VNIVERSIDAD NACIONAL AVTNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"RECONOCIMIENTO DE OBJETOS BASADO EN LA CORRESPONDENCIA ESTRUCTURAL DE CARACTERÍSTICAS LOCALES"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN CIENCIAS (COMPUTACIÓN)

P R E S E N T A:

WENDY ELIZABETH AGUILAR MARTÍNEZ

DIRECTOR DE TESIS: "YANN FRAUEL"

México, D.F. **2006.**

Índice general

Índice de figuras

Índice de Cuadros

Introducción

Una de las funciones primarias del sistema visual humano es el reconocimiento de objetos, una habilidad que permite relacionar los estímulos visuales en las retinas con el conocimiento del mundo. Por ejemplo, el reconocimiento de objetos permite usar el conocimiento de cómo luce una manzana para poder encontrarla en el supermercado, de cómo luce un tiburón para nadar y ponerse a salvo o de usar el conocimiento de los puntos de referencia de la colonia para encontrar el camino a casa. El reconocimiento nos permite comprender el contenido de las imágenes y sólo cuando asignamos una etiqueta a la imagen de un objeto es cuando lo podemos anexar a nuestra propia experiencia.

El reconocimiento de objetos que hacemos los humanos parece que lo realizamos sin mucho esfuerzo y mas aún, somos capaces de reconocer nuevas instancias de un objeto a partir del conocimiento de prototipos de éste o de conocimiento en general de cómo luce. Por ejemplo, el modelo interno de lo que sabemos de cómo luce un perro, es suficiente como para que reconozcamos a cualquier perro, aunque sea de otra raza o color que nunca hayamos visto antes, e incluso aunque el perro se encuentre en alguna posición (sentado, parado o corriendo, por ejemplo) distinta a los perros que conocemos. Más sorprendente aún es que podemos reconocer a los objetos aunque éstos se encuentren en escenas complicadas y aunque sólo se vean parcialmente debido a que otro objeto lo esta tapando. Ésto nos lleva a preguntarnos: ¿cómo se codifica la información visual de un objeto?, ¿qué información se recupera de una imagen para que podamos reconocerlo? y ¿cómo se compara ésta información con nuestro conocimiento ya aprendido y almacenado para poder reconocer el objeto?

Actualmente existe una gran variedad de propuestas para intentar resolver este problema (por ejemplo: [52], [55], [99], [22] y [15]), sin embargo aún no se ha encontrado el método que lo resuelva por completo debido a la gran complejidad de éste. En los últimos años la aproximación basada en características locales invariantes ha tenido una gran aceptación [61] [55]. Típicamente consiste en extraer características locales de manera independiente a la imagen del modelo y a la imagen de la escena, caracterizarlas por descriptores invariantes y finalmente realizar la correspondencia. Su éxito se debe en gran parte a que la representación local esta basada en la apariencia y con ésto se evita la necesidad de realizar una segmentación previa, los objetos de interés se pueden reconocer incluso si se encuentran parcialmente ocultos, son tolerantes a variaciones complejas en la apariencia de los objetos causadas por la variación del punto de vista y por las condiciones de iluminación gracias a

que usan aproximaciones basadas en transformaciones simples a una escala local y a que las mediciones tanto en la base de datos como en las imágenes de los objetos a reconocer se obtienen y se representan de la misma manera. A pesar de su éxito, la robustez y generalidad de estas aproximaciones esta limitada a la repetición de las características en ambas imágenes y a la dificultad de realizar una correspondencia correcta de éstas.

En este último punto es curioso notar que la mayoría de las propuestas basadas en características locales invariantes siguen un esquema general para realizar la fase de verificación, la cual consiste en que una vez que se han detectado y descrito los puntos característicos se les aplica algún algoritmo que permita obtener un primer conjunto de correspondencias candidatas (por lo general se usa una correlación), en seguida se les aplica algún método robusto como *RANSAC* [20] o *LMedS* (ajuste de mínimos cuadrados) para estimar la geometría epipolar del sistema de la cámara y rechazar aquellas correspondencias incompatibles con la geometría de ésta para finalmente realizar un proceso de correspondencia guiado que hace uso de la geometría estimada para encontrar más correspondencias.

