho desarrollo, debido a las diversas aplicaciones de uso industrial y actividades de la vida diaria. Por ejemplo, el área militar cuenta con un importante presupuesto de inversión en gran cantidad de países, en actividades tales como: la vigilancia y detección automática de objetivos; otros sectores de aplicación son la industria automotriz, cine, entretenimiento y salud, por ejemplo.

Este trabajo se centra en la implementación y comparación del rendimiento de los algoritmos de alineación rígida para el seguimiento de objetos con enfoque aditivo, composicional inverso y por aprendizaje, propuestos por Lucas-Kanade, Baker-Matthews y Jurie-Dhome, respectivamente. El objetivo de este trabajo es probar su desempeño en imágenes RGB-D sintéticas generadas de un modelo 3D, en este caso de la parte superior de una persona utilizando *Blender*. Para lograr la obtención de la profundidad en el sintetizado de imágenes en *Blender* se utilizaron nodos de composición. Para validar el correcto desempeño de los algoritmos antes de ejecutarlos en las imágenes de profundidad sintéticas, se utilizaron en una secuencia de imágenes 2D.

Se comprobó que a pesar de que el algoritmo propuesto por Lucas-Kanade es el más antiguo y lento sigue siendo el más confiable y preciso tanto en imágenes 2D como de profundidad. El algoritmo propuesto por Baker-Matthews comprobó ser indiscutiblemente superior en velocidad comparado con los otros dos algoritmos, y además presentó un buen desempeño en casi toda la secuencia de imágenes 2D, sin embargo, su desempeño en imágenes de profundidad fue malo. Por su parte, el algoritmo propuesto por Jurie-Dhome presentó buenos resultados en los primeros *frames* para después dejar de converger, esto se atribuye a un condicionamiento incorrecto de la matriz calculada. Por el lado de la eficiencia de tiempo, este algoritmo resultó ser el más caro debido al proceso de entrenamiento, sin embargo, una vez entrenado el tiempo de procesado de cada *frame* es casi instantáneo.

Estos resultados me llevan a estimar que con un correcto entrenamiento, el algoritmo de

Jurie-Dhome podría tener la velocidad del algoritmo propuesto por Baker-Matthews con una precisión cercana a la obtenida por la propuesta de Lucas-Kanade, así como también se sugieren otros métodos de entrenamiento para trabajos futuros.

Capítulo I

Introducción

1.1. Planteamiento del problema

El problema de seguimiento de objetos utilizando alineación rígida es un tema ampliamente estudiado, en el cual existe un gran interés y esfuerzo en generar mejores soluciones conforme van evolucionando los sensores de adquisición de imágenes.

El método consiste en definir una imagen *template* (plantilla) y realizar una búsqueda sobre una imagen objetivo, que comúnmente es un *frame* de un vídeo que va cambiando en el tiempo. Para realizar esta búsqueda es necesaria una estrategia que puede ser por fuerza bruta en toda la imagen o una búsqueda dirigida con algún método de optimización. Estos métodos de optimización funcionan basados en métricas de similitud o desimilitud, donde es necesario definir la estrategia del cálculo de los "pasos" dados en cada iteración del proceso de búsqueda.

La dificultad de este problema va en constante crecimiento, debido a las nuevas dimensiones disponibles que ofrecen los sensores de adquisición de imágenes, donde la mayoría de los trabajos se basaban en imágenes 2D. Sin embargo, en estos últimos años los esfuerzos están siendo enfocados en nuevas técnicas de detección y seguimiento de imágenes 3D. El presente trabajo de tesis explota esta nueva línea de trabajo y se centra en la transición entre estos enfoques utilizando imágenes de profundidad RGB-D sintéticas.

El algoritmo propuesto por Lucas-Kanade y Baker-Matthews cuentan con la capacidad para trabajar en imágenes de profundidad, sin embargo, para el algoritmo de Jurie-Dhome aún se está

buscando definir un método óptimo. La principal ventaja del algoritmo de Jurie-Dhome es que no es necesario hacer cálculos complejos para la estimación de la pose (matriz Hessiana). A diferencia de los métodos tradicionales, Jurie-Dhome utiliza una aproximación basada en aprendizaje que disminuye considerablemente los tiempos de cálculo y permite estimar de manera efectiva los cambios de pose del objeto.

1.2. Descripción formal del problema

El problema de alineación rígida puede ser expresado formalmente como:

$$\sum_x [I(w(x; p + \Delta p)) - T(x)]^2 \quad (\text{I.1})$$

Donde x representa las coordenadas del píxel en un vector $[x, y]$, I es una imagen que va cambiando conforme transcurre el tiempo, w es la función de transformación la cual nos permite convertir las coordenadas de la imagen I , mediante los parámetros de transformación p , en coordenadas de la imagen *template* T para posteriormente calcular la diferencia. Esto con la finalidad de minimizar la diferencia entre las imágenes, lo que se traduce como una mayor similitud.

Para lograr encontrar los parámetros que permitan minimizar la diferencia entre las imágenes es necesario tener una estrategia de búsqueda, la cual sea capaz de encontrar los parámetros Δp óptimos.

1.3. Escenario de aplicación

Debido a la naturaleza experimental de este trabajo, el escenario fue establecido en Blender como se puede apreciar en la Figura 1.1. Esto con la finalidad de probar el funcionamiento de los algoritmos mencionados anteriormente en imágenes RGB-D generadas sintéticamente.

Se eligió este *software* debido a su capacidad de generar imágenes RGB-D sintéticas y al preciso control de la cámara virtual que ofrece. La importancia del control de la cámara reside en que se pretende generar una secuencia con una rotación paulatina del modelo 3D.

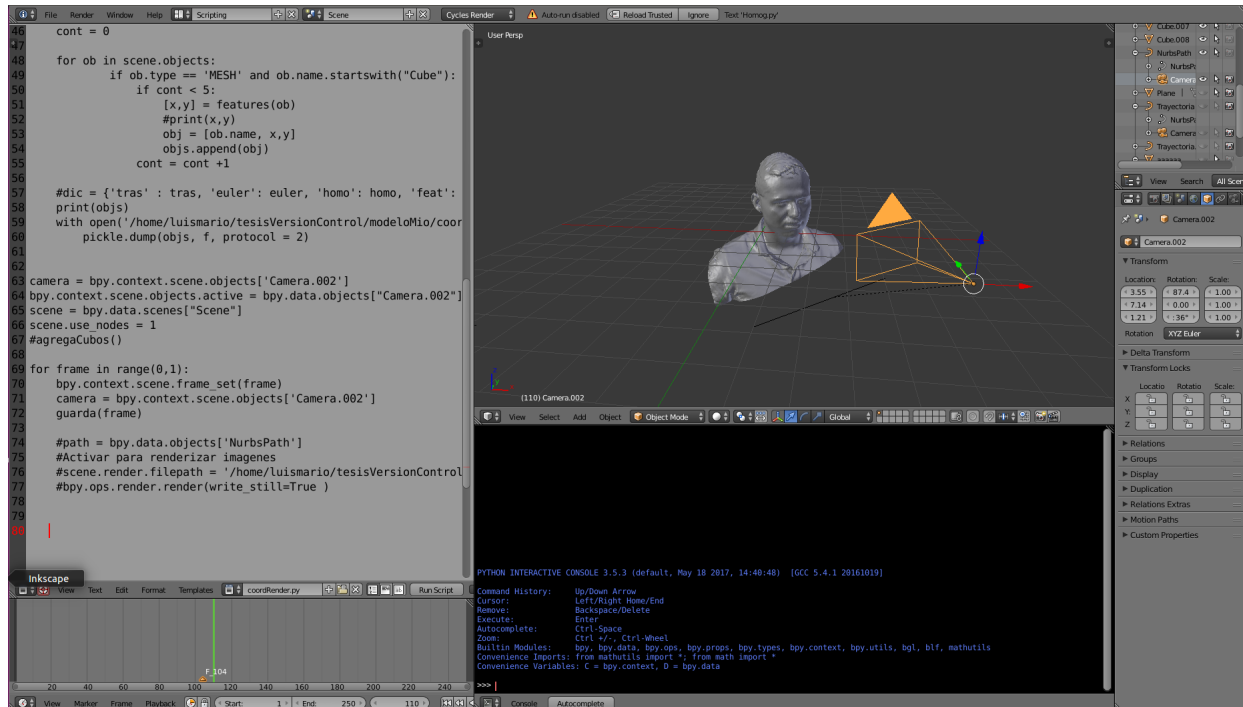


FIGURA 1.1: Escenario experimental

1.4. Descripción de la solución propuesta

El proceso de trabajo inicia con la creación de las imágenes de profundidad sintéticas utilizando los nodos de composición que permiten adquirir la información de profundidad al momento de renderizar la imagen en *Blender*, así como generar una trayectoria de cámara sin cambios bruscos debido a que la alineación rígida asume que los cambios entre los *frames* son pequeños. Posteriormente es necesario implementar el algoritmo propuesto por Lucas-kanade, Baker-Matthews y Jurie-Dhome. Para comprobar su correcto funcionamiento se realizarán pruebas en imágenes 2D. A continuación se procederá a aplicar estos algoritmos en las imágenes de profundidad sintéticas generadas y se comprobarán sus resultados. Esto proporcionará la información necesaria para evaluar y comparar los resultados, así como para proponer técnicas óptimas de entrenamiento del algoritmo de aprendizaje.

1.5. Motivación

Actualmente, el campo de detección de objetos en imágenes es un área con mucho desarrollo, debido a las diversas aplicaciones de uso industrial y actividades de la vida diaria. Por ejemplo, el área militar cuenta con un importante presupuesto de inversión en gran cantidad de países, en actividades tales como: la vigilancia y detección automática de objetivos; otros sectores de aplicación son la industria automotriz, cine, entretenimiento y salud, por ejemplo.

Es posible expandir la funcionalidad de la detección de objetos, si se agrega la capacidad de seguimiento de los mismos. El seguimiento de objetos ayuda a mejorar el rendimiento en aplicaciones de detección debido a que estos son mucho más eficientes en costo computacional y tiempo. Al combinar ambas técnicas se pueden obtener un conjunto nuevo de aplicaciones que no se podrían desarrollar de manera independiente.

Dentro de las técnicas que conforman el seguimiento de objetos, se encuentran los algoritmos de alineamiento rígido. Reciben este nombre debido a que se basan en alinear un *kernel* o *template* con otras imágenes, utilizando formas de figuras geométricas básicas para codificar información de apariencia en imágenes 2D y profundidad en imágenes 3D. Para lograr la alineación se realiza una búsqueda, puede ser por fuerza bruta en toda la imagen o utilizando una estrategia. En general, existen varias estrategias que permiten solucionar un problema de alineación rígida utilizando un *template*, en este trabajo se eligió el descenso por gradiente ya que es considerada la más utilizada. Parte de la efectividad de los algoritmos depende del modelo de movimiento, el cual define el tipo de transformación esperada del objeto en la imagen.

En relación al área de imágenes 2D el problema de alineación rígida está casi resuelto, sin embargo, éstos cuentan con las desventajas que proporcionan este tipo de imágenes como son la sensibilidad al cambio de color o iluminación. Por otro lado, las imágenes 3D logran invarianza a dichos problemas porque en lugar de utilizar las intensidades de la imagen usan la profundidad. Normalmente, los sensores láser capaces de generar nubes de puntos muy precisas son bastante caros, por este motivo se ha optado por utilizar un sensor RGB-D para generar un modelo 3D aproximado pero con un precio más accesible. Con esto se fomenta la continuidad de esta línea de investigación debido al uso de tecnologías económicas y accesibles para casi cualquier individuo. Para poder validar el desempeño de los algoritmos de alineación rígida en imágenes de profundidad sintéticas se utilizará el *software* Blender, el cual fue elegido debido a su capacidad de generar imágenes de cualquier ángulo deseado y obtener la información de profundidad de la cámara virtual con sus nodos de composición.

En este trabajo, se han estudiado y se han comparado los algoritmos propuestos por: Lucas-Kanade, Baker-Matthews y Jurie-Dhome. Cada uno con un enfoque diferente, los cuales efectúan el alineado de manera aditiva directa, composicional inversa y por aprendizaje, respectivamente. El algoritmo de Lucas-Kanade fue uno de los primeros en entregar buenos resultados, comúnmente es identificado como un algoritmo que tiene un buen desempeño, pero es muy costoso computacionalmente, por esta razón se le suele elegir como un método de comparación. En este caso, cuenta también con la ventaja de que existen versiones de este mismo que funcionan con imágenes de profundidad. El algoritmo propuesto por Baker-Matthews es una modificación del algoritmo de Lucas-Kanade, principalmente en la regla de actualización de los parámetros de movimiento y en los roles que juegan la imagen y el *template*, este algoritmo ha mostrado una mejora substancial en los tiempos de procesamiento en imágenes 2D y a la vez entregado una gran precisión.

El algoritmo propuesto por Jurie-Dhome se basa en un enfoque de aprendizaje el cual se comporta como un composicional inverso, debido a que primero es necesario un proceso de entrenamiento para generar la matriz *Hessiana*, para después realizar la actualización de los parámetros de movimiento mediante una composición de funciones en lugar de aditivamente como lo es en el enfoque propuesto por Lucas-Kanade.

La ventaja del composicional inverso se basa en el cálculo de la matriz Hessiana, ya que permite calcularla solo una vez durante la ejecución del algoritmo, debido a que esta es calculada en base a la imagen *template*. A pesar de que los algoritmos de composición inversa propuestos por Baker-Matthews y Jurie-Dhome tienen el mismo enfoque, tienen diferencias fundamentales. El algoritmo con un enfoque de aprendizaje, crea numéricamente la matriz Hessiana a partir de movimientos aleatorios donde se calculan las diferencias de intensidades.

Se considera que al utilizar diferencias de profundidades se generará una versión más eficiente y robusta del algoritmo y será posible el alineado de imágenes de profundidad.

Contar con versiones de profundidad de cada algoritmo, hará posible comparar los desempeños y verificar si el algoritmo propuesto por Jurie-Dhome es una opción viable para la alineación de imágenes de profundidad sintéticas con sensores de bajo coste. Algunas de las aplicaciones serían la identificación de objetos en entornos con poca luz, la navegación de robots utilizando *landmarks*, la detección más robusta de rostros, entre otros.

1.6. Hipótesis

Esta tesis intentará responder la siguiente pregunta:

- ¿Es posible que el algoritmo propuesto por Jurie-Dhome, entrenado con imágenes RGB-D sintéticas, pueda rastrear el movimiento de un rostro humano con imágenes de profundidad?

1.7. Objetivos

Objetivo general El objetivo principal de esta tesis es:

Comparar la eficiencia y el índice de rendimiento del algoritmo aditivo de Lucas-Kanade, el de composición inversa de Baker-Matthews y el método de aprendizaje propuesto por Jurie-Dhome para el seguimiento de rostros en imágenes RGB-D sintéticas.

Objetivo específicos

1. Analizar críticamente el estado del arte sobre los métodos de seguimiento de rostro utilizando imágenes RGB-D sintéticas.
2. Generar las imágenes de entrenamiento RGB-D sintéticas para entrenar los algoritmos de seguimiento.
3. Evaluar el índice de rendimiento del algoritmo propuesto por Lukas-kanade, Baker-Matthews y el de Jurie-Dhome en imágenes sintéticas RGB-D.
4. Realizar un análisis comparativo del algoritmo propuesto por Lukas-kanade, Baker-Matthews y el algoritmo de Jurie-Dhome en imágenes RGB-D sintéticas.
5. Difundir los resultados en medios científicos nacionales o internacionales.

1.7.1. Principales contribuciones

En este trabajo de tesis se hacen las siguientes contribuciones en el área de alineación rígida:

- Un método para realizar experimentos con algoritmos de alineación rígida en imágenes sintéticas de profundidad generadas en Blender.
- Un análisis comparativo en términos de costo y eficiencia computacional de los algoritmos estudiados en imágenes 2D y de profundidad.
- El código de implementación del algoritmo de Lucas-Kanade con imágenes de profundidad en Python.
- El código de implementación del algoritmo de Baker-Matthews con imágenes de profundidad en Python.
- El código de implementación del algoritmo de Jurie-Dhome con imágenes de profundidad en Python.

1.8. Estructura de la tesis

En el capítulo II de este trabajo de tesis se da un repaso de los trabajos existentes en el área del seguimiento de objetos en imágenes 2D y RGB-D obtenidas por cámaras y generadas sintéticamente.

La fundamentación teórica es abordada en el capítulo III, donde se proporciona una introducción a los temas fundamentales para comprender este trabajo, tales como: la representación del objeto, selección de atributos a seguir, detección y seguimiento del objeto, modelos de movimiento y elementos fundamentales para comprender la estrategia de búsqueda.

En el capítulo IV se detallan las herramientas utilizadas para la generación de las imágenes sintéticas, Blender y el proceso seguido para realizar el conjunto de entrenamiento, la definición de los *keypoints* en las imágenes generadas y se muestran los modelos que conforman la base de datos creada.

En el capítulo V se describen los resultados experimentales obtenidos, así como el ajuste de los parámetros efectuado a los algoritmos y se presentan las secuencias de imágenes generadas.

Finalmente, en el capítulo VI la discusión de los resultados y las conclusiones generadas son presentadas. Además, se encuentran líneas para posibles trabajos futuros que puedan continuar esta línea de investigación.

Capítulo II

Estado del arte

En esta sección se ha realizado una revisión del origen hasta el estado actual de los enfoques de *tracking*, enfatizando el *kernel tracking* y el *tracking* en imágenes RGB-D.

2.1. Introducción

En relación al campo del *tracking* de objetos, existen dos formas de realizar el seguimiento. La primera consiste en realizar el proceso de detección en cada *frame*, lo cual es computacionalmente muy caro ya que normalmente se tiene que realizar un proceso de búsqueda en toda la imagen. La segunda consiste en detectar el objeto a seguir en la imagen y actualizar iterativamente el cálculo de la ubicación del área del objeto teniendo en cuenta su desplazamiento entre *frames*, debido a que se realiza el seguimiento tomando en cuenta la posición previa del algoritmo este enfoque es más efectivo computacionalmente. Cualquiera que sea el método que se elija, esta tarea se realiza utilizando la forma o la apariencia del objeto a seguir.

Existen varios métodos para realizar el *tracking* entre ellos se encuentran el *point tracking*, *silhouette tracking* y *kernel tracking*.

2.1.1. Point tracking

El enfoque de *point tracking* se centra en la correspondencia de puntos entre el objeto detectado en la imagen y el objeto a buscar, este se divide en dos categorías: la determinista y por métodos estadísticos.

Método determinista En 1987 Sethi y Jain [1] resolvieron la correspondencia por un enfoque *greedy* basados en los límites de proximidad y rigidez, sin embargo, su propuesta no toleraba las oclusiones, entradas o salidas de objetos. En 1990 Salari y Sethi [2] manejaron estos problemas al primero establecer la correspondencia de los puntos detectados y después extender el seguimiento de los objetos perdidos agregando un número hipotético de puntos. En 1991 Rangarajan y Shah [3] propusieron un enfoque *greedy*, que era restringido por la uniformidad próxima. En 1997 Intille y otros [4] propusieron una versión modificada del trabajo de [3] para encontrar correspondencias entre los centroides de los objetos, que son detectados utilizando sustracción de fondo.

En 2001 Veenman y otros [5] extendieron el trabajo de [1] y [3], introduciendo la restricción *common motion* para la correspondencia. En 2003 Shafique y Shah [6] propusieron un enfoque *multiframe* para conservar la coherencia temporal de la velocidad y la posición.

Método estadístico Este método a diferencia del anterior, tiene un enfoque estadístico que permite realizar tareas que no son fácilmente modelables y han sido usados extensamente para seguir contornos por Isard y Blake [7], reconocimiento de actividad por Vaswani y otros [8], identificación de objetos por Zhou y otros [9], y estructuración desde movimiento por Matthies [10].

2.1.2. Silhouette tracking

El enfoque de *silhouette tracking* se centra en el seguimiento de objetos difíciles de modelar con figuras geométricas simples, como lo pueden ser partes del cuerpo, como las manos de una persona, o algún otro objeto con varios grados de libertad de movimiento. El objetivo de un *silhouette tracker* es encontrar la región del objeto que corresponda con el modelo generado en *frames* anteriores, este modelo puede ser un histograma de color, las esquinas de un objeto o su contorno. Este método se divide en dos enfoques: la correspondencia por forma y la correspondencia por contorno. El seguimiento por forma busca la silueta del objeto en el *frame* actual. El seguimiento por contorno,

consiste en la evolución de un modelo a través de los *frames* con medidas de modelos de estado espacial o la minimización directa de alguna función de energía.

Seguimiento por forma Un trabajo relevante fue presentado en 1993 cuando Huttenlocher y otros [11] realizaron correspondencia de forma usando una representación basada en bordes. Utilizaron la distancia *Hausdorff*, la cual es una métrica matemática para comparar dos conjuntos de puntos en términos de los miembros menos parecidos. En 2001 Li y otros [12] propusieron utilizar la distancia *Hausdorff* para la verificación de trayectorias y estimación de pose. Posteriormente en 2004 Kang y otros [13] utilizaron histogramas de color y bordes como modelos de objetos. En contraste con los histogramas tradicionales, ellos propusieron generar histogramas de círculos concéntricos con radios centrados en un conjunto de puntos de control. En este mismo año Sato y Aggarwal [14] propusieron generar pistas de objetos aplicando la transformación de Hough en el espacio de velocidad a las siluetas del objeto en *frames* consecutivos.

Seguimiento por contorno El seguimiento utilizando modelos de estado espacial se remonta a 1992 donde Terzopoulos y Szeliski [15] definieron el estado del objeto por la dinámica de los puntos de control. Posteriormente en 1998 Isard y Blake [7] definieron el estado del objeto en términos de la forma curva del spline y los parámetros de movimiento afín. En el año 2000, MacCormick y Blake [16] extendieron el seguidor de objetos basado en filtros de partículas propuesto por Isard y Blake en 1998 [7] para seguir múltiples objetos incluyendo el *exclusion principle* para manejar las oclusiones. Después, Chen y otros [17] propusieron en 2001 un seguidor de contornos donde el contorno es parametrizado como si fuera un elipse.

El seguimiento por minimización directa de una función de energía del contorno tiene un funcionamiento parecido al de los algoritmos de segmentación, ya que los dos utilizan métodos *greedy* y descenso de gradiente para lograr la convergencia. La energía del contorno es definida en términos de información temporal en la forma del gradiente temporal (flujo óptico) Bertalmio y otros [18], Mansouri [19], Cremers y Schnorr [20] o estadísticas de apariencia generadas desde regiones del objeto y el fondo Yilmaz y otros [21], Ronfard [22].

Este trabajo se centra en la implementación de algoritmos basados en el *kernel tracking*, también identificado como *template tracking*, por lo que se hará una revisión más profunda de este tema.