Objetivo. El objetivo de esta tesis es proponer un algoritmo que resuelva el problema de correspondencia de características pero con una aproximación estructural basada en grafos. En esta gama del espectro existe una gran variedad de algoritmos, entre los cuales destacan los de correspondencia de grafos, los basados en la minimización de una distancia de edición y aquellos que encuentran el máximo *clique*. Desafortunadamente, los métodos basados en grafos tienden a ser NP-completos y no son prácticos a menos que el tamaño de los grafos sea pequeño. Esto lleva a la necesidad de contar con heurísticas para poder comparar los vértices, las aristas y calificar correspondencias de subgrafos. Un ejemplo típico en esta categoría es el algoritmo de *Softassign* [24], cuya complejidad es polinomial. Sin embargo, se ha visto experimentalmente que es muy sensible a *outliers* y que soporta grafos menores de 50 vértices para tiempos menores de 5 segundos.

Contribución. La contribución de esta tesis consiste en proponer e implementar un algoritmo que permita resolver el problema de correspondencia estructural de características locales independiente del detector de características, con grafos de mayor densidad de vértices (del orden de cientos), robusto a una gran cantidad de correspondencias iniciales erróneas, con complejidad polinomial y con tiempos de ejecución que permita considerar llevar a cabo todo el proceso de reconocimiento de objetos en tiempos del orden de segundos.

Organización de la tesis. La tesis se encuentra organizada en 6 capítulos. El capítulo 1 presenta una visión general de las diferentes aproximaciones que han surgido en el reconocimiento de objetos, poniendo en contexto la aproximación basada en características locales. El capítulo 2 resume los algoritmos más usados para la correspondencia de características haciendo énfasis en los algoritmos basados en grafos. En el capítulo 3 se presenta y analiza la complejidad del algoritmo (de fuerza bruta y optimizado) propuesto en esta tesis para realizar la correspondencia de características locales correspondiente a la fase de verificación del proceso del reconocimiento

de objetos. Este algoritmo esta basado en la transformación simultánea de los grafos correspondientes al modelo y a la escena. Los resultados de las pruebas realizadas se muestran en los capítulos 4 y 5. En el capítulo 4 se reportan los resultados de comparar el algoritmo propuesto contra el algoritmo de *Softassign* (de enfoque estructural) y contra el algoritmo *RANSACing Thin-Plate Splines* (de enfoque de alineación). Adicionalmente se reportan los tiempos obtenidos de dos implementaciones del algoritmo de fuerza bruta, uno escrito en *Matlab* (para fines de comparación con otros métodos) y otro escrito en *C* contra la implementación en *C* del algoritmo optimizado de *Transformación de Grafos*. Al final del capítulo se presentan los resultados de pruebas realizadas con objetos que están parcialmente ocultos y se documentan dos casos particulares en los que el algoritmo no funciona correctamente. El capítulo 5 presenta las pruebas de robustez a correspondencias iniciales erróneas aplicadas al algoritmo de *Transformación de Grafos*. El primer experimento consistió en introducir correspondencias erróneas de manera aleatoria y controlada. El segundo experimento consistió en diseñar 2 filtros que permitieran tener diversas cantidades de correspondencias erróneas que de manera natural están presentes en los datos. Finalmente, en el capítulo 6 se presentan las conclusiones.

Capítulo 1

Aproximaciones al reconocimiento de objetos

El reconocimiento de objetos es uno de los aspectos más importantes de la percepción visual, y que no se ha llegado a comprender aún. Para muchos de los sistemas de visión biológicos, el reconocimiento y clasificación de objetos es una actividad espontánea y natural. Los niños pequeños son capaces de reconocer inmediatamente y sin esfuerzo una gran variedad de objetos. En el caso de los animales, también exhiben un comportamiento de reconocimiento extraordinario. Por ejemplo, los insectos tales como las abejas, usan el reconocimiento visual para propósitos como la navegación para encontrar su casa o para identificar las formas de las flores [25].