2.1.3. Template tracking

En el campo de visión por computador, el *template tracking* consiste en identificar la posición de una imagen plantilla que, desde ahora en adelante, se hará referencia como *template*: patrón o conjunto de características que se quieren para reconocer en una imagen.

El *template* es identificado en una imagen inicial y esto permite seguir sus movimientos a través del tiempo. Se puede ver en [23] que existen 2 métodos para realizar esta tarea: el *feature-based* y el *template-based*. Existen otras maneras de representar el objeto en lugar de un *template*, como pueden ser histogramas de color [24] o mezclas de modelos pueden ser calculados usando la apariencia de los píxeles dentro de una región geométrica básica [25]. Sin embargo estos enfoques no serán abordados en este trabajo debido a que no son relevantes dentro del mismo.

El método *template-based* trata la imagen *template* como un todo. La ventaja es que puede reconocer patrones más complejos y difíciles de modelar que si solo se utilizarán las características locales. Este método es bastante robusto y ampliamente utilizado. Por otro lado, el método *feature-based* compara algunas características del *template*, como pueden ser puntos marcados manualmente, líneas o bordes. El beneficio de este método en comparación con el *template-based*, es que presenta una mayor robustez a las oclusiones parciales, ya que aunque algunas características estén ocluidas, con las que permanecen visibles se puede estimar la pose del *template*.

El trabajo de Lukas y Kanade [26] propuesto en 1981, donde exponen una técnica de registrado de imagen que utiliza la intensidad espacial del gradiente de la imagen y que presenta un buen desempeño comparado con el estado del arte de esa época, se considera fundamental en el área. Gracias a este, han ido apareciendo un gran número de enfoques para el *template tracking*. Como podemos ver en [27] existen 3 enfoques que se pueden utilizar en el *template tracking*: *tracking-by-detection*, *energy minimization* y *aprendizaje*.

El enfoque de *tracking-by-detection* cuenta con varios años de investigación, algunos de los trabajos recientes relacionados fue propuesto en 2007 por Özuysal y otros [28] donde presentó un enfoque basado en TBD llamado **FERNs** donde se extraen los puntos de interés de una imagen y se relacionan usando un enfoque de clasificación al estimar la probabilidad de en cual clase pertenece cada punto. En 2009 Holzer y otros [29] presentaron un enfoque *Distance Transform Templates*, que se construye al encontrar contornos cerrados y los relaciona usando una técnica similar a la relación de puntos utilizada en [28]. Del 2008 al 2010 se han hecho grandes avances en esta área por parte de Hinterstoisser y otros, sus primeros trabajos llamados Leopard [30] y Gepard [31]

hacen uso de parches de la imagen que rodean un punto de interés. Estos parches (*templates*) son utilizados posteriormente para estimar la pose y encontrar relaciones. En su trabajo de Dominant Orientation Templates (DOT) [32], intenta superar la dependencia de la detección de puntos. Se propone utilizar un enfoque basado en *template matching* donde *templates* sean aprendidos por cada pose.

Este enfoque consiste en la identificación del *template* en toda la imagen por cada *frame*, independiente de la posición anterior de este mismo. Esto se traduce en tiempos muy grandes de entrenamiento y un desempeño *frame a frame* poco viable.

Otra propuesta es el *tracking* por *energy minimization* donde numerosos trabajos han continuado el trabajo propuesto por Lukas-Kanade [26] en 1981, algunos han propuesto diferentes órdenes de aproximación a las funciones de error como Malis [33] utilizando técnicas de minimización de segundo orden, o Benhimane y otros [34] utilizando homografías 2D para el tracking. El problema del cambio de iluminación y la oclusión, también ha sido atacado por Hager y otros en 1998 [35]. El principal cambio en este enfoque han sido las propuestas en las reglas de las funciones de transformación y pueden ser clasificadas en 4 tipos, el enfoque aditivo propuesto por Lukas-Kanade en 1981 [26]. El composicional, propuesto por Shum y Zseliski en el año 2000 [36]. En 1998 fue propuesto el enfoque aditivo inverso por Hager y Belhumeur [35], para después ser retomado por Cascia y otros [37] en el año 2000. Y el último propuesto fue el composicional inverso por Baker-Matthews [38] en 2001.

2.1.4. Aprendizaje

En 2002 Jurie y Dhome [23] propusieron un algoritmo que utiliza hiperplanos en vez de una aproximación Jacobiana. Esto permitió un mejor desempeño basado en una técnica de entrenamiento con matrices homogéneas aleatorias. Una ventaja de este método, es que la matriz generada con los hiperplanos solo se calcula al principio del algoritmo y posteriormente se actualizan los parámetros de movimiento. Un poco después, estos mismos autores generaron una propuesta robusta al problema de las oclusiones [39].

A pesar de tener un buen desempeño respecto a lo que tiempo se refiere y presentar robustez a las oclusiones, aún carecía de invariancia a la iluminación, una propiedad que es muy importante en el campo de visión por computador. Graßl y otros [40] se percataron de esto y en el año 2003 presentaron una versión mejorada con invariancia a la iluminación.

Los últimos 2 enfoques cuentan con la característica de poder realizar un análisis *frame a frame*. La diferencia entre ambos enfoques es que el de *energy minimization* proporciona una mayor flexibilidad cuando se está ejecutando y el enfoque de aprendizaje permite una mayor velocidad de rastreo.

2.2. RGB-D tracking

El uso de imágenes de profundidad sintéticas para entrenar un seguidor de objetos ya ha comprobado su eficiencia en 2013 [41], donde utilizan imágenes sintéticas renderizadas de una nube de puntos de una mano, para entrenar un modelo de regresión *Hough forest* en 2011[42]. Demostrando que las imágenes sintéticas de profundidad son un modelo ideal de un objeto y que es necesaria la adición de ruido para generar entrenamientos útiles para su uso en entornos reales. En [43] 2013 se demuestra (e implementa) como a partir de un *data-set* descargado se generaron 5000 imágenes sintéticas para entrenar bosques de decisión paralelizables invariantes a la pose, la forma del cuerpo y oclusiones. Además de presentar tiempos de ejecución "*super-realtime*" en hardware comercial.

En el trabajo de Choi y Christensen en 2013 [44] se utilizan las imágenes RGB-D sintéticas y reales para evaluar el desempeño de un filtro de partículas para obtener un parámetro de medida más preciso y compararlo con otros algoritmos.

Capítulo III

Fundamentación Teórica

3.1. Introducción al tracking

El objetivo del tracking en imágenes es detectar y seguir el movimiento de los objetos que sean de interés. Estas detecciones deberán ser preferentemente cuadro a cuadro para brindar una mayor precisión. Entre mayor sea la precisión, mejor será el análisis de las trayectorias de los objetos y se podrá entender con más detalle su comportamiento logrando distinguir algún objeto específico entre un conjunto de objetos interesantes.

Los seguidores de objetos constan de 4 partes:

Definir una representación adecuada del objeto En este punto existen dos enfoques, la representación por forma y por apariencia. La decisión de que representación usar recaerá en el tipo de objeto que se quiere detectar, el poder de computación disponible y si será *on line* o no.

Definir las características de la imagen usadas como entrada para el seguidor de objetos

Dependiendo del método de representación seleccionado, se tendrá que tener en cuenta que información se puede obtener de las características que se podrán extraer. Para así evaluar cuales serán los atributos que aporten un mayor grado de robustez al seguimiento.

Detectar el objeto en la escena Todo seguidor de objetos necesita que se detecte el objeto a seguir ya sea en cada *frame* o solo en el primero. Dependiendo de las elecciones en los dos puntos anteriores, se evaluará cuando es necesaria la detección. Algunas técnicas de detección son:

detectores de puntos [45] [46] [47] [48], segmentación [49][50][51] [52][53], extracción de fondo [54] y clasificadores supervisados [55][56][57][58] [59].

Seleccionar un método de seguimiento En esta etapa se selecciona algún método de las 3 categorías que abarcan el *tracking* de objetos, las cuales son: *Point tracking* [2][5][60], *Kernel Tracking* [24][47][61] y *Silhouette tracking* [7][18][14]. Al momento de seleccionar algún enfoque de estas categorías, se tendrá que tener en cuenta la información disponible que entregan las características de la imagen elegida, así como también el enfoque de representación.

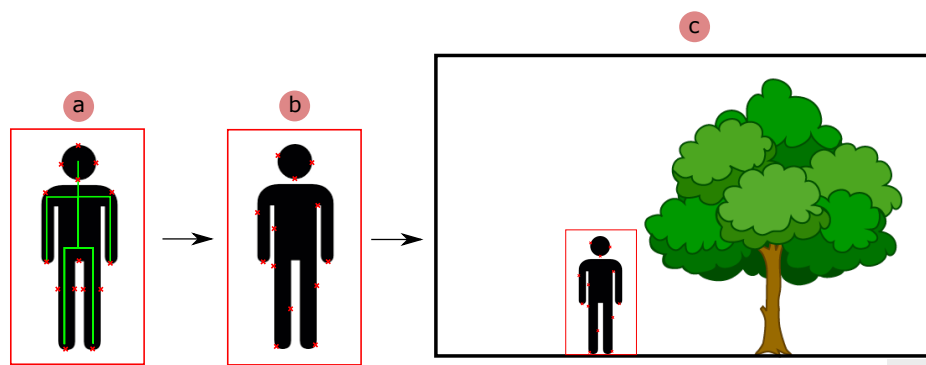


FIGURA 3.1: Estructura de un seguidor de objetos. (a) Se elige un tipo de representación del objeto, en este caso como ejemplo se seleccionó el *template*, modelo esquelético y los puntos. (b) Para la descripción del objeto se muestra como ejemplos el *template* y los puntos. En (c) se muestra como se puede detectar un objeto con un detector de puntos y seguir un objeto con técnicas de *point tracking* o *krenel tracking*.

Como se puede ver en la Figura 3.1, existen varias formas para detectar un objeto en la escena. Este trabajo se enfoca en la alineación rígida de imágenes, la cual es una técnica que se encarga de disminuir la distancia entre la imagen objetivo (*template*) y una imagen. Para medir esta distancia existen varias métricas, algunas de ellas son *sum of squared differences (SSD)*, *sum of absolute differences (SAD)*, *Normalized cross-correlation (NCC)*, *Mutual information (MI)* (ver sección 3.2.6). Es necesario contar con una manera de dirigir la búsqueda, de tal forma que minimice la diferencia o maximice la similitud entre las imágenes dependiendo de cual sea la métrica que estamos utilizando. El algoritmo más comúnmente utilizado para este problema es el descenso de gradiente, sin embargo existen otras propuestas como lo son la descomposición de diferencias [62] o la regresión lineal [63].

El descenso de gradiente puede ser realizado de diferentes maneras. Es necesario estimar el tipo de aproximación en cada paso del descenso de gradiente, algunas de las opciones son Gauss-Newton, Newton, *steepest-descent* o Levenberg-Marquardt.

3.2. Componentes de un seguidor de objetos

Dentro de esta sección se definirá la representación de un objeto para detección y seguimiento en sus dos enfoques: forma y apariencia.

Posteriormente se continuará con la revisión del proceso y consideraciones a la hora de la selección de atributos a seguir, la detección misma del objeto y las técnicas de seguimiento actuales.

Se aborda el tema del modelo de movimiento, los grados de libertad que ofrece cada transformación, así como las ventajas que ofrecen cada una.

Las métricas de similitud son descritas, así como también se da una descripción de la notación matemática utilizada y se hace un repaso de temas de gradiente y la matriz Hessiana.

Por último tenemos la estrategia de búsqueda utilizada para solucionar el problema de optimización que abarca esta tesis.

3.2.1. Representación del Objeto

Un paso fundamental a la hora de construir un seguidor de objetos es seleccionar las mejores características que describan el objeto a seguir. Para esto existen dos enfoques: la representación por **forma** y **apariencia**, tal como se pudo observar en la Figura 3.2.

3.2.1.1. Representación del objeto por su forma

La representación por forma es muy utilizada en el seguimiento de objetos y por eso es la primera en ser abordada. Dentro de este enfoque se destacan:

Puntos Esta se caracteriza por representar al objeto con un punto, en el centro del objeto. Esto se puede apreciar en la Figura 3.2 (a), así también se puede representar el objeto con puntos en ciertas zonas de interés como podemos ver en (b). Esta representación es utilizada principalmente cuando el objeto a representar es pequeño.

Figuras geométricas primitivas El objeto o alguna parte de el, es representado por figuras

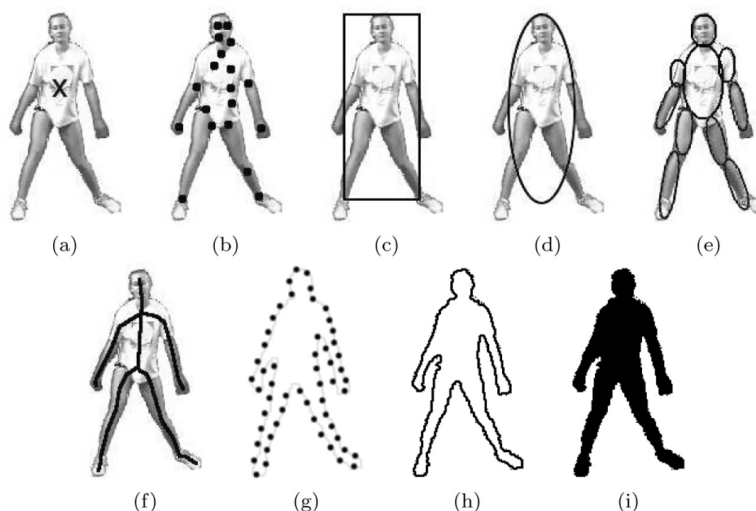


FIGURA 3.2: Representaciones de un objeto. (a) Centroide, (b) puntos múltiples , (c) parche rectangular, (d) parche elíptico, (e) parche múltiple basado en partes, (f) esqueleto del objeto, (g) contorno completo del objeto, (h) puntos de control en el contorno del objeto, (i) silueta del objeto. Yilmaz y otros [64]

geométricas primitivas como un rectángulo o un elipse (ver Figura 3.2 (a) y 3.2 (d)). El objetivo de utilizar estas figuras, es aplicar los modelos de movimiento definidos en la sección 4.2.7 como traslación, afín o proyectiva. Normalmente las transformaciones geométricas primitivas sirven para representar objetos rígidos simples, sin embargo también son utilizados para seguir objetos no rígidos.

Silueta y contorno del objeto El contorno define los límites de un objeto (ver Figura 3.2 (g) , 3.2 (h)). La silueta representa la parte que se encuentra dentro de los límites definidos por el contorno del objeto, como se puede ver en la Figura 3.2 (i). Esta representación es utilizada para seguir formas complejas no rígidas.

Modelos articulados del objeto Esta es especialmente útil en la representación de objetos con varios grados de libertad de movimiento, como puede ser el cuerpo humano. Los componentes del cuerpo, como la cabeza, torso, piernas, entre otros, son representados por elipses o cilindros como podemos ver en la Figura 3.2 (e). Para representar la relación entre las diferentes partes del objeto, es necesario un modelo de movimiento cinemático, por ejemplo el *joint angle*.

Modelos esqueléticos Esta es utilizada comúnmente como representación de forma en objetos detectados, y puede ser utilizada para modelar objetos articulados y rígidos, ver Figura 3.2 (f). Para extraer el esqueleto del objeto es necesario aplicar la transformación de eje media [65].

Cabe resaltar que la representación por forma también puede mezclarse con la representación por apariencia.

3.2.1.2. Representación del objeto por su apariencia

Otro enfoque a utilizar en la descripción de un objeto en una imagen es por la apariencia. Dentro de esta categoría, se encuentran:

Densidad de probabilidades de la apariencia del objeto Esta puede ser calculada de regiones especificadas en la imagen por los modelos de forma (región interior de un elipse o el contorno). La densidad de probabilidad de la apariencia del objeto puede ser paramétrica como el método Gaussiano [66] o una mezcla de Gaussianas [67], así como también puede ser no paramétrica, como son las ventanas Parzen [68] e histogramas [24].

Templates Los *templates* están formados por figuras geométricas simples o siluetas [25], y tienen la ventaja de que contienen información de apariencia y espacial. Sin embargo, esta técnica cuenta con la desventaja de solo contar con una perspectiva del objeto, por lo que solo pueden ser utilizados en ocasiones donde el objeto no varíe mucho durante el período de seguimiento.

Modelos Activos de Apariencia Los modelos activos de apariencia son generados al modelar simultáneamente la forma del objeto y su apariencia [69]. Esta técnica consiste en definir el borde del objeto con puntos de interés, estos incluso pueden estar dentro del área del objeto. Cada punto de interés es almacenado en un vector con las características del color, textura o la magnitud del gradiente. Para realizar este tipo de representación es necesario realizar una etapa de entrenamiento, donde la forma y su apariencia relacionada es aprendida de un conjunto de datos, por ejemplo, el análisis de componentes principales(PCA)[70].

Modelos de apariencia Multi-vista A diferencia de los *templates*, este método de representación contiene varios ángulos del objeto a seguir. Una manera de representar este conjunto de diferentes ángulos, es generar sub espacios a partir de los perfiles entregados. Algunas técnicas para generar sub espacios utilizadas en este método son: el PCA [70] y el análisis de componentes independientes (ICA) [71], estas han sido utilizadas para la representación de forma [72] y apariencia [73].

3.2.2. Selección de atributos a seguir

Uno de los pasos más importantes en la estructura de un seguidor, es la decisión de qué atributos son los más adecuados en el contexto del problema a solucionar. Algunas de las posibles opciones son:

Color Dentro del campo de color, existen varias representaciones de color como RGB, CMYK, entre otras. Cada uno tiene sus ventajas y desventajas, por lo que si se elige el color como el atributo de seguimiento de un objeto, se tendrán que tener en cuenta las dos propiedades físicas generales de este mismo: con qué tipo de luz se ilumina el objeto y las propiedades reflectantes de la superficie del objeto. Al utilizar este tipo de descriptor es muy importante tener en cuenta que si las condiciones de iluminación cambian, el espectro de color a seguir también cambiará.

Esquinas Este enfoque es uno de los más utilizados y existen varias opciones a considerar, entre las más importantes se encuentran Canny edge detector [74] y Harris [46]. Cada uno tiene diferentes ventajas y desventajas, sin embargo todos concuerdan en que son sensibles a las imágenes con ruido, pero que son menos sensibles a las variaciones de luz respecto al color debido a que los límites del objeto generan un cambio notable con el cálculo del gradiente o las intensidades de la imagen.

Flujo óptico Este enfoque consiste en calcular la diferencia de intensidad de un template en una imagen a través del tiempo para calcular el vector de desplazamiento de los píxeles que la conforman. Para lograr esto existen varios enfoques ya expuestos anteriormente: el **aditivo** [26] propuesto por Lukas-Kanade; el **aditivo inverso** [35] por Hager-Belhumeur y el **composicional** [36], de Shum y Szeliski, que incorporó mejoras en el rendimiento del algoritmo. Adicionalmente el **composicional inverso** [38] presentado por Baker-Matthews ofrece una mejora eficiente similar a la propuesta por Hager-Belhumeur. [35]. Este enfoque es muy utilizado en el seguimiento de objetos y segmentación basada en movimiento.

Textura Este enfoque a diferencia del color, requiere un paso extra, el cual genera los descriptores con base en la consistencia y rugosidad de la superficie del objeto. Dentro de estos métodos existe el *local binary patterns* (LBP) [75], *Histogram of Oriented Gradients* (HOG) [76], wavelets [77], entre otros. Este método también presenta una robustez a los cambios de luz a comparación del enfoque de color.

Como se puede observar, cada método propuesto funciona en diferentes condiciones. Es necesario

considerar cuál descriptor se usará, ya que es parte fundamental del desempeño de la solución desarrollada.

La selección del atributo posee un gran impacto en la solución desarrollada. Muchas veces, no se cuenta con la experiencia o conocimientos suficientes para seleccionar el atributo óptimo para un problema determinado, por este motivo, se realizan investigaciones para atacar el problema con diversos métodos de selección automática de atributos, como se puede ver en [78].

3.2.3. Detección del objeto

Una vez que se tiene decidido que atributos describen a nuestro objeto a seguir, se puede proceder a detectarlo. Para esto existen varias opciones dentro de las que se encuentran:

Detectores de puntos De la misma manera que los detectores de esquinas, los detectores de puntos son usados para encontrar características interesantes en una imagen. En algunos de estos detectores, existe la posibilidad de refinar la búsqueda de características interesantes a ciertos parámetros o umbrales, algunos ejemplos de estos detectores son SIFT [45], Harris [46], KLT [47] y Moravec [48]. Como en las demás secciones, cada uno tiene ventajas y desventajas dando como resultado tener que valorar que es más importante: la rapidez o la robustez.

Background subtraction Esta técnica consiste en generar un modelo del fondo de la escena, con el objetivo de calcular variaciones en esta misma, para detectar objetos en movimiento. Normalmente para detectar los objetos se utiliza un algoritmo de componentes conexas. Este enfoque ganó popularidad en el año 1997 con el trabajo de Wren et al [54], con el que se calculaban pequeños cambios a través del tiempo.

Segmentación La finalidad de estos algoritmos es separar los objetos que sean parecidos en la imagen. Para lograr esto, se consideran dos factores: el primero es el criterio a seguir para separar los grupos de objetos similares y el segundo es encontrar el mejor método para hacerlo. En esta categoría existen varios métodos dentro de los que se encuentran: Mean-shift Clustering [49], Graph-Cuts [50][51] y Active Contours [52][53].

Aprendizaje supervisado Esta técnica consiste en utilizar un algoritmo que es entrenado a partir del mapeo de ciertas características de la imagen con un valor pre-establecido, el cual puede ser continuo o una clase.

Es muy importante considerar qué características de la imagen se van a utilizar, ya que de ellas depende el desempeño y eficiencia del algoritmo. Algunos ejemplos de los algoritmos usados en este enfoque son: redes neuronales[55][56], adaptive boosting [57], arboles de decisión [58], y máquinas de vectores de soporte [59].

Dentro del área del aprendizaje supervisado algunas de las técnicas más utilizadas son:

Boosting La técnica de Boosting consiste en generar iterativamente un clasificador preciso, con base en un gran numero de clasificadores simples y con un porcentaje de acierto despreciable. Para generar estos clasificadores simples se pueden utilizar umbrales o simples perceptrones, donde en cada iteración se asignara un peso de votación a cada clasificador. Estos pesos serán asignados de acuerdo al mayor porcentaje de acierto que presente cada clasificador, esta técnica permitirá organizar los clasificadores del de mayor porcentaje de acierto hasta el menor.

SVM Las máquinas de soporte vectorial o *Support Vector Machine*, son clasificadores encargados de agrupar datos de manera separada de dos diferentes clases. Para realizar esta tarea, este clasificador se encarga de encontrar el mayor espacio marginal entre clases y los separa con un hiperplano. En caso de que el problema a resolver no sea lineal, es necesario aplicar un kernel dependiendo del tipo de problema para linealizarlo.

3.2.4. Seguimiento del objeto

Ahora que se conocen varias formas de describir un objeto y varios enfoques de detección, se puede proceder a la etapa de seguimiento. El objetivo de un seguidor de objetos es generar una trayectoria basada en las detecciones en cada imagen a través del tiempo. Existen varios métodos para hacer esto como se puede observar en la Figura 3.3.

Tal como su nombre lo indica, en el enfoque de **point tracking** los objetos detectados en imágenes consecutivas se representan con puntos. Este método necesita de algún detector para poder funcionar, una vez detectados los puntos se calcula una relación entre ellos como se ejemplifica en la Figura 3.4.

El **kernel tracking** consiste en buscar una imagen plantilla dentro de otras imágenes, a través del tiempo. Para buscar esta imagen existen varias aproximaciones, como puede ser por forma

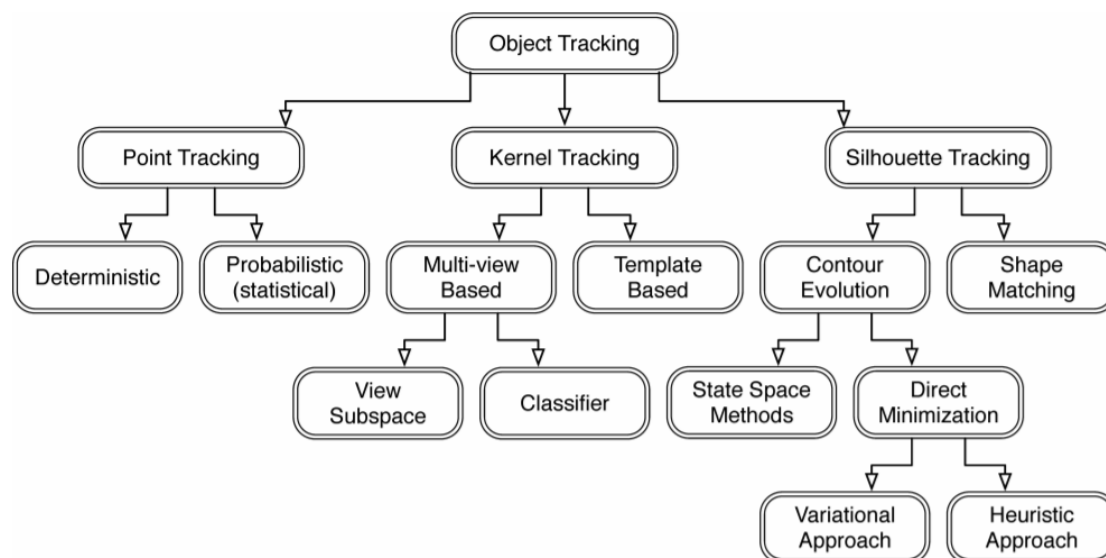


FIGURA 3.3: Clasificación de los métodos de seguimiento de objetos [64]

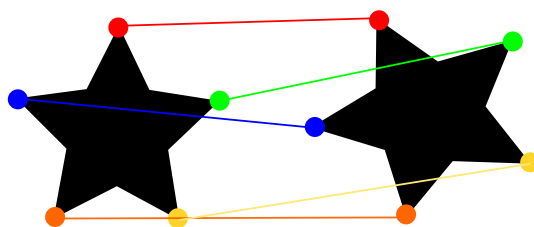


FIGURA 3.4: Ejemplo de relación de puntos.

o apariencia. La idea principal de este enfoque es calcular el movimiento del kernel o imagen plantilla, que normalmente se presenta como función de transformación paramétrica como rotación, traslación o afín.

El último método es el **Silhouette tracking** el cual consiste en utilizar la información que se encuentra dentro del objeto seleccionado. Para lograr esto, el seguimiento se realiza mediante emparejamiento de formas o evolución de contorno.

3.2.5. Modelo de Movimiento

Dentro del campo de seguimiento de objetos en imágenes, es muy importante tener claro qué es el modelo de movimiento y cual es mejor usar en cada situación. Existen diferentes tipos de transformaciones y entre más compleja se vuelve la transformación se necesitan más puntos de referencia y más componentes en la matriz de transformación. Las siguientes transformaciones son en 2D a menos que se especifique de otra manera.

Como se puede ver en [79] la transformación de traslación, consta de dos grados de libertad y modifica la ubicación del objeto en la imagen, preservando su orientación como se puede ver en la figura 3.5.

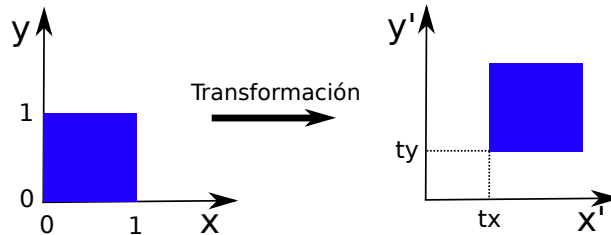


FIGURA 3.5: Transformación de traslación.

Para lograr esta transformación se agregan incrementos en x y y , los cuales se denominan t_x y t_y resultando en la ecuación $x' = x + t_x, y' = y + t_y$.

Y la cual se representa matricialmente como

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (\text{III.1})$$

Como se puede encontrar en las operaciones básicas de un software de edición de imágenes, además de la traslación también existe la transformación de escala, la cual cuenta con dos grados de libertad y se representa en la Figura 3.6. Donde se multiplica el valor de escala s , por los valores de x y y .

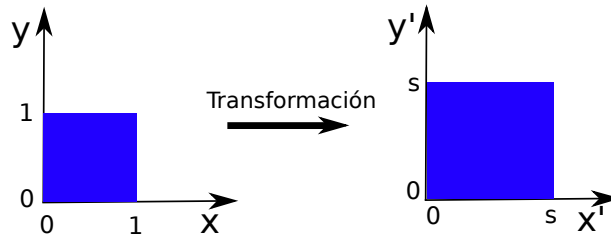


FIGURA 3.6: Transformación de escalado.

Representada matricialmente como

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (\text{III.2})$$

Dentro de las transformaciones básicas se tiene por último la rotación del objeto, la cual cuenta con dos grados de libertad y se logra con la multiplicación de un θ de la siguiente manera

$$x' = x_i \cos \theta - y_i \sin \theta, y' = x_i \sin \theta + y_i \cos \theta \quad (\text{III.3})$$

y que se puede apreciar en la Figura 3.7, resultando en la expresión matricial

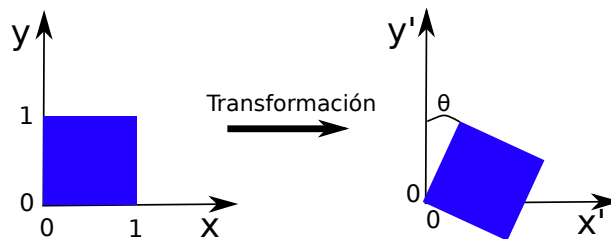


FIGURA 3.7: Transformación de rotación.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (\text{III.4})$$

La transformación euclídea rígida, es una combinación de una transformación de rotación y

traslación, que cuenta con tres grados de libertad y se expresa de la siguiente manera

$$x' = x_i \cos \theta - y_i \sin \theta + t_x, y' = x_i \sin \theta + y_i \cos \theta + t_y \quad (\text{III.5})$$

Esta transformación se puede observar en al Figura 3.8 y se expresa matricialmente como

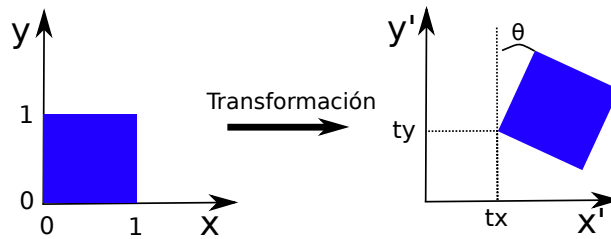


FIGURA 3.8: Transformación euclidiana.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (\text{III.6})$$

La transformación de similitud o rotación escalada, es una combinación de las transformaciones de traslación, escala y rotación como se puede observar en la figura 3.9. Puede ser escrita como en III.7.

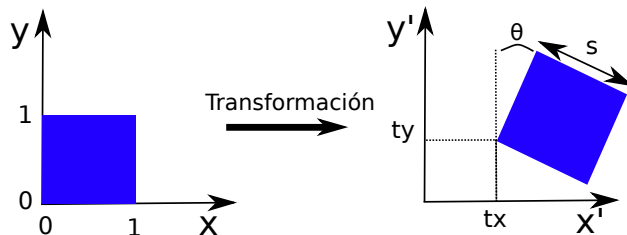


FIGURA 3.9: Transformación de similitud.

$$x' = \begin{bmatrix} sR \\ t \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (\text{III.7})$$

Resultando en

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} \quad (\text{III.8})$$

donde $\mathbf{R}$ es la matriz de rotación, $\mathbf{t}$ es el vector de traslación y $\mathbf{p}$ es el vector de coordenadas $\mathbf{x}$ y $\mathbf{y}$. Esta transformación cuenta con cuatro grados de libertad y siguiendo la nueva nomenclatura se expresa como $p' = sRp + t$.

La transformación afín o *affine* tiene la particularidad de que mantiene las líneas paralelas como se puede observar en la Figura 3.10, y cuenta con seis grados de libertad donde se introduce el termino de $\mathbf{A}$

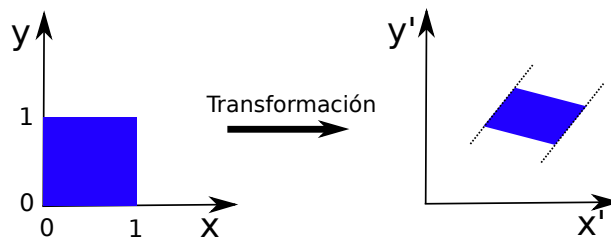


FIGURA 3.10: Transformación afín.

$$A = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \quad (\text{III.9})$$

Esta matriz esta compuesta de parámetros dependientes de la transformación, resultando en la expresión

$$p' = Ap + b \quad (\text{III.10})$$

donde $\mathbf{b}$ es la rotación, resultando en la expresión matricial

$$\begin{bmatrix} p' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ 1 \end{bmatrix} \quad (\text{III.11})$$

La última transformación es la proyectiva, en la cual se logra el mayor grado de deformación en la imagen y brinda ocho grados de libertad. En esta transformación lo único que se mantiene es la rectitud de las líneas, como se observa en la Figura 3.11, a diferencia de las demás

transformaciones, aquí es necesaria una matriz de 3 x 3 con parámetros dependientes del proceso de transformación. Esto resulta en la expresión

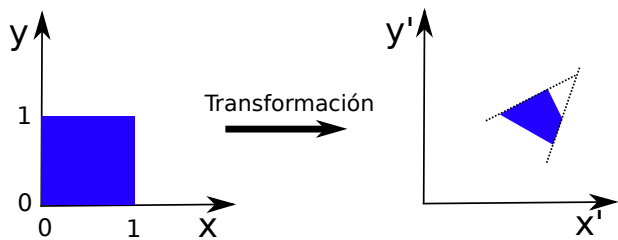


FIGURA 3.11: Transformación proyectiva.

$$p' = Ap$$

(III.12)

Transformación	Matriz	Grados de libertad	Conserva	Icono
Traslación	$[I \mid t]_{2 \times 3}$	2	Orientación	
Rígida (Euclidiana)	$[R \mid t]_{2 \times 3}$	3	Distancias	
Similitud	$[sR \mid t]_{2 \times 3}$	4	Ángulos	
Afín	$[A]_{2 \times 3}$	6	Paralelismo	
Proyectiva	$[H]_{3 \times 3}$	8	Líneas rectas	

FIGURA 3.12: Diferentes tipos de transformaciones geométricas. Se muestran además los grados de libertad que ofrecen y las características que mantienen [79]

Como se puede ver en la Figura 3.13 y en la Figura 3.12, se resumen las transformaciones geométricas que es posible realizar.

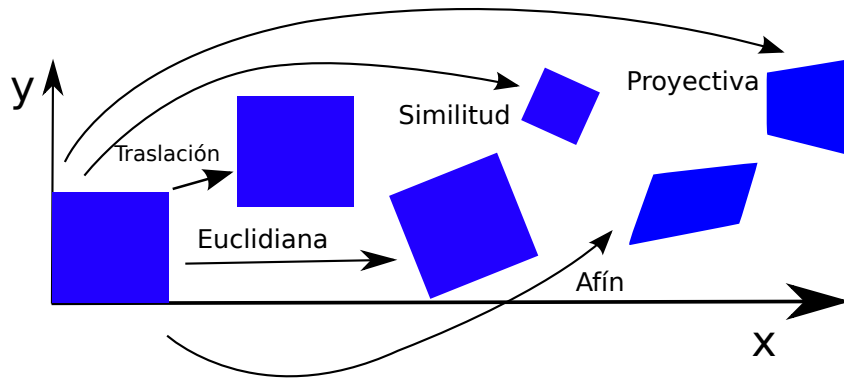


FIGURA 3.13: Diferentes tipos de transformaciones geométricas tipos de transformaciones geométricas. [79]

3.2.6. Métricas de similitud

Uno de los enfoques principales para solucionar los problemas de alineamiento es obtener la diferencia de intensidad entre imágenes para poder comparar los cambios que ocurren en una con respecto a la otra. Para el estudio de esta tesis se han considerado las siguientes medidas de error:

Sum of squared differences (SSD) Una de las métricas más utilizadas en el cálculo de la diferencia de intensidad entre imágenes, es la suma de errores cuadrados, donde el objetivo es obtener una solución de mínimos cuadrados al minimizar la Ecuación III.13

$$E_{SSD}(\mathbf{p}) = \sum_x [I(w(x; p)) - T(x)]^2 \quad (\text{III.13})$$

Sum of absolute differences (SAD) La función de suma de diferencias absolutas o *SAD* por sus siglas en inglés, es usada en ocasiones en la estimación de movimiento en vídeo debido a su buen desempeño, y se obtiene con la ecuación III.14. Sin embargo como se puede ver en [79] esta función no es diferenciable en el origen, por lo que no es recomendable para enfoques por descenso de gradiente como el que se pretende utilizar en este trabajo.

$$E_{SAD}(\mathbf{p}) = \sum_x |I(w(x; p)) - T(x)| \quad (\text{III.14})$$

Estos dos métodos son bastante eficientes computacionalmente, sin embargo también son sensibles a los cambios de intensidades. Debido a esto se han propuesto otros enfoques como lo son la *NCC* y *MI*.

Normalized cross-correlation (NCC) Este enfoque presenta una mayor robustez en el tema de los cambios de intensidad, sin embargo, esta métrica sufrirá una reducción de su desempeño con imágenes con ruido o bajo contraste. Una ventaja que entrega la NCC es que el resultado de la comparación entre las imágenes estará en el rango de -1 a 1, por lo que es más fácil de interpretar.

$$E_{NCC}(\mathbf{p}) = \frac{\sum_x [T(x) - \bar{T}] [I(w(x; p)) - \bar{I}]}{\sqrt{\sum_x [T(x) - \bar{T}]^2} \sqrt{\sum_x [I(w(x; p)) - \bar{I}]^2}} \quad (\text{III.15})$$

donde

$$\bar{T} = \frac{1}{N} \sum_x T(x) \quad (\text{III.16})$$

y

$$\bar{I} = \frac{1}{N} \sum_x I(w(x; p)) \quad (\text{III.17})$$

Normalized Mutual Information (NMI) La particularidad que presenta esta métrica es que en lugar de comparar intensidades de la imagen, esta evalúa la cantidad de información que comparten dos variables aleatorias. La información compartida por las variables $\mathbf{I}$ e I^* calculada por esta métrica, es entregada por la siguiente ecuación:

$$MI(I, I^*) = h(I) + h(I^*) - h(I, I^*) \quad [80] \quad (\text{III.18})$$

donde la entropía de shannon $\mathbf{h(I)}$ es una medida de varianza de una variable aleatoria $\mathbf{I}$ (señal, imagen). Donde $\mathbf{r}$ son los posibles valores de $\mathbf{I}$ y $p_I(r) = P(I = r)$ es la función de distribución probabilística de r . La entropía de Shannon denominada $h(I)$ de una variable discreta $\mathbf{I}$ es dada por la siguiente expresión.

$$h(I) = - \sum_r p_I(r) \log(p_I(r)). \quad (\text{III.19})$$

Como se puede ver en [81] la ecuación de la normalización es:

$$NMI(I, I^*) = \frac{h(I) + H(I^*)}{h(I, I^*)} \quad (\text{III.20})$$

Mean structural similarity index (SSIM) En [82] se explica que esta métrica fue propuesta debido a que métodos anteriores como el *mean squared error* o SSD, estiman los errores

percibidos como la cuantificación de la degradación de la imagen. Esta degradación de la imagen no es cuantificada correctamente si esta es modificada con ruido como *sal y pimienta* o filtros gaussianos, por lo que en el enfoque que presenta esta métrica (SSIM), la degradación de la imagen se considera como los cambios percibidos en la variación estructural de la información. Esta métrica propone que la información de cada píxel esta correlada con los demás píxeles, especialmente si están espacialmente cercanos. Para esto se toma en cuenta el enmascaramiento de la iluminación y el contraste. El SSIM es calculado en varias ventanas de la imagen.

El **SSIM** es una combinación ponderada de medidas comparativas:

$$SSIM(x, y) = [I(x, y)^\alpha \cdot c(x, y)^\beta \cdot s(x, y)^\gamma] \quad (III.21)$$

Esta formula está basada en tres medidas de comparación entre los ejemplos x y y : luminosidad ((I) ecuación III.22), contraste ((c) ecuación III.23) y estructura ((s) ecuación III.24). Las funciones de comparación individuales son:

$$I(x, y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \quad (III.22)$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \quad (III.23)$$

$$s(x, y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \quad (III.24)$$

con la adición de $c_3 = c_2/2$.

Donde μ_x es el promedio de x , μ_y es el promedio de y , σ_x^2 es la varianza de x , σ_y^2 es la varianza de y , $2\sigma_{xy}$ es la covarianza de x y y , $c_1 = (k_1)^2$, $c_2 = (k_2)^2$ son variables para estabilizar la división con denominadores débiles, L es el rango dinámico del valor de los sub-píxeles donde $k_1 = 0,01$ y $k_2 = 0,03$.

3.2.7. Nomenclatura matemática para el seguimiento de objetos

En esta sección se procederá a explicar la terminología utilizada en los procesos de un seguidor de objetos en imágenes. El término $x=[U,V]$ se refiere a las coordenadas del píxel x expresadas en un vector. Como se dijo anteriormente, para tener un seguidor son necesarias dos imágenes: la

plantilla o *template* y la imagen donde se buscará dicha plantilla. El template es representado como $T(x)$, lo cual se refiere a las coordenadas x en la imagen T . De la misma manera, el término $I(x)$ se refiere a las coordenadas del píxel x , en la imagen en un tiempo t . Para transformar los espacios de coordenadas entre I y T es necesaria una función de transformación llamada **warping** que es representada como $w(x,p)$, donde p son los parámetros de la función de transformación. Por último, es necesario obtener la imagen retro proyectada de I en coordenadas de T lo cual se expresa como $I(w(x,p))$ y que se puede ver en la Figura 3.14.

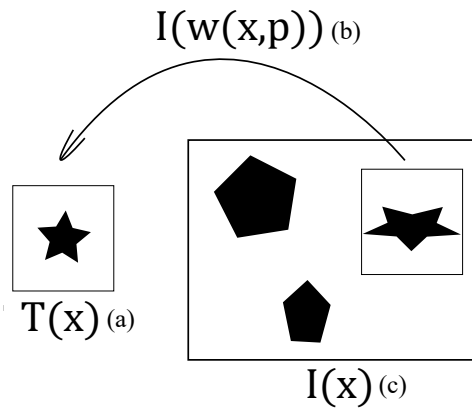


FIGURA 3.14: Proceso de un seguidor de objetos en imágenes. (a) Imagen *template*, (b) Función de transformación (*warping*), (c) imagen de referencia.

3.2.8. Gradiente

El gradiente es un operador aplicado a una función que retorna un vector que contiene las derivadas parciales de primer orden de dicha función de entrada. Este operador es utilizado para identificar las zonas donde se producen cambios de intensidad en una imagen, lo que permite segmentar un objeto.

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \quad (\text{III.25})$$

3.2.9. Hessian

El matemático alemán Ludwig Otto Hesse desarrolló este concepto en el siglo XIX, posteriormente, a esta matriz se le asignó ese nombre en su honor. Su objetivo es expresar la curvatura de una función, lo cual es útil en el campo de visión por computador para describir zonas de una imagen. Como podemos ver en [83] la matriz hessiana contiene las derivadas parciales de segundo orden de una función $f(x)$ donde x es un vector n -dimensional y tiene un tamaño de $n \times n$ como se puede observar en la ecuación III.26.

$$H o \nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (\text{III.26})$$

Otro concepto utilizado en el seguimiento de objetos es la matriz **Jacobiana**, la cual consiste en las primeras derivadas parciales de una función de entrada. Por lo que la Hessiana de una función, sería la Jacobiana de sus gradientes. Tal como se verá en las secciones posteriores, existen varias maneras de calcular esta matriz. En este trabajo se abordara el método analítico utilizado por el algoritmo de Lucas-Kanade, Baker-Matthews y el método numérico implementado por el algoritmo de Jurie-Dhome.

3.2.10. Estrategia de búsqueda

Ahora que se conoce la manera de describir los cambios en una imagen utilizando los gradientes y la forma de comparar el grado de similitud entre esos cambios, es necesario hablar de la estrategia de búsqueda utilizada para lograr la convergencia óptima en este problema de optimización. Para esto se utilizará descenso por gradiente. Una de las características de este método es que es necesario tener la certeza de que el *match* del *template* que estamos buscando está en un lugar cercano al origen de nuestra búsqueda. Debido a esta peculiaridad, al inicio de este algoritmo es necesario definir el área donde se iniciaran las iteraciones del descenso de gradiente.

Como ya se había mencionado anteriormente existen varias maneras de efectuar el descenso por gradiente, en este trabajo se utilizará el enfoque aditivo, utilizado por Lucas-kanade, y el enfoque

composicional inverso utilizado en el algoritmo propuesto por Jurie-Dome. Sin embargo en los dos se utiliza el método de optimización de Newton en cada paso del descenso.

Como se puede ver en [84] los problemas de optimización se pueden dividir en dos categorías: con restricciones y sin restricciones. Generalmente la mayoría de los problemas prácticos son con restricciones, sin embargo es común primero transformar este tipo de problema a un problema de optimización sin restricciones.