En contraste, el reconocimiento de objetos está todavía muy lejano de las posibilidades de los sistemas artificiales, o de cualquier modelo de reconocimiento que se haya propuesto hasta el momento. El contraste entre el cerebro y una computadora en sus capacidades de realizar reconocimiento visual y clasificación, hace que el problema sea particularmente fascinante. El cerebro generaliza espontáneamente a partir de ejemplos visuales sin la necesidad de contar con reglas explícitas ni con ins-trucción laboriosa. Por ejemplo, intuitivamente podemos darnos cuenta de que un niño puede generalizar a partir de una cantidad limitada de vistas de un perro, es decir, a partir de esas vistas es capaz de reconocer una gran cantidad y diversidad de perros bajo diferentes vistas. En contraste, se pueden programar a las computadoras para reconocer objetos específicos con formas fijas y bien definidas, tales como partes de máquinas. Es considerablemente más dificil capturar en un sistema de reconocimiento por computadora la escencia de un perro, una casa o un árbol, que es el tipo de clasificación natural e inmediata para el sistema visual humano.

El reconocimiento visual de objetos no es un solo problema. Una de las razones por las cuales se han propuesto una gran diversidad de aproximaciones al problema es que existen varios caminos diferentes que nos llevan al reconocimiento visual de los objetos, los cuales se basan en diferentes fuentes de información. Nosotros normalmente reconocemos visualmente a un objeto (un carro, un rostro familiar o un caracter impreso) basándonos en su forma característica. También podemos utilizar

información visual, pero que no es de forma, tal como el color y la textura. El reconocimiento de un árbol de un cierto tipo está basado en algunos casos más en propiedades de textura, patrones de ramificaciones y color, que en una forma precisa en particular. Ciertos animales como los tigres o las jirafas se pueden también identificar algunas veces con base en los patrones de textura y color en vez de en su forma. Similarmente, varios tipos de materiales, así como diferentes tipos de escenas, tal como la escena de un lago o de un terreno montañoso, se pueden reconocer visualmente sin usar información precisa de su forma.

Los objetos también se pueden reconocer visualmente si usamos información acerca de su posición relativa a otros objetos. Por ejemplo, la perilla de una puerta puede no tener una forma estándar, y aún así se puede reconocer inmediatamente si nos basamos en su posición relativa a la puerta (contexto). Aún otra posibilidad es reconocer a los objetos basándonos en su movimiento característico. Por ejemplo, una mosca en un cuarto se podría percibir únicamente como una pequeña cosa negra, y sin embargo aún así la podríamos reconocer como una mosca con sólo observar su movimiento errático característico.

En todos los ejemplos mencionados, el reconocimiento se puede decir que está basado principalmente en información visual. También existen situaciones en las que el proceso de reconocimiento usa fuentes que no son principalmente visuales por naturaleza. Un ejemplo es el que tiene que ver con el conocimiento a priori, espectativas y continuidad temporal. Por ejemplo, uno podría reconocer un objeto blanco sobre el escritorio como un teléfono incluso cuando la imagen visual no contenga suficiente detalle para realizar un reconocimiento claro del objeto (debido por ejemplo a que se observó muy rápidamente o porque el nivel de iluminación era muy bajo). Finalmente, en algunos casos, el reconocimiento visual emplea procesos que podrían ser descritos como razonamiento, por ejemplo, el reconocer una cerca alrededor de un campo podría basarse en parte no sólo en la forma visual específica, sino en el razonamiento acerca de su uso potencial (experiencia).

1.1. Principales teorías del reconocimiento de objetos

En esta sección se presentan las principales teorías del reconocimiento de objetos.

1.1.1. Aproximación computacional de Marr

Marr [52] propuso que se forman tres representaciones visuales de complejidad incremental durante la percepción visual:

1. Bosquejo fundamental. Es la representación inicial que se forma durante el procesamiento perceptual, contiene información acerca de las características tales como bordes y contornos. Es usada para formar el bosquejo 2 1/2 D.

2. Bosquejo 2 1/2 D. Es la representación centrada en el observador, la cual incluye información acerca de la profundidad y orientación.

3. Representación del modelo 3D. Se forma durante el procesamiento perceptual y es independiente del punto de vista del observador.

Marr y Nishihara [53] argumentaron que el reconocimiento de objetos involucra el comparar la representacón actual del modelo 3D con representaciones 3d almacenadas del modelo. Asi mismo, argumentaron que se identifican primeramente concavidades, seguidas de segmentos y finalmente ejes.