Como ejemplo se puede suponer que se quiere encontrar el conjunto de datos $x_1, x_2, \dots, x_N$ que minimice la función objetivo $f(x_1, x_2, \dots, x_N)$. El problema de optimización puede ser descrito de la siguiente manera en caso de que los datos sean con **restricciones**

$$\min f(x), \text{ sujeto a } x \in U \quad (\text{III.27})$$

donde $x = [x_1, x_2, \dots, x_N]^T, f(x) : R^N \rightarrow R$ y $U \subset R^N$. Por ejemplo, consideremos que x representa una imagen con N píxeles, cada píxel tiene un valor continuo que varía de $[0, 255]$. Si la imagen se expresa como y y es obtenida por la relación $y = Dx$ entonces la imagen original puede ser estimada por

$$\min ||y - Dx||^2 \text{ sujeto a } x \in \{x_i | 0 \leq x_i \leq 255, i = 1, \dots, N\} \quad (\text{III.28})$$

El problema descrito en la ecuación III.28 esta considerada dentro de la estructura general de optimización dada en la ecuación III.27 si $f(x) = ||y - Dx||^2$ y $U = \{x_i | 0 \leq x_i \leq 255, i = 1, \dots, N\}$.

El problema presentado tiene la forma general de un problema de optimización con restricciones. Esto es debido a que las variables son limitadas al conjunto U . Si $U = R^N$, donde N es el numero de variables, a este tipo de problema se le conoce como *problema de optimización sin restricciones*.

3.2.10.1. Método de Newton

Como ya se ha mencionado, la información derivativa es extremadamente importante debido a que la primera derivada indica donde se produce el mayor cambio en una función y la segunda derivada, indica la velocidad con que se realiza.

Basado en la idea de que cualquier valle de la función que contenga un punto mínimo se ve similar a una función cuadrática, podemos ajustar una función cuadrática $g(x)$ por x^k que relacione su

primer y segunda derivada con las respectivas de la función $f(x)$, como se puede ver en la figura 3.15

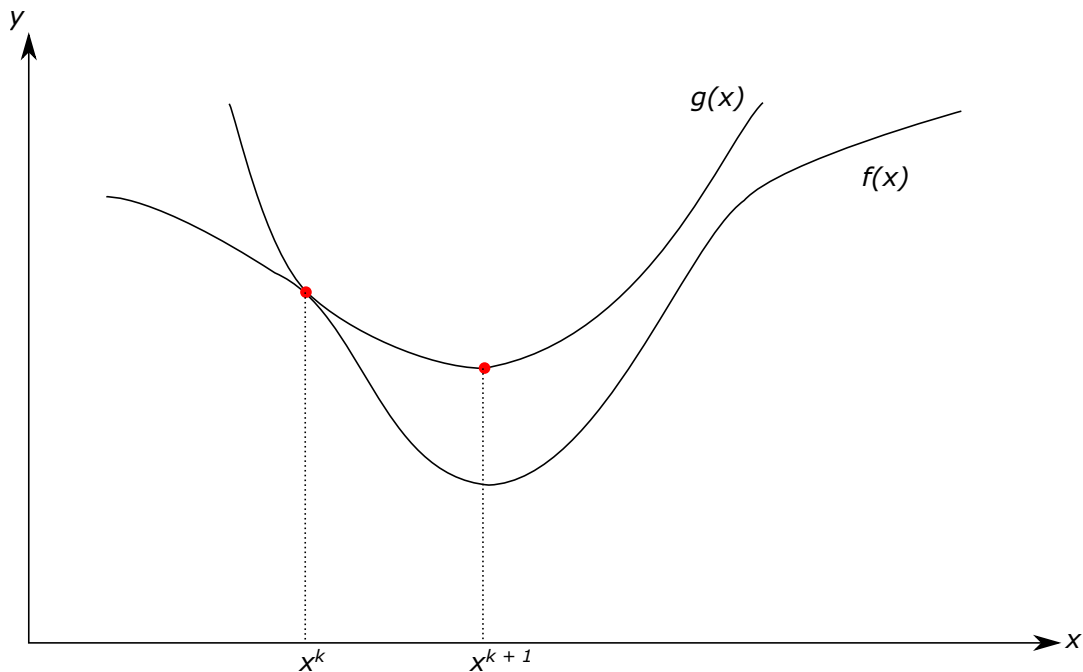


FIGURA 3.15: Ajuste de una función cuadrática al valle de un punto mínimo de una función.

Esta función cuadrática tiene la forma

$$g(x) = f(x^k) + f'(x^k)(x - x^k) + \frac{1}{2}f''(x^k)(x - x^k)^2 \quad (\text{III.29})$$

Como se puede intuir, $g(x) = f(x^k)$, $g'(x) = f'(x^k)$ y $g''(x) = f''(x^k)$. La idea es minimizar la función cuadrática $g(x)$, en vez de $f(x)$. La condición de primer orden necesaria para obtener un mínimo en $g(x)$ es que $g'(x) = 0$, que resulta en

$$f'(x^k) + f''(x^k)(x - x^k) = 0 \quad (\text{III.30})$$

Resultando el cálculo de el siguiente punto x^{k+1} como:

$$x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)} \quad (\text{III.31})$$

Al algoritmo del método de búsqueda unidimensional de Newton resulta:

1. Establecer un punto inicial estocástico como x^0 y $k \leftarrow 0$.
2. Evaluar $f'(x^k)$ y $f''(x^k)$.
3. Actualizar el punto de medida $x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)}$
4. Si $|x^{k+1} - x^k|$ es menor que el límite de error especificado anteriormente, se debe detener el algoritmo, y el punto de medida actual será la solución estimada. De lo contrario, $k \leftarrow k + 1$, e ir al paso 2.

Para permitir una optimización N-dimensional, es necesario agregar la matriz Hessiana (**H**) a la ecuación III.29

$$g(x) = f(x^k) + \nabla f(x^k)(x - x^k) + \frac{1}{2}(x - x^k)^T H(x^k)(x - x^k). \quad (\text{III.32})$$

Para que $g(x)$ tenga un mínimo en $x = x^m$, la condición de primer orden necesaria a satisfacer resulta

$$\nabla g(x)|_{x=x^m} = \nabla f(x^k) + H(x^k)(x^m - x^k) = 0, \quad (\text{III.33})$$

que da lugar a

$$x^m = x^k - H(x^k)^{-1} \nabla f(x^k) \quad (\text{III.34})$$

Teniendo en cuenta que x^m es el mínimo de la función cuadrática $g(x)$, debe reemplazar el estimado actual del mínimo entregado por la función $f(x)$. Este algoritmo de minimización se puede resumir de la siguiente manera.

1. Establecer un punto inicial estocástico como x^0 y $k \leftarrow 0$.
2. Evaluar $H(x^k)d^k = \nabla f(x^k)$ para d^k .

3. Actualizar el punto de medida $x^{k+1} = x^k + d^k$.
4. Si $\|x^{k+1} - x^k\|$ es menor que el límite de error especificado anteriormente, se debe detener el algoritmo, y el punto de medida actual será la solución estimada. De lo contrario, $k \leftarrow k + 1$, e ir al paso 2.

Debido a que el paso 2 necesita la solución de un sistema de ecuaciones lineales de $N \times N$, es necesario un método eficiente de resolución de sistemas de ecuaciones lineales para correr el algoritmo.

3.3. Algoritmo aditivo de Lucas-Kanade(LK)

A continuación se presentará el primer algoritmo de alineación de movimiento rígido que fue propuesto por LK [26]. Este algoritmo es considerado el punto de referencia para comparar el desempeño de nuevos algoritmos, ya que fue uno de los primeros en dar buenos resultados.

En la Figura 3.16 se puede ver el proceso de alto nivel seguido por el algoritmo de LK.

Como se puede ver en [85], el algoritmo propuesto por Lucas-Kanade está basado en el esquema de optimización de Newton. La medida de disimilitud que es usada en este algoritmo es el residuo cuadrático. Este residuo es la disimilitud entre la imagen de entrada y la imagen *template*, definida como:

$$\sum_x [I(w(x; p)) - T(x)]^2 \quad (\text{III.35})$$

El objetivo es minimizar la función para obtener la menor diferencia. Para lograr esto se generan instancias modificadas de $\mathbf{p}$, generando incrementos parciales en $\mathbf{p}$, y que se definen como Δp , lo que resulta en:

$$\sum_x [I(w(x; p + \Delta p)) - T(x)]^2. \quad (\text{III.36})$$

Como se mencionaba anteriormente, este enfoque es un problema de optimización resuelto con un

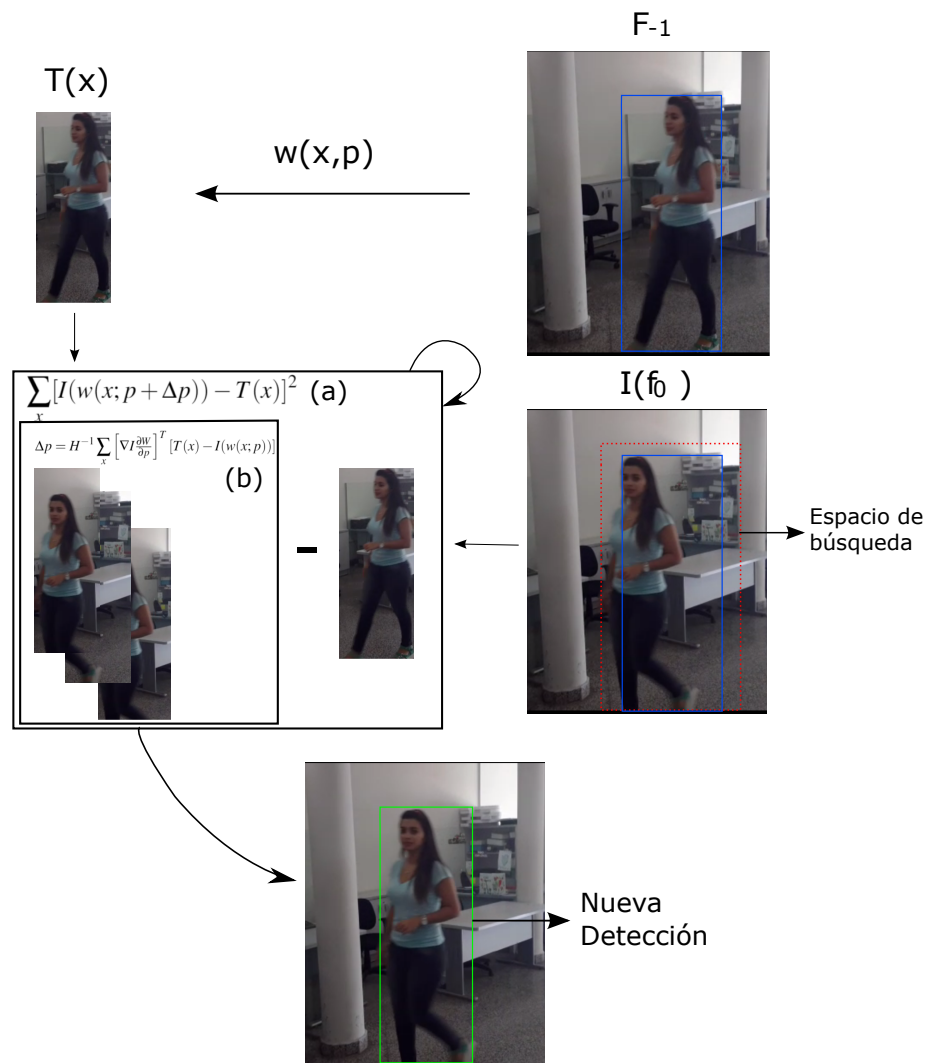


FIGURA 3.16: Resumen del proceso seguido por el algoritmo de LK. $T(x)$ es obtenido del *frame* f_{-1} y se obtienen los primeros parámetros p de la función de transformación w . En el siguiente *frame*, en este caso llamado f_0 , se realiza la búsqueda de la nueva detección utilizando la ecuación **b**, la cual actualiza los parámetros de la función de transformación con el parámetro Δp . El algoritmo converge cuando la minimización de la ecuación **a** logra alcanzar un umbral establecido.

algoritmo no lineal de Gauss-Newton por descenso de gradiente, para esto es necesario abordar el proceso de derivación de este mismo.

Para lograrlo es necesario linealizar la ecuación III.36. realizando una expansión de Taylor en $I(w(x; p + \Delta p))$ para obtener:

$$\sum_x [I(w(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x)]^2 \quad (\text{III.37})$$

La expresión $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ expresa la obtención de los gradientes de la imagen **I** evaluada en la función de transformación **W(x;p)**. Para realizar este procedimiento es necesario obtener los gradientes de **I** y posteriormente compararlos con los de la imagen **T**.

La matriz *Jacobiana* es obtenida con la expresión $\frac{\partial W}{\partial p}$ de la función de transformación, considerando esto tenemos que si $W(x; p) = (W_x(x; p), W_y(x; p))^T$ entonces:

$$\frac{\partial W}{\partial p} = \begin{pmatrix} \frac{\partial W_x}{\partial p_1} & \frac{\partial W_x}{\partial p_2} & \cdots & \frac{\partial W_x}{\partial p_n} \\ \frac{\partial W_y}{\partial p_1} & \frac{\partial W_y}{\partial p_2} & \cdots & \frac{\partial W_y}{\partial p_n} \end{pmatrix} \quad (\text{III.38})$$

En [85] se recomienda utilizar la estructura casi estandarizada, donde las derivadas parciales de un vector columna sean dispuestas como un vector fila, esto con la finalidad de generar una multiplicación de matrices cuando se aplique la regla de cadena.

Al ser un problema de optimización, es necesario minimizar la ecuación III.37 lo cual puede lograrse con el método de mínimos cuadrados y tiene una solución de forma cerrada que puede ser derivada de la siguiente manera respecto a Δp :

$$f' = 2 \sum_x [\nabla I \frac{\partial W}{\partial p}]^T [I(w(x; p)) + \nabla I \frac{\partial W}{\partial p} \Delta p - T(x)] \quad (\text{III.39})$$

despejando con respecto a Δp

$$\Delta p = H^{-1} \sum_x \left[\nabla I \frac{\partial W}{\partial p} \right]^T [T(x) - I(w(x; p))] \quad (\text{III.40})$$

donde

$$H = \sum_x \left[\nabla I \frac{\partial W}{\partial p} \right]^T \left[\nabla I \frac{\partial W}{\partial p} \right] \quad (\text{III.41})$$

por último solo resta actualizar los parámetros de movimiento $p = p + \Delta p$.

Ahora que el algoritmo ha sido explicado matemáticamente, en la Figura 3.17 se muestra un diagrama detallado del algoritmo propuesto por Lukas-Kanade y los pasos a seguir en el algoritmo 1.

Algorithm 1 LK aditivo

Iterar:

- 1: Transformar I con $\mathbf{W}(\mathbf{x}; \mathbf{p})$ para calcular $\mathbf{I}(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
 - 2: Calcular el error de la imagen $\mathbf{T}(\mathbf{x}) - \mathbf{I}(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
 - 3: Transformar el gradiente ∇I con $\mathbf{W}(\mathbf{x}; \mathbf{p})$
 - 4: Calcular la Jacobiana $\frac{\partial W}{\partial p}$ en $(\mathbf{x}; \mathbf{p})$
 - 5: Calcular las imágenes *Steepest descent* $\nabla I \frac{\partial W}{\partial p}$
 - 6: Calcular la matriz Hessiana usando la ecuación III.41
 - 7: Calcular $\sum_x \left[\nabla I \frac{\partial W}{\partial p} \right]^T [T(x) - I(w(x; p))]$
 - 8: Calcular Δp usando la ecuación III.40
 - 9: Actualizar los parámetros $p \leftarrow p + \Delta p$ hasta que $\|\Delta p\| \leq \epsilon$
-

3.4. Composicional Inverso (CI) de Baker-Matthews

Como se menciona en el punto 2.2.4, una de las ventajas del composicional inverso propuesto por Baker-Matthews [38] es que se puede pre calcular y generalizar la matriz Jacobiana. Esto se traduce en un mejor desempeño en tiempo de ejecución, lo que permite un análisis de movimiento *frame to frame*.

El enfoque por composición inversa, como su nombre lo dice invierte los papeles que desempeñan la imagen I y T . Por lo que la ecuación III.35, se expresa

$$\sum_x [T(w(x; \Delta p)) - I(w(x; p))]^2 \quad (\text{III.42})$$

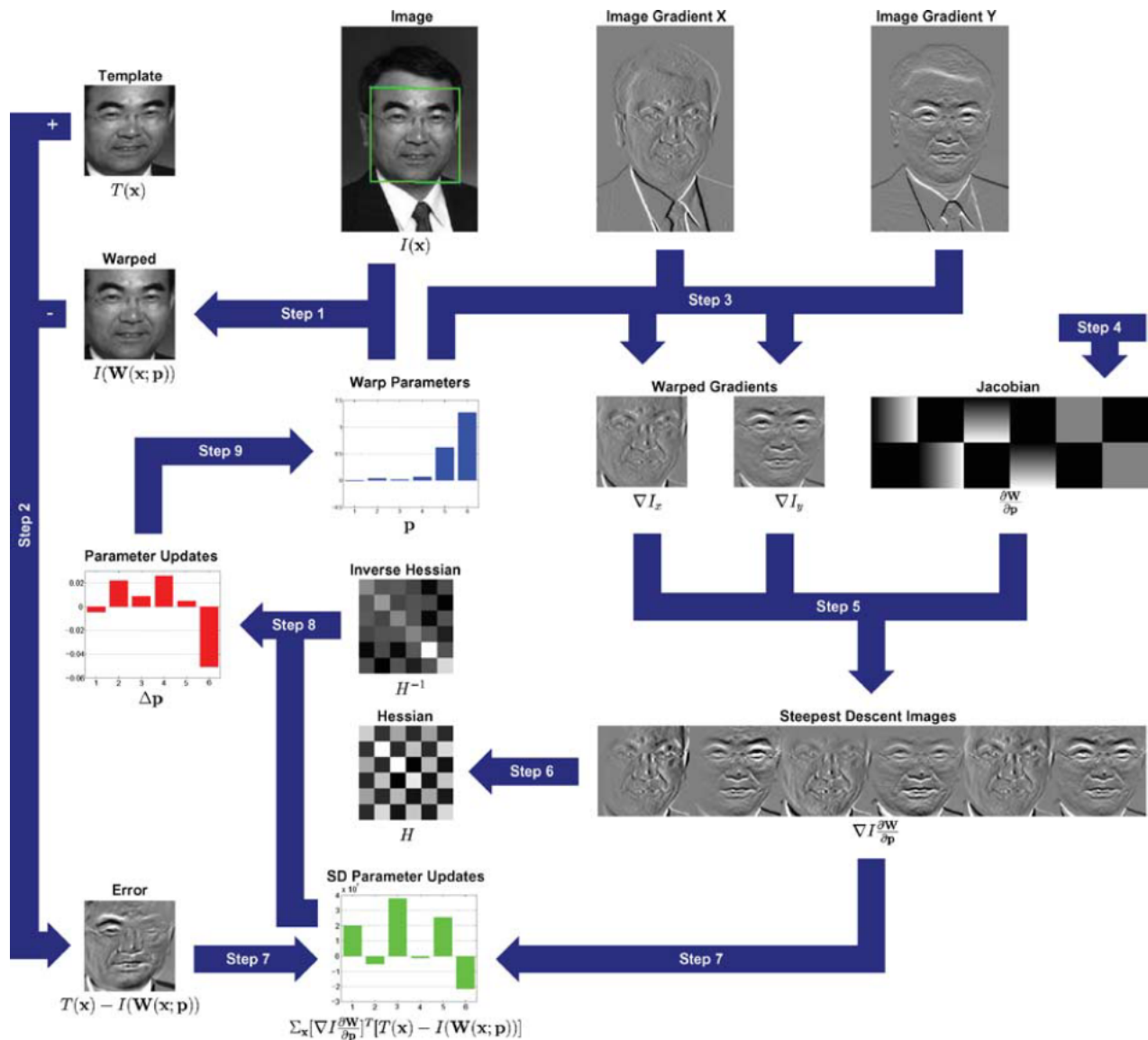


FIGURA 3.17: Diagrama del proceso seguido por el algoritmo de Lucas-Kanade [26]. La imagen I es transformada con las estimaciones actuales de la función de transformación en el paso 1 y el resultado es sustraído del *template* en el paso 2 para obtener el error de la imagen. El gradiente de I es transformado en el paso 3, y el Jacobiano es calculado en el paso 4, donde posteriormente son combinados en el paso 5 para entregar las imágenes de descenso de gradiente. En el paso 6 la Hessiana es calculada desde las imágenes de descenso de gradiente. En el paso 7 se calculan las actualizaciones de los parámetros del descenso de gradiente, al obtener el producto cruz de la imagen de error con las imágenes de descenso de gradiente. En el paso 8 la Hessiana es invertida y multiplicada con los parámetros de actualización del descenso de gradiente para obtener los parámetros de actualización finales Δp , los cuales son posteriormente agregados a los parámetros p en el paso 9. [85]

la cual tras la descomposición de Taylor resulta en

$$\sum_x [T(w(x;0)) + \nabla T \frac{\partial W}{\partial p} \Delta p - I(w(x;p))]^2 \quad (\text{III.43})$$

al ser derivada, resulta en

$$f' = 2 \sum_x \left[\nabla T \frac{\partial W}{\partial p} \right]^T [T(w(x;0)) + \nabla T \frac{\partial W}{\partial p} \Delta p - I(w(x;p))] = 0 \quad (\text{III.44})$$

el proceso de actualización de los parámetros de movimiento resulta en

$$\Delta p = H_{ic}^{-1} \sum_x SD_{ic}^T(x) E(x) \quad (\text{III.45})$$

Los cálculos de las *Steepest descend images* (SD) III.46, la Hessiana III.47 y el error entre las imágenes III.48 se realizan de manera similar cambiando la imagen I por el *template* T . Por último se actualizan los parámetros de movimiento $W(x;p) \leftarrow W(x;p) \circ W(x;\Delta p)^{-1}$. En la Figura 3.18, se puede observar una descripción gráfica del proceso y los pasos a seguir se listan en el algoritmo 2.

$$SD_{ic}(x) = \left[\nabla T \frac{\partial W}{\partial p} \right] \quad (\text{III.46})$$

$$H_{ic} = \sum_x SD_{ic}^T(x) SD_{ic}(x) \quad (\text{III.47})$$

$$E = [I(w(x;p)) - T(x)] \quad (\text{III.48})$$

3.5. Algoritmo de aprendizaje de Jurie-Dhome

El algoritmo de aprendizaje propuesto por Jurie-Dhome [86], se encuentra dentro de la categoría del composicional inverso. Cabe resaltar que la notación original en el trabajo de Jurie-Dhome será

Algorithm 2 composicional inverso (CI)

Pre-calcular:

- 1: Evaluar el gradiente ∇T del *template* $T(x)$
- 2: Calcular la Jacobiana $\frac{\partial W}{\partial p}$ en $(x;0)$
- 3: Calcular las imágenes *Steepest descent* $\nabla T \frac{\partial W}{\partial p}$
- 4: Calcular la matriz Hessiana usando la ecuación III.41 (ajustar)

Iterar:

- 1: Transformar I con $W(x;p)$ para calcular $I(W(x;p))$
 - 2: Calcular el error de la imagen $I(W(x;p)) - T(x)$
 - 3: Calcular $\sum_x \left[\nabla T \frac{\partial W}{\partial p} \right]^T [I(W(x;p)) - T(x)]$
 - 4: Calcular Δp usando la ecuación III.40 (ajustar)
 - 5: Actualizar los parámetros $W(x;p) \leftarrow W(x;p) \circ W(x; \Delta p)^{-1}$ hasta que $\|\Delta p\| \leq \epsilon$
-

reemplazada por la utilizada en este documento, la cual sigue la empleada en el artículo de [85] para facilitar el entendimiento y comparación entre los tres algoritmos.

Tal como Lukas-Kanade [26], en Jurie-Dhome el objetivo es minimizar la función III.49 por mínimos cuadrados para obtener el vector de movimiento del *template*.

$$p(t) = \|T(x), t_0) - I(w(x; p), t)\| \quad (\text{III.49})$$

Donde t es el tiempo que transcurre durante la ejecución del algoritmo. Para esto el primer paso a realizar es la fase de entrenamiento, donde se precalcula y generaliza una Jacobiana que es modelada por hiperplanos.

3.5.1. Cálculo de Jacobiana por hiperplanos

Para entender por qué es posible generalizar la matriz Jacobiana, primero es necesario entender de donde se obtiene.

En el trabajo propuesto por Hager y Belhumeur [88] se presenta un cálculo muy eficiente y directo de los parámetros de movimiento al escribir:

$$p(t + \tau) = p(t) + \mathbf{A}(t + \tau)(I(f(R; \mu_0^*), t_0) - I(f(R; \mu(t)), t + \tau)) \quad (\text{III.50})$$

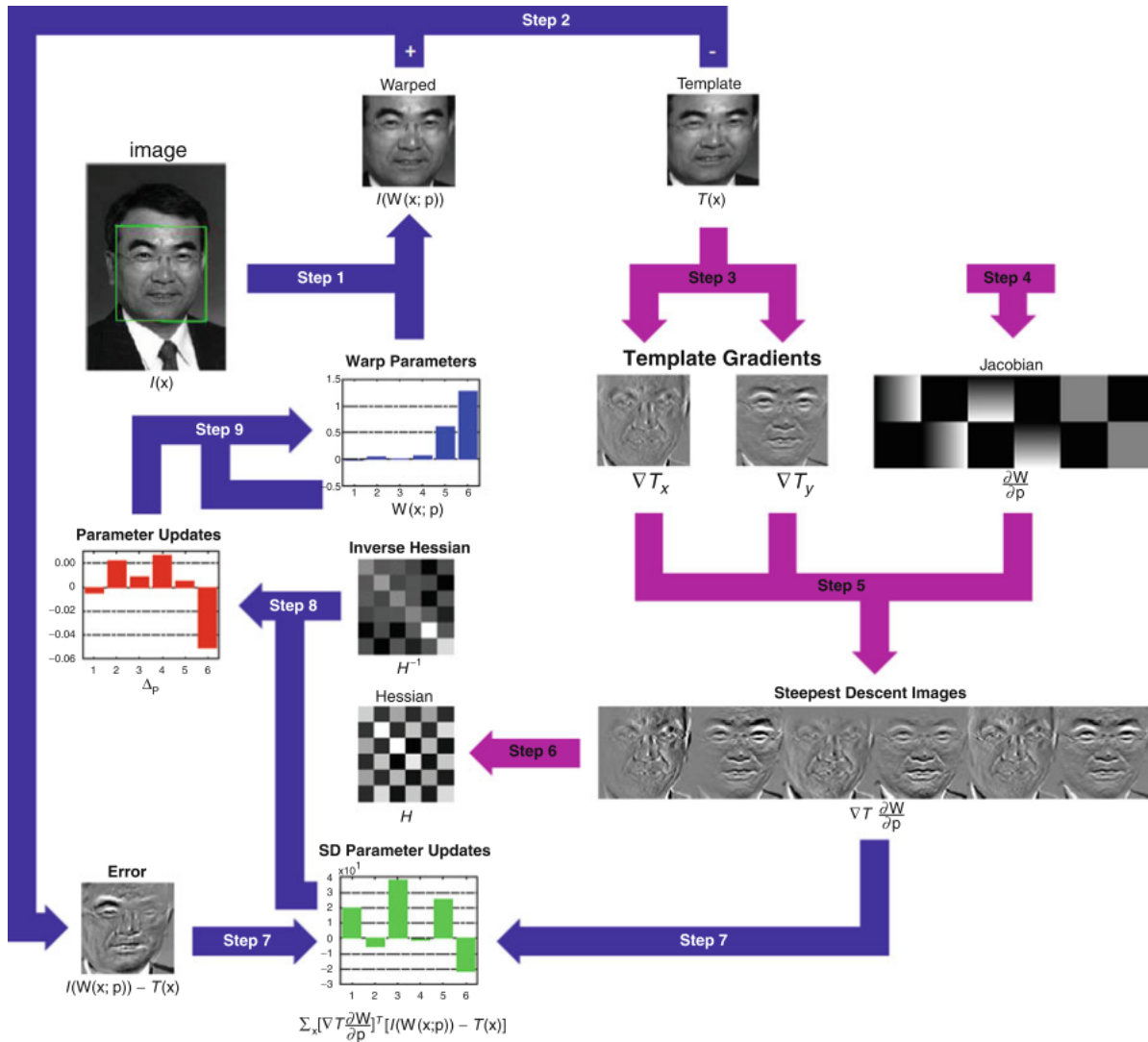


FIGURA 3.18: Diagrama seguido por el algoritmo de composición inversa. Los pasos 3-6 son realizados una vez como un paso previo a las iteraciones del algoritmo principal. El algoritmo principal consiste en iterar la transformación de la imagen (paso 1), obtener la diferencia de imágenes (paso 2), calcular el producto punto (paso 7), multiplicación con la inversa de la Hessiana (paso 8) y la actualización de los parámetros de transformación (paso 9). [87]

donde t representa el tiempo y τ representa el tiempo entre dos imágenes.

Si se define la diferencia de intensidades entre la imagen $\mathbf{I}$ y la imagen $\mathbf{T}$ como $\delta i(t + \tau) = T(x, t_0) - I(w(x; p), t)$ y $\Delta p(t + \tau) = p(t + \tau) - p(t)$, entonces la ecuación III.50 puede ser escrita como:

$$\Delta p(t + \tau) = A(t + \tau) \delta i(t + \tau) \quad (\text{III.51})$$

Como podemos ver en la ecuación III.51 $\mathbf{A}$ es la matriz Jacobiana. Esta ecuación puede ser vista como la modelización de $\mathbf{n}$ hiperplanos, por lo que ahora será definida como A_h . Si escribimos a_{ij} como los elementos de la matriz A_h la ecuación puede ser escrita como III.51.

$$\begin{aligned} (a_{11}, \dots, a_{1j}, \dots, a_{1N}, -1)(\delta_{i1}, \dots, \delta_{ij}, \dots, \delta_{iN}, \delta_{p1})^T &= 0 \\ (a_{i1}, \dots, a_{ij}, \dots, a_{iN}, -1)(\delta_{i1}, \dots, \delta_{ij}, \dots, \delta_{iN}, \delta_{pi})^T &= 0 \\ (a_{n1}, \dots, a_{nj}, \dots, a_{nN}, -1)(\delta_{i1}, \dots, \delta_{ij}, \dots, \delta_{iN}, \delta_{pn})^T &= 0 \end{aligned} \quad (\text{III.52})$$

De esta manera estos coeficientes pueden ser estimados por una función de mínimos cuadrados.

3.5.2. Fase de entrenamiento

Para esto es necesario definir el *template*, el cual consiste en un conjunto de N puntos $\mathbf{x}$. El proceso de aprendizaje consiste en la generación de pequeñas perturbaciones en los parámetros de movimiento p_0 , denominadas δp . El proceso de perturbación, es definido como $p'_0 = p_0^* + \delta p$, donde p_0^* es la posición inicial del *template* en la imagen. Este proceso es el que genera los cambios de brillo en $\delta i = T(x, t_0) - I(w(x; p'_0), t)$.

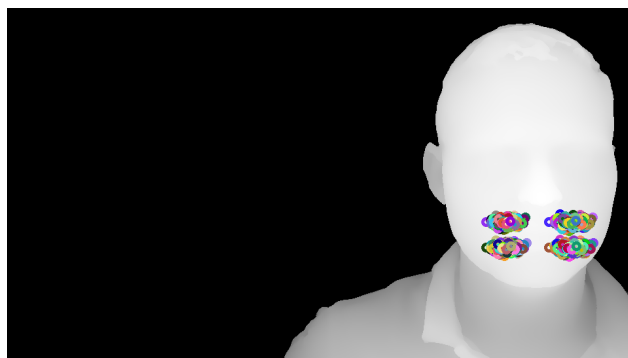
Para obtener el modelo lineal es necesario realizar N_p perturbaciones δp , donde $N_p > N$. Posteriormente se crean N_p parejas $(\delta i^k, \delta p^k)$, $k \in [1, N_p]$, con $\delta i^k = (\delta i_1^k, \dots, \delta i_N^k)^t$ y $\delta p^k = (\delta p_1^k, \dots, \delta p_N^k)^t$. La matriz A_h es obtenible cuando $\sum_{k=1}^{N_p} (\delta p^k - A_h \delta i^k)$ sea mínima.

El proceso de perturbación de la imagen, se puede observar en la Figura 3.19 donde a) es una imagen de textura y b) una de profundidad.

Si se reescribe $H = (\delta i_1^k, \dots, \delta i_N^k)$, $Y = (\delta p_1^k, \dots, \delta p_N^k)$ y se asume que $N_p > N$ se habrá obtenido un



(a) Perturbaciones en una imagen de textura



(b) Perturbaciones en una imagen de profundidad

FIGURA 3.19: Proceso de perturbación del algoritmo de aprendizaje, a) imagen de textura y b) imagen de profundidad.

sistema sobre determinado. Por lo que A_h podrá ser calculada desde $Y = A_h H$ a $A_h = YH^t (HH^t)^{-1}$.

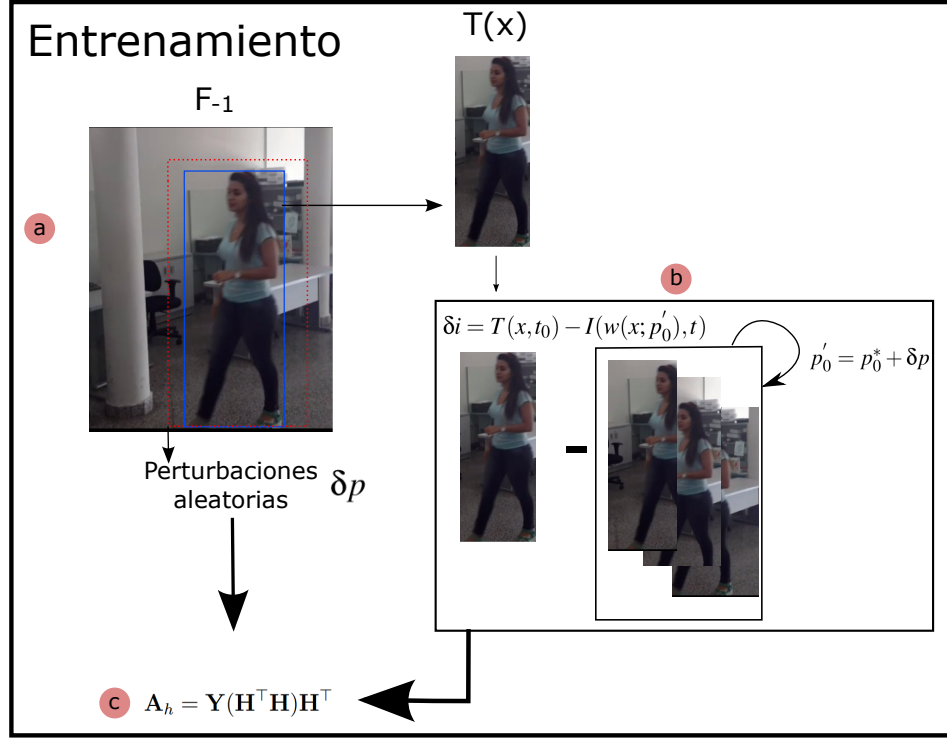


FIGURA 3.20: Fase de entrenamiento Jurie-Dhome. a) Se generan perturbaciones aleatorias alrededor del *template* (T), lo que genera el conjunto δp . b) Se utiliza el conjunto δp para calcular el conjunto de la diferencia de brillo δi . c) Si convertimos $H = (\delta i_1^k, \dots, \delta i_N^k)$, $Y = (\delta p_1^k, \dots, \delta p_N^k)$ entonces podemos calcular A_h .

3.5.3. Fase de seguimiento

Como se puede ver en [86] para calcular los parámetros δp , es necesario calcular:

$$\delta p = A_h \delta i = A_h [T(x) - I(w(x; p'_0))] \quad (\text{III.53})$$

Con esta ecuación se obtiene el cambio de brillo δi , si el vector de parámetros hubiera sido p_0^* .

Por último, con el parámetro δp calculado, es posible transformar la ubicación de x de las coordenadas del *template* a las coordenadas de la imagen con la ecuación III.54, donde $p' = p_0^* + \delta p$.

$$x' = w^{-1}(w(x; p'); p_0^*) \quad (\text{III.54})$$

Si se definen las coordenadas del *template* como $x = (x, y)$ y las coordenadas de la imagen actual como $u = (u, v)$. Entonces se puede agregar la relación $u = (u, v)$ con la ecuación III.54, se tiene que

$$u' = f(x'; p') = w(w^{-1}(w(x; p_0^*); p_0') p') \quad (\text{III.55})$$

En la Figura 3.21 se puede observar una explicación grafica del algoritmo y en el Algoritmo 3 se observa el procedimiento seguido por el enfoque de aprendizaje.

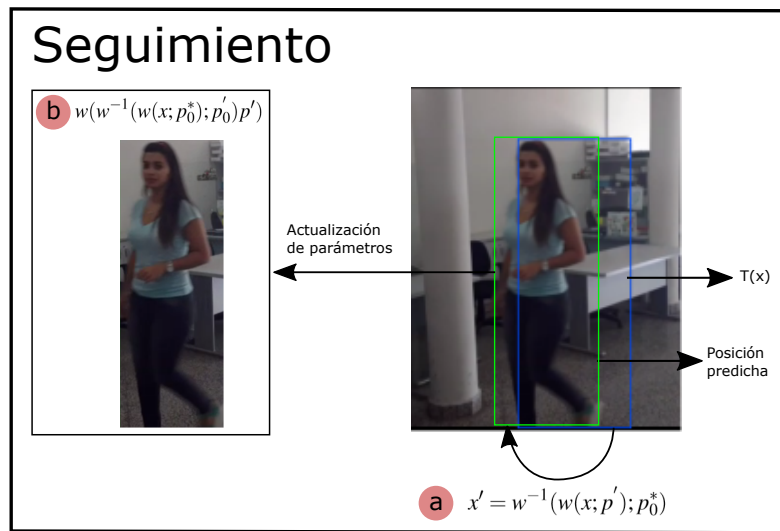


FIGURA 3.21: Fase de seguimiento Jurie-Dhome. a) Con el parámetro p' es posible predecir la posición donde se encontrara la mejor correspondencia. b) Con esta ecuación se actualiza el modelo de movimiento para calcular el movimiento en la siguiente iteración.

Algorithm 3 Algoritmo de aprendizaje (JD)

Entrenamiento:

- 1: Generar el p_0 que transforme la imagen F_{-1} a la imagen T.
- 2: perturbar los parámetros de p_0 para generar los conjuntos H y Y .
- 3: Utilizando la ecuación $A_h = YH^t(HH^t)^{-1}$ se calcula A_h .

Iterar:

- 1: Transformar I con $W(x; p)$ para calcular $I(W(x; p))$.
 - 2: Calcular el error de la imagen $I(W(x; p)) - T(x)$.
 - 3: Calcular Δp con la ecuación III.53.
 - 4: Actualizar los parámetros $p = p + \Delta p$ hasta que $\|\Delta p\| \leq \epsilon$.
-

Capítulo IV

Herramientas de implementación

4.1. Blender

Blender es un software gratuito y open source de creación de contenido 3D. Este software se utilizó en este proyecto para la creación de imágenes RGB-D sintéticas a partir de un modelo real obtenido con una cámara estereoscópica *Sense 3D* [89] (Figura 4.1).



FIGURA 4.1: Cámara utilizada para generar modelo 3D.

Para generar el conjunto de entrenamiento, se utilizó el modelo de la Figura 4.2. Para generar las imágenes sintéticas se agregó una trayectoria a la cámara default de Blender, lo que permitió generar una secuencia de imágenes, las cuales al unirse forman un vídeo que simula la adquisición de datos en condiciones no controladas.

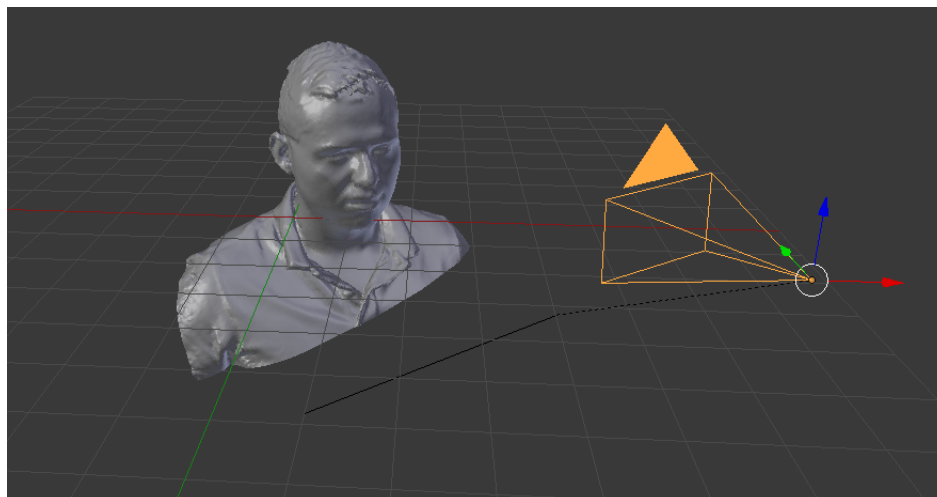


FIGURA 4.2: Modelo utilizado para la generación de imágenes sintéticas.

El proceso del renderizado de imágenes consta de 200 *frames*, siguiendo una trayectoria en línea recta, la cámara se mantiene en un ángulo de 87.43, 0 y 36.01 X, Y, Z de Euler. Una vez configurada la trayectoria a seguir, fue necesario configurar los **nodos de composición**, los cuales permiten mejorar o adquirir características de una imagen. En Blender es posible construir y asociar los nodos de composición por medio de una interfaz sencilla tal como se muestra en la Figura 4.3 .

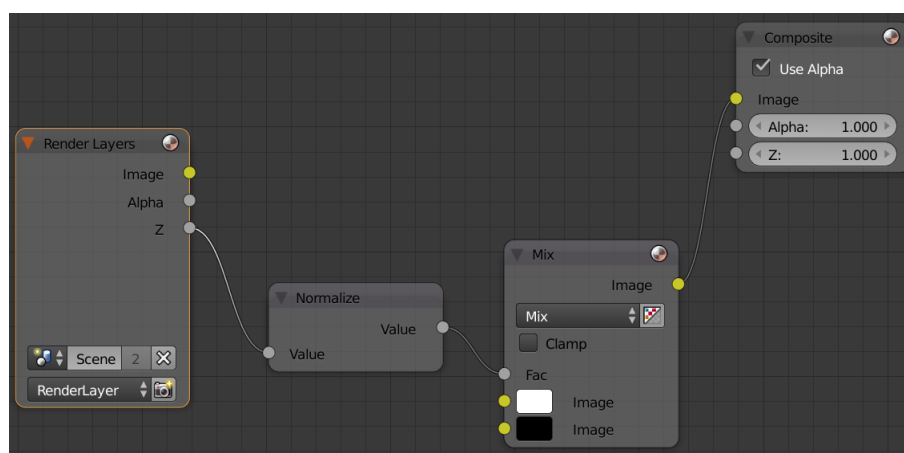


FIGURA 4.3: Nodos de composición utilizados para generar las imágenes RGB-D sintéticas. De izquierda a derecha los nodos: (a) render layers, (b) normalize, (c) mix y (d) composite.

En la Figura 4.3 se pueden observar los nodos utilizados en el proceso de renderizado, los cuales se describen a continuación:

Render layers El nodo (a) *render layers* permite obtener el eje Z del sistema de coordenadas, el cual representa la profundidad, sin embargo sin un correcto formateo la imagen resultará en blanco total a la hora del renderizado.

Normalize El uso del nodo (b) *normalize* se debe a que es una manera sencilla de ajustar los valores mínimos y máximos de profundidad adquiridos por la cámara, ya que de lo contrario será necesario ajustar el valor de *clipping* manualmente para ajustar el inicio y el final del objeto que se tiene frente a la cámara.

Mix Para representar el rango de valores generado por el nodo *normalize*, es necesario encadenarlo con un nodo (c) *mix* en el cual se definen los colores para los valores mínimos y máximos y genera la interpolación entre estos.

Composite Por último se agrega el nodo (d) *composite* para ver el resultado del encadenamiento de los nodos antes mencionados.

Esta configuración permitió la creación de imágenes RGB-D sintéticas como la que podemos ver en la Figura 4.4.

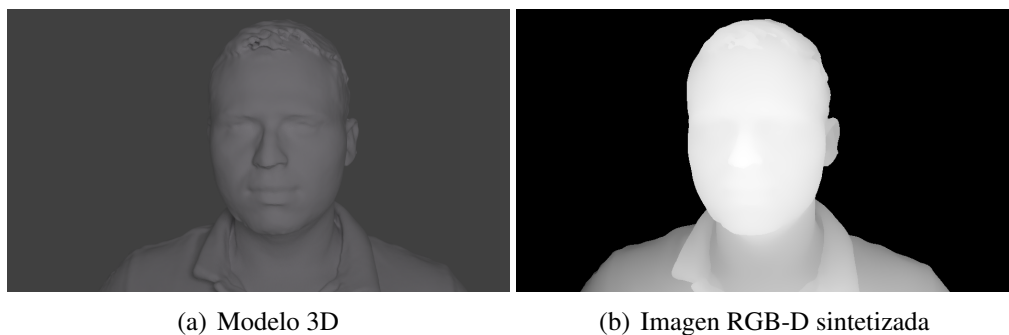


FIGURA 4.4: En a) Se puede observar el modelo 3D utilizado. b) Se muestra el resultado obtenido del proceso de sintetización de una imagen RGB-D .