1.1.2. Teoría de reconocimiento por componentes de Biederman

En esta teoría [5] se sugiere que los objetos consisten de componentes llamados *geons*, por ejemplo bloques, cilindros, etc. Los *geons* se determinan por medio de la extracción de bordes y por medio de un análisis del número de partes que tiene un objeto (enfocándose en las áreas cóncavas).

De acuerdo con Biederman, existen cinco propiedades invariantes de los bordes, los cuales son usados para construir los *geons*. Estas son:

1. Curvatura
2. Paralelismo
3. Co-terminación
4. Simetría
5. Co-linearidad

Cuando se determinan los *geons* de un objeto, la información se compara con las representaciones almacenadas de los objetos. La identificación del objeto se determina por cuál de las representaciones almacenadas produce una mejor correspondencia.

Biederman sugiere tres razones por las cuales somos capaces de reconocer objetos aunque existan condiciones de observación pobres.

1. Las propiedades invariantes se pueden detectar incluso cuando se ven sólo partes de los objetos.
2. Siempre y cuando se puedan observar las concavidades de un contorno, las partes faltantes de éste pueden ser reconstruídas.
3. Mucha de la información disponible para objetos complejos es redundante.

Biederman (al igual que Marr) asume que el reconocimiento de objetos es independiente del punto de vista.

1.1.3. Independencia del punto de vista

Tarr y Bülthoff [101] propusieron una teoría del reconocimiento de objetos que combina ambas aproximaciones. Ellos argumentaron que las representaciones almacenadas de los objetos se basan en experiencias previas con estos objetos, y que por lo tanto el reconocimiento de objetos es más fácil cuando la vista actual del objeto es la misma que la almacenada. Sin embargo, también reconocieron la importancia del mecanismo de la independencia del punto de vista al realizar fácilmente discriminaciones categóricas.

1.2. ¿Porqué es tan difícil el reconocimiento de objetos?

Podría parecer incialmente que el problema pudiera resolverse si contaramos con sistemas que tuvieran memoria lo suficientemente grande y eficiente como para poder determinar cuando una imagen que estamos viendo corresponde a un objeto que ya hayamos visto en el pasado. Podría ser posible aproximar el problema del reconocimiento de objetos si almacenaramos un número suficiente de diferentes vistas asociadas con cada objeto y después comparar la imagen de la vista actual con todas las vistas almacenadas en la memoria [1]. Se han propuesto varios mecanismos, conocidos como memorias asociativas, que implementan esta aproximación directa al reconocimiento. Estos mecanismos, normalmente basados en redes neuronales, pueden almacenar una gran cantidad de patrones $(P_1, P_2, ..., P_n)$, y después, dado un patrón de entrada Q, son capaces de recuperar el patrón P_i que sea más similar a Q [113] [38] [30].

La pregunta es: ¿Las memorias asociativas de este tipo han resuelto el problema de reconocimiento de objetos? Las discusiones de memorias asociativas algunas veces han sugerido que si lo han resuelto. Cuando el sistema ha almacenado una vista representativa, o varias vistas, de cada objeto, una nueva vista puede recuperar automáticamente la representación almacenada que más se le parece. El principal problema con esta aproximación directa es que se basa en la noción simple y restringida de una medida de similitud que da la distancia entre la imagen de entrada y cada una de las imágenes almacenadas previamente en memoria. Sin embargo, la simple comparación entre dos imágenes no es suficiente por si misma para lidiar con las variaciones tan grandes que puede haber entre las diferentes imágenes de un objeto dado. Una medida de similitud típica usada en los modelos de memoria asociativa es la llamada distancia de Hamming [26]. Otra medida de similitud que se usa frecuentemente es la llamada norma L_2 entre imágenes de niveles de grises, la cual suma las diferencias cuadráticas entre las intensidades de las imágenes en los puntos correspondientes.