4.1.1. Definición de Keypoints

Este proceso consta de dos partes, la primera consistió en ubicar cubos en algunas partes que se consideran significativas de un rostro como son la boca con cinco cubos y la punta de la nariz con cuatro, como se puede apreciar en la Figura 4.5.

La segunda parte consiste en la proyección de las coordenadas 3D de los cubos, a coordenadas 2D en el *renderizado* de la imagen. Cabe destacar que al momento del *renderizado* de las imágenes los cubos se ocultan para evitar crear resultados favorables. Para esto se utiliza la función de *blender world to camera view* de la librería *bpy extras*, la cual recibe la escena en la que se está trabajando, la referencia a la cámara con la que se quiere *renderizar* la imagen y la matriz global de traslación



FIGURA 4.5: Método de definición de keypoints.

de esta misma. Posteriormente se debe definir la escala del *renderizado*, la cual se define con el valor de la escena denominado *resolution porcentaje* dividido entre cien. El siguiente paso es obtener el tamaño del *renderizado*, para lograr esto se debe crear un arreglo con la resolución en X multiplicada por la escala de *renderizado*, la cual se obtiene de la propiedad *scene render resolution x*, el mismo procedimiento aplica para calcular los valores de Y . Por último se multiplica el valor de X obtenido con la función *world to camera view* por el primer valor del arreglo del tamaño de *renderizado*. Respecto a Y se hace la misma operación pero con el segundo valor de dicho arreglo, y después se resta el valor resultante al segundo valor de este arreglo. Este proceso es resumido en el Algoritmo 4.

Algorithm 4 Algoritmo de proyección de puntos 3D a 2D

- 1: se entrega a la función $2Dcoords = world\ to\ camera\ view(scene, obj, co)$ donde *scene* es la referencia a la escena, *obj* a la cámara y *co* la matriz global de traslación.
 - 2: $renderScale = resolutionPercentage / 100$
 - 3: $renderSize = (resolutionX * renderScale, resolutionY * renderScale)$
 - 4: $x = 2Dcoords.x * renderSize[0]$, $y = 2Dcoords.x * renderSize[1]$
 - 5: $y = renderSize[1] - y$
-

Este proceso permitió la medición del desempeño de los algoritmos comparados en este trabajo, permitiendo mostrar el *ground truth* generado como se muestra en la Figura 4.6.

4.2. Base de datos

Como se mencionó anteriormente la base de datos fue generada a partir de los miembros del programa de magíster: profesores y alumnos. Algunos ejemplos de las entradas de esta se pueden

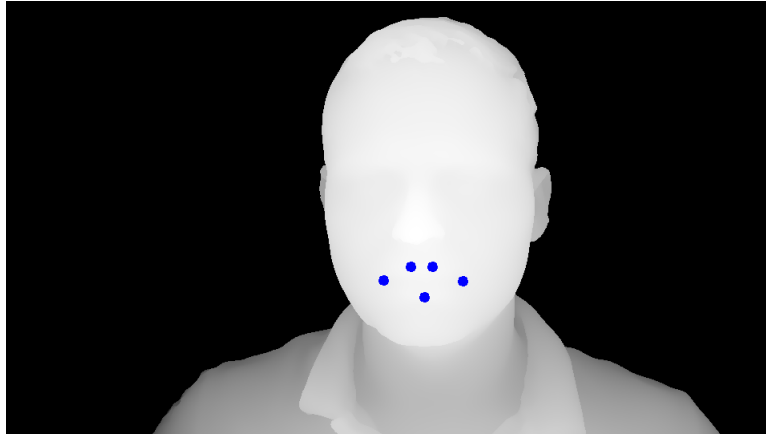


FIGURA 4.6: Representación del *ground truth*.

observar en la Figura 4.7. Esta base de datos puede ser accedida de manera gratuita, haciendo la solicitud con el profesor Juan Bekios [90] del departamento de ingeniería de sistemas y computación.

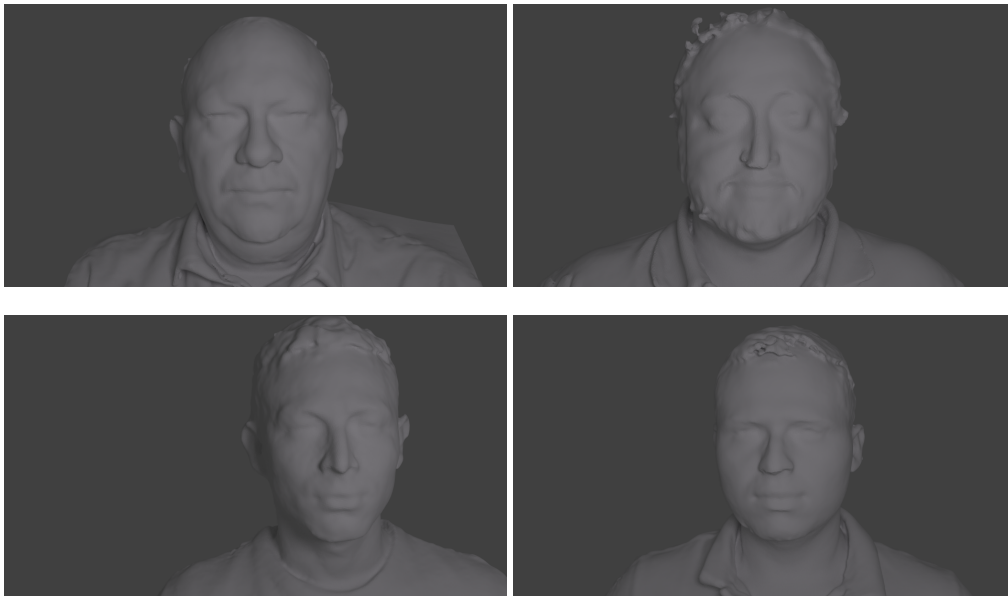


FIGURA 4.7: Modelos que conforman la base de datos creada .

El tamaño nativo del escaneo es muy grande para ser visualizado correctamente por una cámara estándar de blender, por lo que se redujo su tamaño en X, Y, Z a 0.012.

Capítulo V

Resultados experimentales

Para comparar la efectividad de los algoritmos en imágenes RGB-D sintéticas, primero se decidió hacer pruebas en una secuencia de imágenes de textura con el objetivo de comparar el comportamiento de los algoritmos sobre la información aportada por las profundidades y la de las intensidades de grises. Para realizar esta comparación se seleccionaron los algoritmos clásicos de *tracking* propuestos por Lucas-Kande [26] y Baker-Matthews [38], el primero debido a su comprobada eficiencia y el segundo debido a su velocidad de procesamiento. También se aborda un tercer enfoque más reciente que los dos anteriores y menos utilizado, este es efectuado por aprendizaje y fue propuesto por Jurie-Dhome [86]. Se considera que con un correcto entrenamiento podría funcionar con la precisión del algoritmo aditivo propuesto por Lucas-Kanade y con la velocidad del composicional inverso propuesto por Baker-Matthews.

Las métricas utilizadas para este experimento fueron SSD, NMI, SSIM y el tiempo de ejecución medido en segundos. Se utilizó el SSD debido a que expresa la diferencia cuadrada entre píxeles y es una de las medidas más sencillas de interpretar y por lo tanto una de las más utilizadas. Sin embargo esta medida no considera la diferencia estructural de una imagen, por este motivo se utilizó NMI ya que toma en cuenta la cantidad de información compartida por las imágenes. Para tener un espectro de comparación más grande se utilizó SSIM, la cual mide la entropía entre las dos imágenes y toma en cuenta la iluminación, el contraste y la estructura.

Ajuste de parámetros

Para los algoritmos aditivo y composicional inverso se fijó un límite de diez iteraciones por frame con un error de convergencia de 1×10^{-5} . La elección de los parámetros de iteración y error de

convergencia fueron seleccionados a partir del trabajo de Dirk-Jan Kroon [91]. El algoritmo de aprendizaje consta de seis mil iteraciones de aprendizaje precalculado y solo una en tiempo de ejecución.

Como resultado de la comprobación en imágenes de textura se obtuvo la secuencia mostrada en la Figura 5.1 y respecto a las imágenes de profundidad se obtuvo la secuencia mostrada en la Figura 5.2.

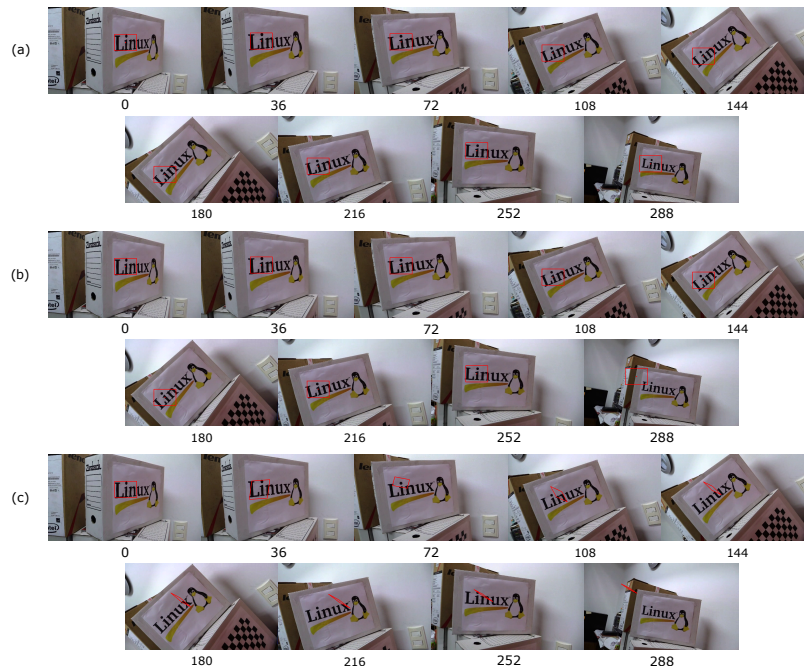


FIGURA 5.1: Secuencia de imágenes de textura generadas y proceso de seguimiento aplicando el algoritmo (a) aditivo, (b) de composición inversa, (c) aprendizaje.

La secuencia de imágenes mostrada en la Figura 5.2 es el resultado del proceso de sintetizado de imágenes RGB-D, donde se encuentra marcado el *ground truth* con puntos azules. Estos puntos describen características de la comisura de la boca, en rojo se puede ver el *template* generado durante la ejecución del (a) algoritmo aditivo, (b) composicional inverso y (c) aprendizaje.

5.1. Imágenes 2D

Esta secuencia de imágenes fue utilizada para probar los algoritmos aditivo, composicional inverso y aprendizaje en varias condiciones que presentan cambios de perspectiva, rotación, traslación y escalado. Esto con el propósito de tener un punto de comparación con las imágenes creadas y así poder evaluar más eficientemente su rendimiento, así como obtener conclusiones más significativas.



FIGURA 5.2: Secuencia de imágenes de profundidad generadas y proceso de seguimiento aplicando el algoritmo (a) aditivo, (b) de composición inversa, (c) aprendizaje.

5.1.1. Aditivo

Como se puede apreciar en la Tabla 5.1 el **SSD** empieza en un valor cercano a cero, lo que indica que la función de transformación inicial funciona correctamente. Desde el frame cero al 150 de la Figura 5.1, aproximadamente, se puede observar un incremento del error para estabilizarse en los *frames* siguientes concordando con una combinación de transformaciones de escala, traslación y rotación. En los *frames* siguientes se aprecian transformaciones menos complicadas que consta en su mayoría de rotaciones. Debido a lo anterior se logra una reducción del error hasta el frame 250, donde se introducen de nuevo transformaciones combinadas de escala, traslación y rotación, lo que presenta un incremento súbito entre los *frames* 250 y 270. Del frame 270 al 320 aproximadamente, se observa una estabilización del error para después lograr una disminución del mismo del 320 en adelante.

Este enfoque tiene el tiempo más lento de ejecución, presenta variaciones muy pequeñas entre *frames* y cuenta con un tiempo máximo de *frame* de 4.93 segundos. El tiempo máximo presentado es la única variación que presenta una mayor cantidad de tiempo, la media es de 4.2 segundos. Las pocas variaciones que presenta consisten en tiempos menores de procesamiento.

En la información proporcionada por la **NMI** se puede observar una estrepitosa disminución durante los primeros *frames*, esto debido a las transformaciones de rotación, escala y traslación aplicadas a la imagen. Sin embargo a pesar de esa caída, se puede observar como a través de todo el tiempo de ejecución se mantiene una equidad de información entre el *template* y la imagen actualizada ya que se observa un rango de 0.0435 entre los valores mínimos y máximos totales. A pesar de la información aportada por **NMI**, en **SSIM** se puede observar un constante descenso a través de los *frames* hasta el final de la secuencia cuando las transformaciones aplicadas a la imagen se vuelven más sencillas. Esto se entiende como que el algoritmo es capaz de seguir el *template*, sin embargo hay una pérdida notable de información.

TABLA 5.1: Enfoque aditivo

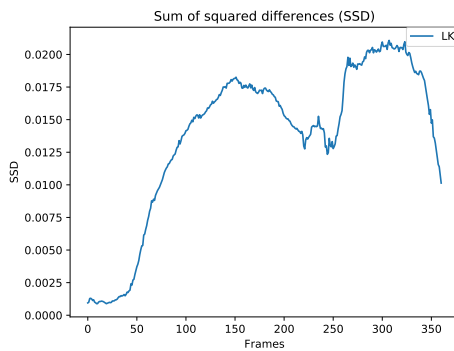
	SSD	NMI	SSIM	Time
Mínimo	0.0008	0.5215	0.5930	3.6742
Máximo	0.0210	0.5651	0.9084	4.9303
Promedio	0.01375	0.5381	0.6844	4.2941
Desviación estándar	0.0060	0.0047	0.0880	0.1198

5.1.2. Composicional inverso

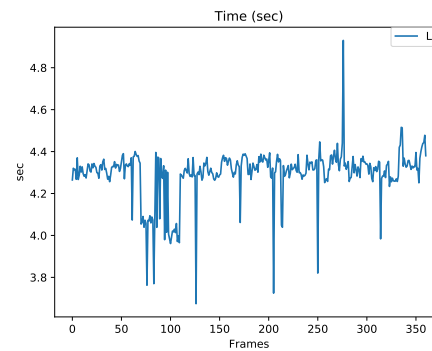
Como se puede observar en la Tabla 5.2 **SSD** muestra un rango de valores mucho mayores que los presentados en el algoritmo aditivo. Un comportamiento notable observado, es que después del *frame* 250, el error incrementa súbitamente debido a la incapacidad de convergencia presentada en los siguientes *frames*, lo que se representa en la Tabla 5.4 (a) como un aumento constante del error.

Este enfoque se posiciona como el algoritmo más rápido de los tres comparados, presentando un tiempo máximo de 0.2590 segundos. El primer frame muestra una velocidad cercana a la máxima debido al procesamiento del precálculo de la matriz *Hessiana*, esto también indica que en el *frame* donde se presentó el tiempo máximo no fue posible lograr la convergencia del algoritmo.

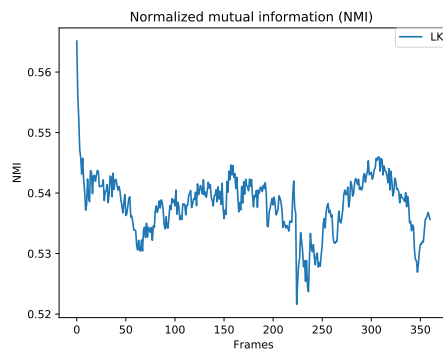
NMI muestra pequeñas variaciones en la cantidad de información compartida hasta el frame 250, entre el *template* y el *frame* analizado, en comparación con el enfoque aditivo. Sin embargo, en todos los *frames* siguientes es incapaz de converger por lo que pierde la posición del *template* y se posiciona en partes cercanas a las letras que conforman el *template* por lo que esta métrica recibe información errónea del frame 300 en adelante.



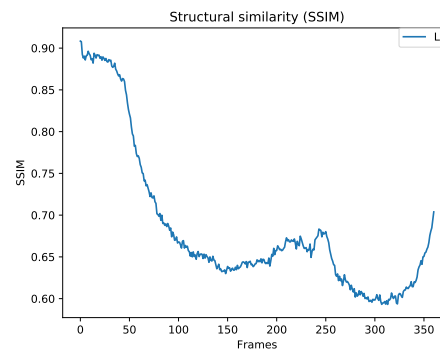
(a) Suma de error cuadrático (SSD)



(b) Tiempo de procesamiento (sec)



(c) Información mutua normalizada (NMI)



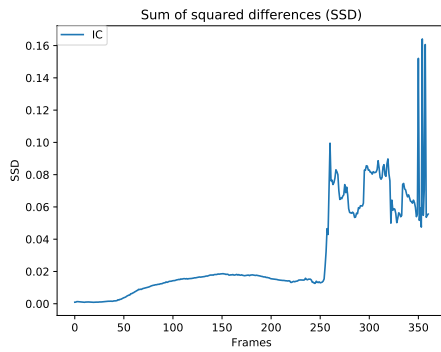
(d) Similitud estructural (SSIM)

FIGURA 5.3: Gráfica de métricas del algoritmo aditivo, en a) Gráfica del error obtenido de la suma de la diferencia cuadrada entre el *template* y la imagen actual. b) El tiempo (segundos) utilizado para procesar cada frame. c) Gráfica de la información mutua normalizada. d) Gráfica del índice de similitud estructural.

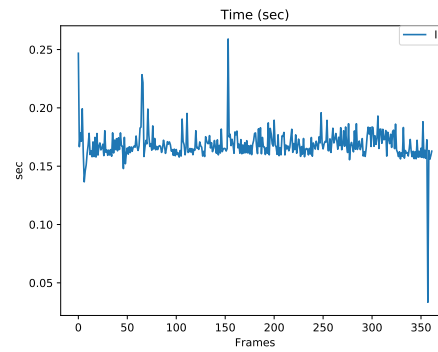
Dentro de los resultados obtenidos **SSIM**, se puede apreciar que durante toda la ejecución se mantiene un nivel bajo de información compartida entre *template* y el *frame*, lo que representa una desventaja frente al enfoque aditivo. Al igual que con **NMI** después del *frame* 300, recibe información considerada errónea.

TABLA 5.2: Composición inversa

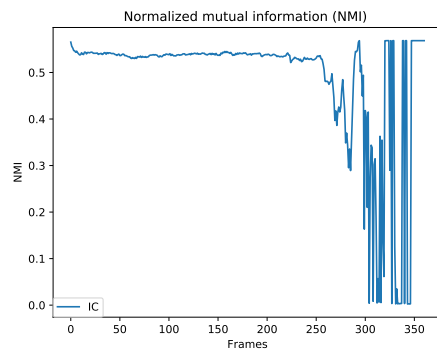
	SSD	NMI	SSIM	Time
Mínimo	0.0008	0.0024	-4.01875e+15	0.0333
Máximo	0.1641	0.5683	2.2038e+16	0.2590
Promedio	0.0289	0.4897	1.9325e+14	0.1684
Desviación estándar	0.0289	0.1330	1.6509e+15	0.0134



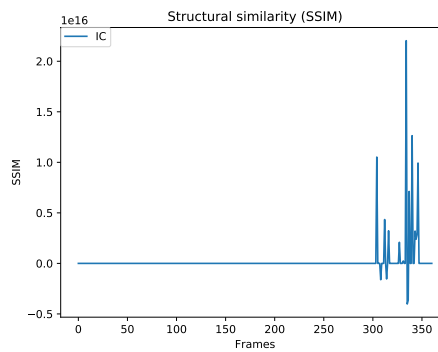
(a) Suma de error cuadrático (SSD)



(b) Tiempo de procesamiento (sec)



(c) Información mutua normalizada (NMI)



(d) Similitud estructural (SSIM)

FIGURA 5.4: Gráfica de métricas del composicional inverso, en a) Gráfica del error obtenido de la suma de la diferencia cuadrada entre el *template* y la imagen actual. b) El tiempo (segundos) utilizado para procesar cada frame. c) Gráfica de la información mutua normalizada. d) Gráfica del índice de similitud estructural.

5.1.3. Aprendizaje

En la Tabla 5.3 y en la Figura 5.5(a) se puede observar un comportamiento de acumulación de error progresivo desde el inicio hasta el *frame* 40, donde se observa un patrón de estabilización del error con pequeñas variaciones hasta el *frame* 220 aproximadamente. Sin embargo como podemos ver en la secuencia (c) de la Figura 5.5 el algoritmo solo sigue exitosamente el *template* hasta el frame 50, lo que demuestra que la métrica **SSD** no es la más adecuada para medir la estructura de una imagen la cual además muestra parámetros de error bastante grandes.

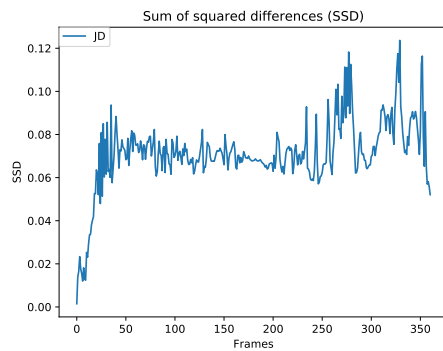
Respecto al tiempo este algoritmo es el más lento de los tres comparados, debido al proceso de entrenamiento necesario para generar la matriz *A*. Como se puede observar en la Tabla 5.3 el costo de entrenamiento es de 84.53 segundos, lo que lo posiciona muy por detrás de sus competidores. Sin embargo una vez entrenada la matriz *A*, este enfoque realiza las operaciones de seguimiento en los *frames* casi de manera instantánea.

NMI muestra una clara tendencia de descenso lo que expresa una gran pérdida de información, cosa que es inferible del hecho que después del frame 50 se deja de lograr la convergencia del algoritmo. Sin embargo en los primeros frames muestra unos valores mayores que los presentados por los algoritmos aditivo y composicional inverso, lo que se interpreta como una mayor tolerancia a los cambios de perspectiva.

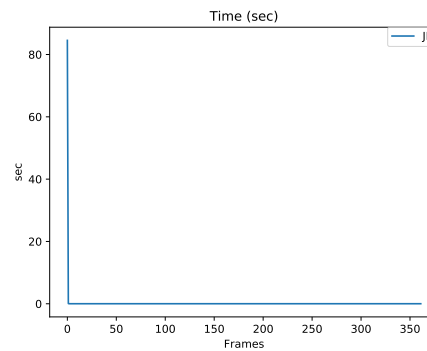
Mostrando una misma tendencia, la curva **SSIM** mostrada en la Figura 5.5 (d) es descendente. Esta misma mantiene valores bajos al momento de no converger el algoritmo, sin embargo estos valores no tienden a cero debido a la constancia de brillo y contraste.