Para el problema general de reconocimiento visual de objetos esta aproximación directa es insuficiente por dos razones. Primero, el espacio de todas las posibles vistas de todos los objetos a reconocer es extremadamente grande. La segunda, y más importante, es que la imagen a ser reconocida por lo general no será lo suficientemente similar a cualquier imagen que se haya visto en el pasado. Existen varias razones para

que se de esta variabilidad, y si queremos que el esquema de reconocimiento funcione, éste deberá tratar con estas variaciones [109]:

- *Punto de vista:* La principal razón de la variabilidad entre una imágen nueva y las almacenadas previamente tiene que ver con el efecto del punto de vista. Los objetos tridimensionales se pueden ver desde una gran variedad de puntos de vista (direcciones y distancias), y éstas diferentes vistas conducen hacia diferentes imágenes. Algunas veces es difícil de apreciar el efecto del cambio de dirección de la vista, ya que dos vistas del mismo objeto que se encuentran separadas por digamos 20 o 30 grados, generalmente se perciben muy similares.

- *Efectos fotométricos:* La segunda fuente de la variabilidad surge de los efectos fotométricos. Estos incluyen las posiciones y distribuciones de las fuentes de luz en la escena, su amplitud, los efectos de iluminación mutua por otros objetos y la distribución de sombras y especularidades. Los efectos fotométricos de este tipo pueden cambiar drásticamente la distribución de la intensidad de la luz en la imagen, pero éstos normalmente no afectan nuestra habilidad de reconocer los objetos en la imagen.

- *Disposición de los objetos:* En escenas naturales, los objetos raramente se encuentran aislados: normalmente se encuentran sobre algun fondo, al lado o con oclusiones parciales con otros objetos. Cuando la imagen contiene un objeto familiar en una nueva disposición, la nueva imagen (tomada como un todo) fallará en encontrar imágenes pasadas del mismo objeto almacenadas en memoria.

- *Cambio de forma:* La cuarta y última fuente de variaciones es el efecto del cambio de forma. Muchos objetos, como es el caso del cuerpo humano, pueden mantener su identidad mientras cambian su forma 3D. Si queremos reconocer ese tipo de objetos, el esquema de reconocimiento debe ser capaz de tratar con los efectos introducidos por cambios de forma. Los objetos cambiantes, tales como unas tijeras, pueden algunas veces estar compuestos de subpartes rígidas. Otros objetos pueden sufrir de distorciones no rígidas, por ejemplo los rostros al realizar diferentes expresiones faciales.

Adicionalmente podemos agregar los efectos provocados por las ilusiones ópticas que llevan a una mala interpretación de los objetos que se estan observando. El estudio científico de éstas inició con J. Oppel en 1854 [73] y continuó siendo un tema de interés para para la psicología. Los libros [76] y [19] presentan un estudio del sistema de percepción humano asi como una recopilación de las principales teorías de las ilusiones ópticas.

1.3. Conceptos y definiciones

El problema de **detección** tiene varios significados, que van desde el identificar la posición de un objeto conocido dentro de una imagen, hasta el identificar compo-

nentes de una clase particular de objetos a varios niveles de detalle. Por ejemplo, el encontrar rostros en una imagen.

El **reconocimiento** se refiere a la clasificación de objetos presentes en una región particular de la imagen [2]. Por ejemplo, después de detectar un rostro, el reconocimiento consistiría en identificar a la persona.

El encontrar un objeto predeterminado en una escena es sólo un subproblema de la meta más general y ambiciosa del área de Visión por Computadora. En términos generales, uno desearía poder desarrollar un sistema artificial que pudiera recibir como entrada una imagen y que identificara todos los objetos (o gran parte de ellos) que se encuentran presentes en una escena compleja. Este problema no sólo implica el uso de algoritmos de detección y de reconocimiento, sino que además requiere métodos para que pueda aprender a identificar nuevos objetos e incorporarlos a sus esquemas actuales de detección y de reconocimiento. Sin embargo, quizas lo más difícil de todo es: ¿cómo iniciar a procesar la imagen de una escena compleja sin ninguna información apriori acerca de su contenido?, ¿a qué ponerle atención primero? y ¿en qué regiones en particular debería implementarse un algoritmo de reconocimiento? Este problema general no tiene aún una solución [2], apesar de que nuestro sistema visual parece que lo puede resolver sin mayor problema y de una manera muy eficiente.