TABLA 5.3: Aprendizaje

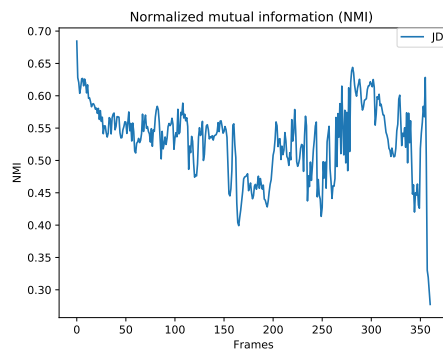
	SSD	NMI	SSIM	Time
Mínimo	0.0014	0.2773	0.0730	0.0126
Máximo	0.1236	0.6845	0.9056	84.5354
Promedio	0.0711	0.5313	0.4525	0.2479
Desviación estándar	0.0160	0.0568	0.0779	4.4361



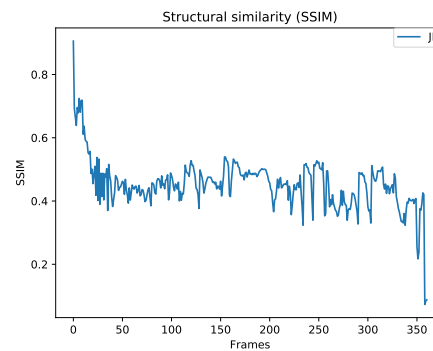
(a) Suma de error cuadrático (SSD)



(b) Tiempo de procesamiento (sec)



(c) Información mutua normalizada (NMI)



(d) Similitud estructural (SSIM)

FIGURA 5.5: Gráfica de métricas del algoritmo de aprendizaje, en a) Gráfica del error obtenido de la suma de la diferencia cuadrada entre el *template* y la imagen actual. b) El tiempo (segundos) utilizado para procesar cada frame. c) Gráfica de la información mutua normalizada. d) Gráfica del índice de similitud estructural.

5.1.4. Comparación imágenes de textura

Como se puede ver en la Figura 5.1 (a) las curvas de **SSD** son bastante diferentes entre el algoritmo de aprendizaje y los otros dos enfoques. El algoritmo aditivo muestra un margen de error muy pequeño a comparación de todo el ciclo del enfoque de aprendizaje, también se puede observar un comportamiento casi idéntico entre el aditivo y el algoritmo de composición inversa, donde este último incrementa su error súbitamente al no lograr converger en el frame 250.

Respecto al tiempo se puede observar una gran diferencia entre el algoritmo de aprendizaje y los otros dos propuestos. El enfoque más rápido es el composicional inverso, seguido por el algoritmo aditivo. El algoritmo de aprendizaje consume una gran cantidad de tiempo en la fase de entrenamiento, sin embargo una vez entrenado puede analizar cada *frame* con una velocidad mayor que los otros enfoques analizados.

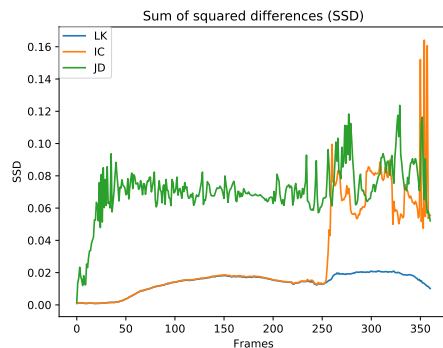
NMI muestra un comportamiento parecido al presentado por **SSD** donde el enfoque aditivo mantiene un espectro de valores sin muchas variaciones, de la misma manera el enfoque composicional inverso sigue un comportamiento parecido hasta el frame 250 donde se deja de lograr la convergencia. El enfoque de aprendizaje presenta un comportamiento superior en los primeros *frames* y después disminuye hasta que cae en un rango de fluctuaciones menores a pesar de no lograr la convergencia.

En los resultados de **SSIM** se puede apreciar como los tres algoritmos tienen una tendencia a cero, sin embargo cerca del frame 300 el enfoque por composición inversa realiza cambios bruscos del rango de valores. Estos cambios son muy pequeños debido a que la escala de los valores es $1e16$, lo que significa que ninguno de los tres algoritmos comparados logran obtener un buen resultado en esta métrica.

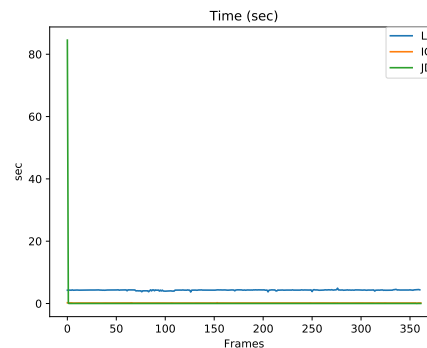
5.2. Imágenes RGB-D

5.2.1. Aditivo

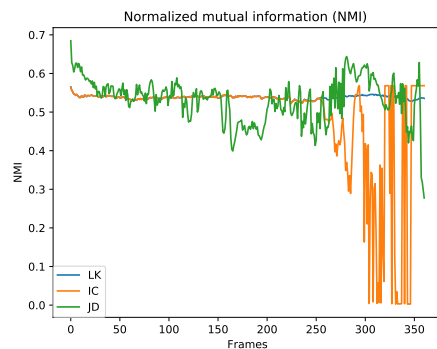
Como se puede ver en la Figura 5.7 (a), en **SSD** se puede apreciar una tendencia de crecimiento en toda la secuencia. Sin embargo, el error obtenido en la secuencia es prácticamente nulo.



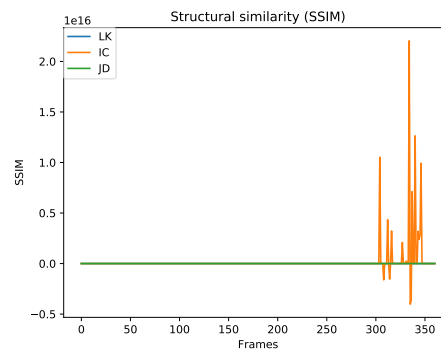
(a) Suma de error cuadrático (SSD)



(b) Tiempo de procesamiento (sec)



(c) Información mutua normalizada (NMI)



(d) Similitud estructural (SSIM)

FIGURA 5.6: Gráfica comparativa de métricas de imágenes de textura, en a) Gráfica del error obtenido de la suma de la diferencia cuadrada entre el *template* y la imagen actual. b) El tiempo (segundos) utilizado para procesar cada frame. c) Gráfica de la información mutua normalizada. d) Gráfica del índice de similitud estructural. En azul enfoque aditivo, en amarillo composicional inverso. y en verde aprendizaje .

Al principio se puede observar una gran desviación estándar en los tiempos de procesado de los *frames*, hasta el frame 75 donde se establece una varianza casi inexistente de tiempo entre los *frames* en comparación con los anteriores al mencionado.

NMI muestra un comportamiento de descenso en toda la secuencia, sin embargo entre los *frames* 10 al 60 se logró una relativa estabilización de la curva. Aproximadamente en el frame 80 se genera una pequeña discrepancia entre el *template* y el *frame* lo que genera un descenso mucho mayor en la cantidad de información que comparten estas variables.

Algo que habla de la calidad de la composición de la imagen es el valor inicial 1 de la **SSIM** en comparación de las imágenes de textura. Sin embargo en toda la secuencia se puede ver una tendencia de descenso hasta el frame 100 donde el rango de valores se posiciona entre 0.62 y 0.71 lo cual corresponde a una buena similitud.

TABLA 5.4: Enfoque aditivo

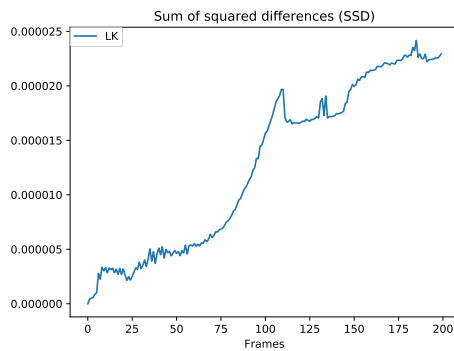
	SSD	NMI	SSIM	Time
Mínimo	2.0200e-16	0.4335	0.6273	0.7139
Máximo	2.4174e-05	0.7183	0.9999	0.8678
Promedio	1.2680e-05	0.5222	0.7256	0.7611
Desviación estándar	7.6817e-06	0.0824	0.0767	0.0224

5.2.2. Composicional inverso

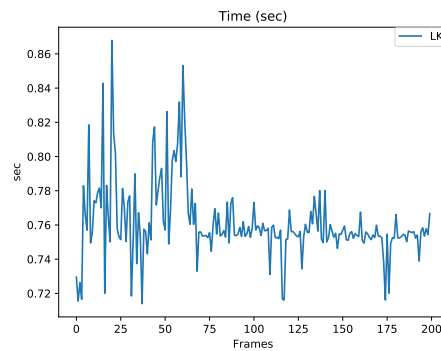
El comportamiento de **SSD** en la Figura 5.8 muestra un descenso en la eficiencia en comparación con su desempeño en imágenes de textura, se pueden observar acumulaciones de error a través del tiempo hasta que en el frame 80 la convergencia deja de ocurrir y se dispara el índice de error.

Por otro lado, los tiempos de procesamiento que presenta en estas imágenes son mucho menores comparados con el enfoque aditivo. Presentando velocidades de hasta 0.01 segundos en los *frames* que no logra converger y alcanzando el límite en los 0.15 segundos como máximo.

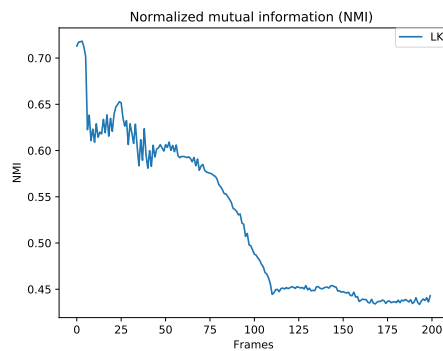
NMI muestra un descenso de la cantidad de información compartida desde el inicio de la secuencia hasta el *frame 10*, en los *frames* siguientes se puede observar fluctuaciones de los valores de la cantidad de información compartida hasta el frame 75 donde se observa desplomarse al momento de no lograr la convergencia en el resto de la secuencia.



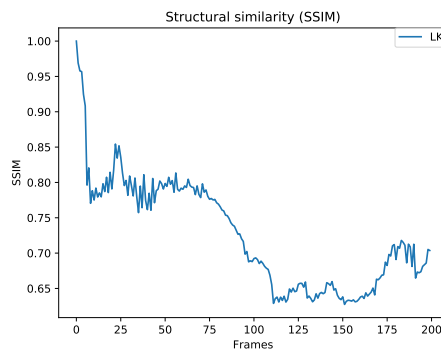
(a) Suma de error cuadrático (SSD)



(b) Tiempo de procesamiento (sec)



(c) Información mutua normalizada (NMI)



(d) Similitud estructural (SSIM)

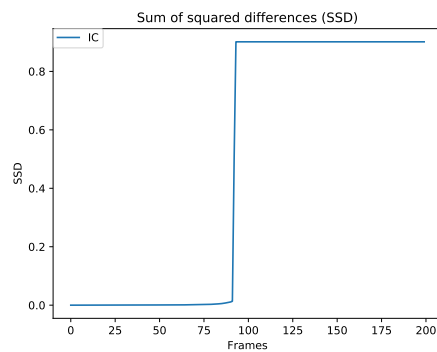
FIGURA 5.7: Gráfica de métricas del algoritmo aditivo en imágenes de profundidad, en a) Gráfica del error obtenido de la suma de la diferencia cuadrada entre el *template* y la imagen actual. b) El tiempo (segundos) utilizado para procesar cada frame. c) Gráfica de la información mutua normalizada. d) Gráfica del índice de similitud estructural.

Por parte de **SSIM** también se puede observar un decremento en la similitud hasta el *frame 10*, posteriormente se logra una estabilización relativa de la información hasta el frame 75 donde la diferencia incrementa y disminuye aún más la calidad de la información compartida.

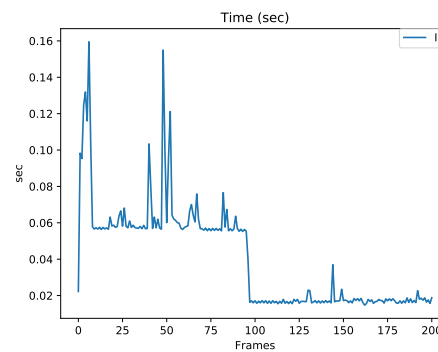
TABLA 5.5: Compisición inversa

	SSD	NMI	SSIM	Time
Mínimo	7.7373e-07	-5.6898e-06	0.2809	0.0147
Máximo	0.9012	0.7354	0.9464	0.1594
Promedio	0.4854	0.2046	nan	0.0406
Desviación estándar	0.4474	0.2227	nan	0.0285

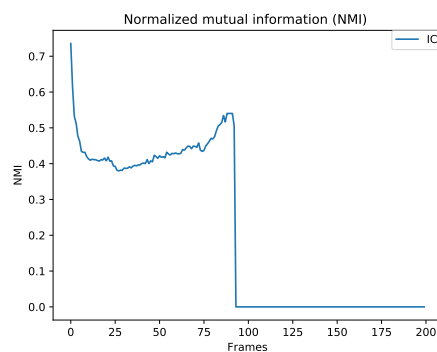
TABLA 5.6: Conclusiones



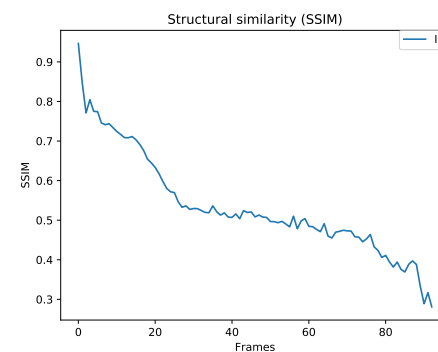
(a) Suma de error cuadrático (SSD)



(b) Tiempo de procesamiento (sec)



(c) Información mutua normalizada (NMI)



(d) Similitud estructural (SSIM)

FIGURA 5.8: Gráfica de métricas del composicional inverso en imágenes de profundidad, en a) Gráfica del error obtenido de la suma de la diferencia cuadrada entre el *template* y la imagen actual. b) El tiempo (segundos) utilizado para procesar cada frame. c) Gráfica de la información mutua normalizada. d) Gráfica del índice de similitud estructural.

5.2.3. Aprendizaje

En la Figura 5.5 (a) se puede observar error bajo constante en los primeros 100 *frames*, sin embargo en la Figura 5.2 (c) se puede observar como a partir del frame 40 ya se tiene un desfase considerable entre el *template* con el *frame*. Por otra parte los resultados obtenidos muestran un desempeño notablemente alto con un error máximo de 0.067 y un mínimo de 4.646E-06.

En el tiempo de procesamiento, se puede observar como este algoritmo es el más lento de los comparados en este trabajo debido al tiempo necesario para su entrenamiento. Sin embargo cabe destacar que una vez realizado el entrenamiento, el procesado de cada frame es casi instantáneo.

Por parte de **NMI** se observa una disminución constante de la cantidad de información mutua hasta el *frame* 50 donde después se estabiliza hasta el *frame* 90 con un índice de 0.33, para después continuar disminuyendo, lo cual representa una gran pérdida de información mutua.

Los resultados obtenidos por **SSIM** son más alentadores con una varianza de 1 en torno a 0.8, lo cual representa una buena similitud. Este valor no es apreciable claramente en la imagen debido a sus componentes de profundidad. Sin embargo, en la Figura 5.2 (c) se puede observar como el contorno del *template* se encuentra un poco alejado del ground truth marcado en los *frames* posteriores al 40

TABLA 5.7: Aprendizaje

	SSD	NMI	SSIM	Time
Mínimo	4.6468e-06	0.5238	0.1966	0.0102
Máximo	0.0674	0.9677	0.6075	55.6426
Promedio	0.0007	0.8145	0.3471	0.2892
Desviación estándar	0.0050	0.0664	0.0569	3.9238

TABLA 5.8: Conclusiones

5.2.4. Comparación de imágenes de profundidad

Como se puede observar en la Figura 5.10 **SSD** permanece en niveles bajos durante toda la secuencia en el algoritmo aditivo, seguido por el enfoque de aprendizaje y por último el

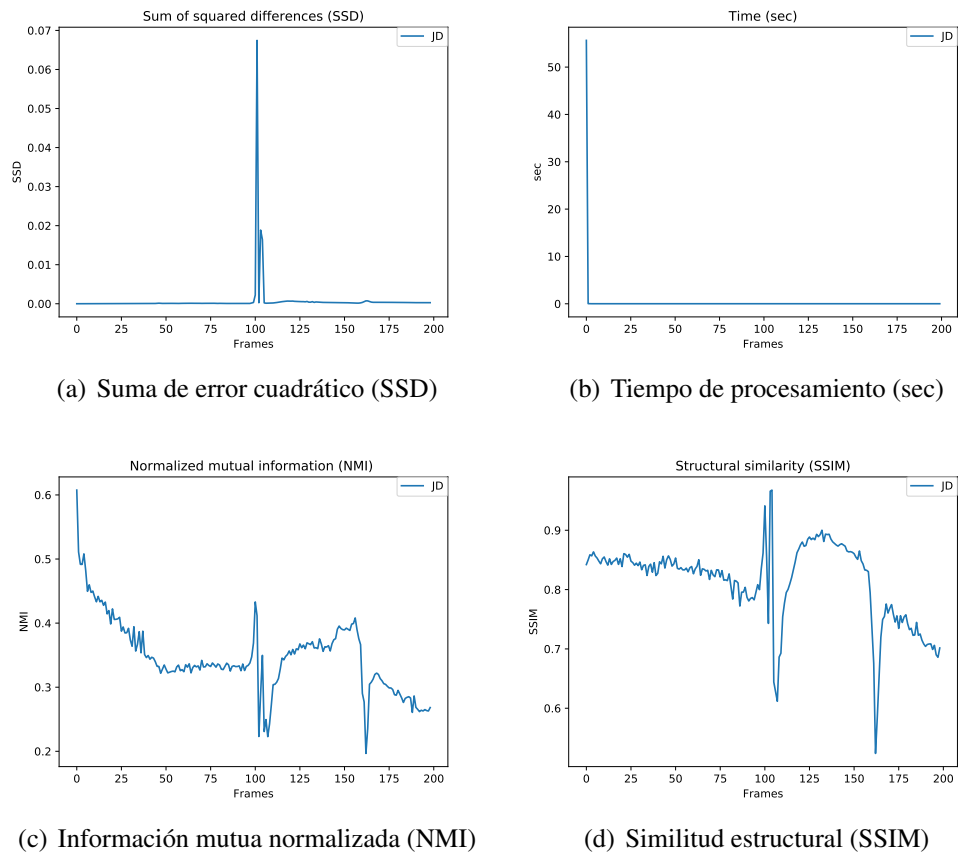


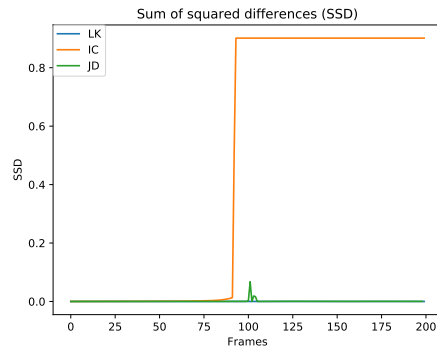
FIGURA 5.9: Gráfica de métricas del algoritmo de aprendizaje en imágenes de profundidad, en a) Gráfica del error obtenido de la suma de la diferencia cuadrada entre el *template* y la imagen actual. b) El tiempo (segundos) utilizado para procesar cada frame. c) Gráfica de la información mutua normalizada. d) Gráfica del índice de similitud estructural.

algoritmo de composición inversa presenta un nivel de error muy superior a partir del frame 85 aproximadamente, concordando con el *frame* donde se deja de lograr la convergencia.

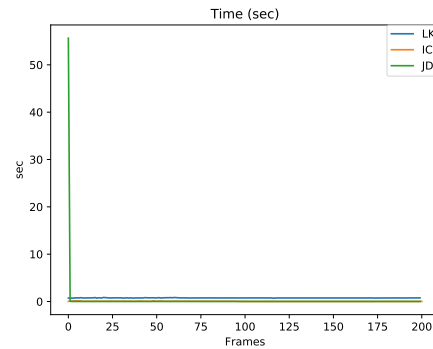
Respecto al tiempo, el algoritmo de composicion inversa tiene una clara ventaja sobre el enfoque aditivo y una aún más marcada sobre el de aprendizaje, mostrando tiempos entre 0.0147 y 0.159 segundos.

La calidad del seguimiento es más notable al comparar la cantidad de información compartida. Ésta se obtiene calculando **NMI**, donde se puede ver una consistencia mayor de los valores, así como valores más altos por parte del enfoque aditivo, seguido por el algoritmo de composicion inversa el cual logra buenos valores hasta el *frame* 85 donde deja de lograr la convergencia y por último el algoritmo de aprendizaje.

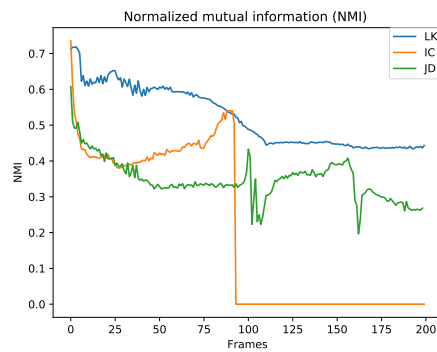
Por la parte de **SSIM** se puede ver un comportamiento similar donde el enfoque aditivo mantiene valores con poca varianza, el composicional inverso pierde información para después recuperarse un poco y por último dejar de converger en el *frame* 85. Aunque no se puede apreciar en la secuencia de imágenes, el enfoque de aprendizaje mantiene una parte de la información del *template* ya que mantiene valores por encima de 0.2 hasta el final de la secuencia.



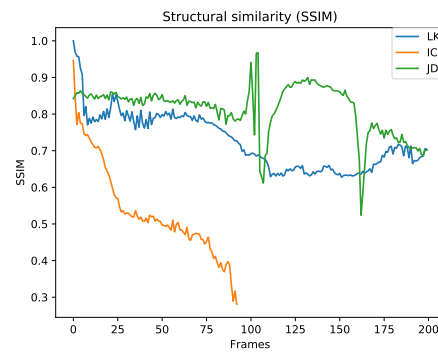
(a) Suma de error cuadrático (SSD)



(b) Tiempo de procesamiento (sec)



(c) Información mutua normalizada (NMI)



(d) Similitud estructural (SSIM)

FIGURA 5.10: Gráfica comparativa de métricas de imágenes de profundidad, en a) Gráfica del error obtenido de la suma de la diferencia cuadrada entre el *template* y la imagen actual. b) El tiempo (segundos) utilizado para procesar cada frame. c) Gráfica de la información mutua normalizada. d) Gráfica del índice de similitud estructural. En azul enfoque aditivo, en amarillo composicional inverso. y en verde aprendizaje .

Capítulo VI

Conclusiones y trabajos futuros

6.1. Conclusiones

El trabajo desarrollado muestra la implementación y la comparación de tres algoritmos de seguimiento de objetos para tres tipos de aproximaciones: enfoque aditivo, composicional inverso y aprendizaje. Para la primera aproximación se estudió el algoritmo de Lucas-Kanade [26] debido a su importancia dentro de los seguidores de objetivos y para establecer una línea base de trabajo. El segundo algoritmo seleccionado fue el de Baker-Matthews [38] ya que presenta una mejora con respecto a Lukas-Kanade en términos de eficiencia computacional. Finalmente, se implementa y estudia el algoritmo propuesto por Jurie-Dhome [86] que se basa en el composicional inverso de Baker-Matthews pero cuya implementación utiliza un método de aprendizaje de hiperplanos (regresores). Para la comparación de los algoritmos se definió un conjunto de métricas basadas en cambios de apariencia, estructura y ganancia de información. Se desarrollaron dos tipos de experimentos: uno basado en imágenes (imágenes RGB) y otro basado en superficie (imágenes RGB-D). Para las imágenes RGB-D se utilizó una base de datos generadas para esta tesis. Para poder obtener secuencias de movimientos de las imágenes RGB-D se utilizó Blender.

Para lograr el término de esta tesis se realizó un estudio exhaustivo del estado del arte donde se pudieron determinar aquellos métodos y algoritmos que podían ser útiles para validar los diferentes objetivos de este trabajo. Al final de este estudio se concluyó que existen tres tipos de enfoques para seguir objetos rígidos: aditivo, composicional inverso y aprendizaje.

Uno de los primeros problemas encontrados fue como evaluar secuencias de imágenes RGB-D con

diferentes poses y orientaciones. Se optó por el desarrollo de imágenes RGB-D sintéticas generadas por Blender y a partir de un conjunto de imágenes 3D obtenidas de diferentes alumnos y profesores del magíster. Como consecuencia de esto, se estudió que era posible utilizar este tipo de imágenes ya que representan una buena aproximación de una imagen obtenida directamente de una cámara RGB-D.

Se evaluó la efectividad de diferentes métricas con los algoritmos propuestos por Lukas-Kanade, Baker-Matthews y Jurie-Dhome utilizando imágenes RGB y RGB-D. Para esto se consideraron aspectos de desempeño que involucran el costo computacional y la eficiencia de cada algoritmo de seguimiento. Como resultado de esto se utilizó el tiempo de procesamiento como medida de costo computacional y las métricas SSD, NMI y SSMI como medidas de eficiencia.

Se validó cada uno de los algoritmos propuestos, donde se pudo concluir que el algoritmo de Lucas-Kanade, el más antiguo y más lento debido a su enfoque aditivo, también es el más preciso y confiable para realizar el seguimiento de objetos en imágenes de textura y de profundidad. Por otro lado, el algoritmo de composición inversa ofrece una velocidad superior y una precisión que varía dependiendo de las transformaciones que sufra el template. Como se puede ver en la sección de resultados experimentales, el desempeño en las imágenes de textura es notablemente bueno, pero después de varias transformaciones deja de converger. En cambio, con imágenes de profundidad tiene un desempeño bastante precario por lo que se puede concluir que su mejor desempeño se logra en secuencias de imágenes de textura con transformaciones menores en el template. El enfoque por aprendizaje muestra una eficiencia menor que el enfoque compositivo y una velocidad de ejecución casi instantánea una vez entrenado, este proceso es muy costoso en términos de tiempo por lo que se concluye que podría ser una mejor opción en secuencias de imágenes largas para compensar el tiempo de entrenamiento.

A partir de lo inferido anteriormente y respondiendo a la hipótesis de trabajo propuesta en la tesis, se puede decir que el algoritmo de composición inversa no es una alternativa viable para realizar el seguimiento en imágenes RGB-D. Sin embargo, el algoritmo aditivo se ve como una buena alternativa para el seguimientos en imágenes RGB-D y que en los experimentos destacó sobre el resto. Por otro lado, se puede agregar que a pesar que el algoritmo de aprendizaje no tuvo el desempeño esperado esto se puede atribuir a que el proceso de entrenamiento por Jurie-Dhome no se adapta completamente al problema propuesto. Por lo que se sugiere modificar el método de aprendizaje para ser más competitivo que el actual. En resumen, se ha logrado evaluar un conjunto de algoritmos para seguimientos de objetos (caras) en imágenes RGB-D donde Lucas-Kanade obtuvo el mejor rendimiento a nivel de eficiencia pero a un costo elevado de tiempo. Los otros dos algoritmos presentaron mejoras significativas en tiempos de ejecución pero con una baja eficiencia

a nivel seguimiento global. Si bien es cierto, la hipótesis de trabajo no se cumple en su totalidad se cree que existe un camino por recorrer sobre todo en técnicas de aprendizaje.

6.2. Trabajos futuros

Con base en los resultados de este trabajo se realizarán pruebas de seguimiento con el algoritmo aditivo en nuevas bases de datos de mayor complejidad y se espera adaptar de mejor forma este algoritmo para el seguimiento de objetos en imágenes RGB-D adicionalmente se probarán otras variantes de algoritmos con el fin de mejorar la eficiencia computacional y la precisión de los métodos mostrados.

Despues de analizar y realizar el proceso de entrenamiento en imágenes 2D del algoritmo con enfoque de aprendizaje, se propone realizar el entrenamiento utilizando directamente la cámara de Blender para capturar diferentes partes del modelo 3D para generar la diferencias de intensidades y los parámetros de movimiento. Se cree que esto aportará la información necesaria para realizar un correcto seguimiento de las imágenes de profundidad. Otro enfoque de mejora al algoritmo es utilizar algún tipo de descriptor de las intensidades o profundidades de la imagen en lugar de utilizarlas directamente para el entrenamiento. Un enfoque que se considera podría mejorar considerablemente los resultados es utilizar alguna técnica de *machine learning* como maquinas de soporte vectorial o redes neuronales profundas en vez de realizar el proceso de entrenamiento manual.

Apéndice A

Expansión de Taylor del algoritmo de Lukas-Kanade

$$\sum_x [I(w(x; p) + \Delta p) - T(x)]^2 \quad (\text{A.1})$$

$$\sum_x [I(w(x; p) + (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})(w(x; p))) + \Delta p - T(x)]^2 \quad (\text{A.2})$$

$$\sum_x [I(w(x; p) + \nabla I(w(x; p)) + \Delta p - T(x)]^2 \quad (\text{A.3})$$

$$\sum_x [I(w(x; p) + \nabla I(\frac{\partial W_x}{\partial p}, \frac{\partial W_y}{\partial p}) + \Delta p - T(x)]^2 \quad (\text{A.4})$$

$$\sum_x [I(w(x; p) + \nabla I \frac{\partial W}{\partial p} + \Delta p - T(x)]^2 \quad (\text{A.5})$$

Apéndice B

Publicaciones

Los resultados generados en este trabajo fueron documentados en los siguientes trabajos:

- Quiñonez, Y., Ramirez, M., Lizarraga, C., Tostado, I., and Bekios, J. (2015). Autonomous Robot Navigation Based on Pattern Recognition Techniques and Artificial Neural Networks (pp. 320–329). *https : //doi.org/10,1007/978 – 3 – 319 – 18833 – 1*₃₄
- Ramirez, L., Bekios, J. (2017). Performance evaluation of direct methods for tracking the human face in synthetic RGB-D images. Workshop on data and knowledge engineering (WDKE).

BIBLIOGRAFÍA

- [1] I. K. Sethi and R. Jain, “Finding Trajectories of Feature Points in a Monocular Image Sequence,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 1, pp. 56–73, 1987.
- [2] V. Salari and I. K. Sethi, “Feature Point Correspondence in the Presence of Occlusion,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 87–91, 1990.
- [3] K. Rangarajan and M. Shah, “Establishing motion correspondence,” *CVGIP: Image Understanding*, vol. 54, no. 1, pp. 56–73, 1991.
- [4] S. S. Intille, J. W. Davis, and A. F. Bobick, “Real-time closed-world tracking,” in *Conference on Computer Vision and Pattern Recognition*, 1997, pp. 697–703. [Online]. Available: <http://ieeexplore.ieee.org/xpls/abs{ }all.jsp?arnumber=609402>
- [5] C. J. Veenman, M. J. T. Reinders, and E. Backer, “Resolving motion correspondence for densely moving points,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 54–72, 2001.
- [6] K. Shafique and M. Shah, “A noniterative greedy algorithm for multiframe point correspondence,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 51–65, Jan 2005.
- [7] M. Isard and A. Blake, “Condensation - conditional density propagation for visual tracking,” *International journal of computer vision*, vol. 29, no. 1, pp. 5–28, 1998. [Online]. Available: <http://link.springer.com/article/10.1023/A:1008078328650>
- [8] N. Vaswani, a. Roy Chowdhury, and R. Chellappa, “Activity recognition using the dynamics of the configuration of interacting objects,” *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, pp. II–633–40, 2003. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1211526>

- [9] S. Zhou, R. Chellappa, and B. Moghaddam, "Adaptive visual tracking and recognition using particle filters," in *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, vol. 2, July 2003, pp. II-349-52 vol.2.
- [10] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *International Journal of Computer Vision*, vol. 3, no. 3, pp. 209-238, 1989.
- [11] D. Huttenlocher, J. Noh, and W. Rucklidge, "Tracking non-rigid objects in complex scenes," pp. 93-101, 1993.
- [12] B. Li, R. Chellappa, Q. Zheng, and S. Z. Der, "Model-based temporal object verification using video," *IEEE Transactions on Image Processing*, vol. 10, no. 6, pp. 897-908, 2001.
- [13] J. Kang, I. Cohen, and G. Medioni, "Object reacquisition using invariant appearance model," in *Proceedings - International Conference on Pattern Recognition*, vol. 4, 2004, pp. 759-762.
- [14] K. Sato and J. Aggarwal, "Temporal spatio-velocity transform and its application to tracking and interaction," *Computer Vision and Image Understanding*, vol. 96, no. 2, pp. 100 - 128, 2004, special Issue on Event Detection in Video. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1077314204000700>
- [15] D. Terzopoulos and R. Szeliski, "Active vision," A. Blake and A. Yuille, Eds. Cambridge, MA, USA: MIT Press, 1993, ch. Tracking with Kalman Snakes, pp. 3-20. [Online]. Available: <http://dl.acm.org/citation.cfm?id=163213.163214>
- [16] J. MacCormick and a. Blake, "A probabilistic exclusion principle for tracking multiple objects," *International Journal of Computer Vision*, vol. 39, no. 1, pp. 57-71, 2000.
- [17] Y. C. Y. Chen, Y. R. Y. Rui, and T. Huang, "JPDAF based HMM for real-time contour tracking," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. 543-550, 2001.
- [18] M. Bertalmio, G. Sapiro, and G. Randall, "Morphing active contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 7, pp. 733-737, 2000.
- [19] A. R. Mansouri, "Region tracking via level set PDEs without motion computation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 947-961, 2002.
- [20] D. Cremers and C. Schnorr, "Statistical shape knowledge in variational motion segmentation," *Image and Vision Computing*, vol. 21, no. 1, pp. 77-86, 2003.

- [21] A. Yilmaz, X. Li, and M. Shah, "Contour-based object tracking with occlusion handling in video acquired using mobile cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1531–1536, 2004.
- [22] R. Ronfard, "Region-based strategies for active contour models," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 229–251, 1994.
- [23] F. Jurie, M. Dhome, and Others, "Real Time Robust Template Matching." *Bmvc*, pp. 1–10, 2002.
- [24] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, 2003.
- [25] P. Fieguth and D. Terzopoulos, "Color-based tracking of heads and other mobile objects at video frame rates," *Computer Vision and Pattern . . .*, pp. 21–27, 1997. [Online]. Available: <http://ieeexplore.ieee.org/xpls/abs{ }all.jsp?arnumber=609292>
- [26] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," pp. 674–679, 1981.
- [27] S. Holzer, M. Pollefeys, and S. Ilic, "Online learning of linear predictors for real-time tracking," *Computer Vision–ECCV . . .*, pp. 470–483, 2012. [Online]. Available: <http://link.springer.com/chapter/10.1007/978-3-642-33718-5{ }34>
- [28] M. Ozuysal, P. Fua, and V. Lepetit, "Fast Keypoint Recognition in Ten Lines of Code," *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, 2007.
- [29] S. Holzer, S. Hinterstoisser, S. Ilic, and N. Navab, "Distance transform templates for object detection and pose estimation," *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pp. 1177–1184, 2009.
- [30] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit, "Online learning of patch perspective rectification for efficient object detection," *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2008.
- [31] S. Hinterstoisser, O. Kutter, N. Navab, P. Fua, and V. Lepetit, "Real-time learning of accurate patch rectification," *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*, pp. 2945–2952, 2009.
- [32] S. Hinterstoisser, S. Ilic, N. Navab, P. Fua, and V. Lepetit, "Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects," 2010.

- [33] E. Malis, “Improving vision-based control using efficient second-order minimization techniques,” *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 2, pp. 1843–1848 Vol.2, 2004.
- [34] E. Malis and S. Benhimane, “Homography-based 2D Visual Tracking and Servoing,” *The International Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, 2007.
- [35] G. D. Hager and P. N. Belhumeur, “Efficient region tracking with parametric models of geometry and illumination,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025–1039, 1998.
- [36] H.-Y. Shum and R. Szeliski, “Construction of panoramic mosaics with global and local alignment,” *International Journal of Computer Vision*, vol. 36, no. 2, pp. 101–130, 2000. [Online]. Available: <http://research.microsoft.com/apps/pubs/default.aspx?id=75614>
- [37] M. La Cascia, S. Sclaroff, V. Athitsos, M. L. Cascia, and M. LaCascia, “Fast, Reliable Head Tracking under Varying Illumination: An Approach Based on Registration of Textured-Mapped {3D} Models,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 322–336, 2000.
- [38] S. Baker and I. Matthews, “Equivalence and efficiency of image alignment algorithms,” *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. 243–245, 2001.
- [39] F. Jurie, M. Dhome, and Others, “Real Time Robust Template Matching.” *Bmvc*, pp. 1–10, 2002.
- [40] C. Gräßl, T. Zinßer, and H. Niemann, *Illumination Insensitive Template Matching with Hyperplanes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 273–280. [Online]. Available: https://doi.org/10.1007/978-3-540-45243-0_{_}36
- [41] C. Xu and L. Cheng, “Efficient hand pose estimation from a single depth image,” *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3456–3462, 2013.
- [42] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky, “Hough forests for object detection, tracking, and action recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 11, pp. 2188–2202, 2011.
- [43] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake, “Efficient Human Pose Estimation from Single Depth Images,” in *Decision Forests for Computer Vision and Medical*

- Image Analysis*. London: Springer London, 2013, pp. 175–192. [Online]. Available: <http://link.springer.com/10.1007/978-1-4471-4929-3{ }13>
- [44] C. Choi and H. I. Christensen, “RGB-D object tracking: A particle filter approach on GPU,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 1084–1091, 2013.
- [45] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [46] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” *Proceedings of the Alvey Vision Conference 1988*, pp. 147–151, 1988. [Online]. Available: <http://www.bmva.org/bmvc/1988/avc-88-023.html>
- [47] C. T. Jianbo Shi, “Good Features to Track.pdf,” *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593 – 600, 1994.
- [48] H. P. Moravec, “Visual Mapping by a Robot Rover,” in *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1*, ser. IJCAI’79. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1979, pp. 598–600. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1624861.1624997>
- [49] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space analysis,” *IEEE Transactions on pattern analysis and machine*, vol. 24, no. 5, pp. 603–619, 2002. [Online]. Available: <http://doi.ieeecomputersociety.org/10.1109/34.1000236>
- [50] Z. Wu and R. Leahy, “An optimal graph theoretic approach to data clustering: theory and its application to image segmentation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, no. 11, pp. 1101–1113, 1993. [Online]. Available: <http://dx.doi.org/10.1109/34.244673>
- [51] J. Shi and J. Malik, “Normalized Cuts and Image Segmentation Normalized Cuts and Image Segmentation Normalized Cuts and Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000. [Online]. Available: <http://repository.upenn.edu/cis{ }papers{ }5Cnhttp://dx.doi.org/10.1109/34.868688{ }5Cnhttp://repository.upenn.edu/cis{ }papers/107>
- [52] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [53] V. Caselles, R. Kimmel, and G. Sapiro, “Geodesic Active Contours,” *International Journal of Computer Vision*, vol. 22, no. 1, pp. 61–79, 1997.

- [54] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfnder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997. [Online]. Available: <http://ieeexplore.ieee.org/xpls/abs{ }all.jsp?arnumber=598236>
- [55] C.-B. Ng, Y.-H. Tay, and B.-M. Goi, "A review of facial gender recognition," *Pattern Analysis and Applications*, vol. 18, no. 4, pp. 739–755, 2015. [Online]. Available: <http://link.springer.com/10.1007/s10044-015-0499-6>
- [56] H. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998. [Online]. Available: <http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=655647>
- [57] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, 2005.
- [58] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, "Random Forests for Real Time 3D Face Analysis," *International Journal of Computer Vision*, vol. 101, no. 3, pp. 437–458, 2013.
- [59] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," pp. 555–562, 1998.
- [60] T. J. Broida and R. Chellappa, "Estimation of Object Motion Parameters from Noisy Images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 1, pp. 90–99, 1986. [Online]. Available: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4767755>
- [61] S. Avidan, "Support Vector Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1064–1072, 2004.
- [62] M. Gleicher, "Projective registration with difference decomposition," pp. 331–337, 1997.
- [63] T. Cootes, G. Edwards, and C. Taylor, "Active appearance models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001. [Online]. Available: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=927467>
- [64] A. Yilmaz, O. Javed, and M. Shah, "Object tracking," *ACM Computing Surveys*, vol. 38, no. 4, pp. 13–es, 2006.

- [65] D. H. Ballard and C. M. Brown, "Computer vision," *en.scientificcommons.org*, 1982. [Online]. Available: <http://en.scientificcommons.org/9088225{%}5Cnpapers://c576dbe6-86f4-4d20-9369-246bcb85dd2d/Paper/p6517>
- [66] S. C. Zhu, T. S. Lee, and A. L. Yuille, "Region competition: unifying snakes, region growing, energy/bayes/mdl for multi-band image segmentation," in *Proceedings of IEEE International Conference on Computer Vision*, Jun 1995, pp. 416–423.
- [67] N. Paragios and R. Deriche, "Geodesic active regions and level set methods for supervised texture segmentation," *International Journal of Computer Vision*, vol. 46, no. 3, pp. 223–247, Feb 2002. [Online]. Available: <https://doi.org/10.1023/A:1014080923068>
- [68] A. Elgammal, R. Duraiswami, D. Harwood, and L. S. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1151–1163, Jul 2002.
- [69] G. Edwards, C. Taylor, and T. Cootes, "Interpreting face images using active appearance models," *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 300–305, 1998. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=670965>
- [70] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37 – 52, 1987, proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0169743987800849>
- [71] P. Comon, "Independent component analysis, a new concept?" *Signal Processing*, vol. 36, no. 3, pp. 287 – 314, 1994, higher Order Statistics. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0165168494900299>
- [72] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696–710, Jul 1997.
- [73] M. J. Black and A. D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," *International Journal of Computer Vision*, vol. 26, no. 1, pp. 63–84, Jan 1998. [Online]. Available: <https://doi.org/10.1023/A:1007939232436>
- [74] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

- [75] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.
- [76] T. Yamasaki, “Histogram of Oriented Gradients (HOG),” *The Journal of the Institute of Image Information and Television Engineers*, vol. 64, no. 3, pp. 322–329, 2010. [Online]. Available: <http://ci.nii.ac.jp/naid/10025988610/en/>
- [77] S. G. Mallat, “A Theory for Multiresolution Signal Decomposition: The Wavelet Representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674–693, 1989.
- [78] P. Langley, “Selection of Relevant Features in Machine Learning,” in *In Proceedings of the AAAI Fall Symposium on Relevance*, 1994, pp. 140–144.
- [79] R. Szeliski, “Computer Vision : Algorithms and Applications,” *Computer*, vol. 5, p. 832, 2010.
- [80] A. Dame and E. Marchand, “Second-order optimization of mutual information for real-time image registration,” *IEEE Transactions on Image Processing*, vol. 21, no. 9, pp. 1–14, 2012.
- [81] D. J. Hawkes, C. Studholme, D. L. G. Hill, and D. J. Hawkes, “An overlap invariant entropy measure of 3D medical image alignment,” *Pattern Recognition*, vol. 32, no. 1, pp. 71–86, 1999. [Online]. Available: <http://linkinghub.elsevier.com/retrieve/pii/S0031320398000910>{%}5Cn<http://www.sciencedirect.com/science/article/B6V14-3W83V9V-R/1/44f708e8fdb91c491ffcc992762ef514>{%}5Cnhttp://www.sciencedirect.com/science?{_}ob=ArticleURL{&{_}udi=B6V14-3W83V9V-R{&{_}user=18704{&{_}rdoc=1{&{_}fm
- [82] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.
- [83] S. I. Gass and M. C. Fu, Eds., *Hessian Matrix*. Boston, MA: Springer US, 2013, p. 694. [Online]. Available: [http://dx.doi.org/10.1007/978-1-4419-1153-7{\[_\]}200288](http://dx.doi.org/10.1007/978-1-4419-1153-7{[_]}200288)
- [84] M. A. Abidi, A. V. Gribok, and J. Paik, *Optimization Techniques in Computer Vision*, ser. Advances in Computer Vision and Pattern Recognition. Cham: Springer International Publishing, 2016. [Online]. Available: <http://link.springer.com/10.1007/978-3-319-46364-3>
- [85] S. Baker and I. Matthews, “Lucas-Kanade 20 Years On : A Unifying Framework,” vol. 56, no. 3, pp. 221–255, 2004.

- [86] F. Jurie and M. Dhome, “Hyperplane approximation for template matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 996–1000, 2002.
- [87] C. Vision, *Computer Vision*, 2009, vol. 3.
- [88] G. D. Hager and P. N. Belhumeur, “Real-Time Tracking of Image Regions with Changes in Geometry and Illumination,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 403–410, 1996.
- [89] D. systems. (2017) Sense 3d scanner. [Online]. Available: <https://www.3dsystems.com/shop/sense>
- [90] J. Bekios. (2017) mail. [Online]. Available: juan.bekios@ucn.c
- [91] D.-J. Kroon. (2009) Lucas kanade affine template tracking. [Online]. Available: <https://www.mathworks.com/matlabcentral/fileexchange/24677-lucas-kanade-affine-template-tracking?requestedDomain=www.mathworks.com>