Resumiendo, el problema del **reconocimiento de objetos** consiste en determinar si alguno o varios objetos conocidos y almacenados previamente en una base de datos, aparece en una imagen dada. De tal forma que el reconocimiento de objetos se convierte en un problema de correspondencia entre los modelos de los objetos que se encuentran en una base de datos y las representaciones de los modelos extraídos apartir de los datos de brillo de una imagen. Las características deseables de un sistema de reconocimiento de objetos son [55]:

- *Generalidad.* La habilidad de reconocer cualquier objeto sin tener restricciones a una tarea específica,
- *Robustez.* La habilidad de reconocer objetos bajo condiciones arbitrarias, y
- *Aprendizaje simple.* La capacidad de evitar procedimientos especiales o demandantes para la obtención de la base de datos de modelos.

Obviamente éstos requerimientos son generalmente imposibles de llevar a cabo, por ejemplo, es imposible reconocer imágenes tomadas en completa oscuridad. Entonces el reto se convierte en desarrollar un método con las mínimas restricciones. De ésta forma las dos principales tareas a realizar en el proceso de reconocimiento de objetos son: la definición de las características y la definición del procedimiento para realizar la correspondencia.

Por supuesto, *la representación* del modelo del objeto es extremadamente importante. Es claro, que es imposible crear una base de datos con todas las vistas posibles de un objeto bajo todas las posibles condiciones de iluminación. Por lo tanto, las vistas de los objetos estarán sujetas a ciertas transformaciones relacionadas

con las condiciones de iluminación y otros posibles factores, lo cual lo convierte en un problema muy complejo y que actualmente es un reto para los investigadores de las áreas de Reconocimiento de Patrones, Visión por Computadora y áreas relacionadas.

Un **invariante** de una configuración geométrica es una función de la configuración cuyo valor es el mismo bajo una transformación en particular [109]. El problema de encontrar invariantes para el reconocimiento de objetos consiste en encontrar cantidades o propiedades de los objetos que no cambian, a pesar de que éstos sean afectados por algún grupo de transformaciones admisibles en el espacio ocupado por estos objetos. Entre las categorías más importantes que se manejan en el estudio de invariantes están las basadas en los momentos centrales y las basadas en las transformadas integrales [119]. El primer artículo basado en los momentos centrales fue publicado por Hu [32], quien presentó una teoría basada en los momentos bidimensionales. Hu también presento en [32] su álgebra de invariantes. Otro trabajo interesante publicado en 1995 sobre invariantes 3D, fue realizado por Burel y Henoq en 1955 [11].

1.4. Componentes de un sistema de reconocimiento de objetos

La entrada de un sistema de reconocimiento de objetos es una imagen digital, junto con algún tipo de conocimiento acerca de uno o más modelos de objetos. La salida es una etiqueta que clasifica al objeto, en algunos casos, también se da la posición del objeto dentro de la imagen. Pero, ¿cómo podemos encontrar un objeto a partir de una matriz de valores de pixeles? La estructura típica de un sistema de reconocimiento de objetos [43] se presenta en la Fig. 1.1. A partir de la imagen digital de entrada, se extraen un conjunto de características distintivas; entre las características más comunmente usadas se encuentran: bordes (discontinuidades de brillo), esquinas (intersecciones de bordes) y regiones (regiones homogéneas en la imagen). La meta del módulo de extracción de características es tomar una gran cantidad de datos de la imagen y quedarse con sólo aquella información necesaria para identificar o distinguir al objeto. Notemos que se puede usar conocimiento apriori acerca de los tipos de objetos que se encuentran presentes en la imagen. Por ejemplo, si sabemos que estamos buscando aeropuertos en imágenes aéreas, entonces podríamos elegir un operador de extracción de características que busque líneas largas en la imagen (correspondientes a los bordes de las pistas).

El reconocimiento de objetos se puede ver simplemente como un problema de búsqueda en bases de datos [43], en la cual las llaves de búsqueda se usan para obtener registros específicos de una base de datos. En nuestro caso, la base de datos contiene modelos de objetos (no registros) y las llaves de búsqueda son colecciones de carac-terísticas extraídas. Regresando a la Fig. 1.1, las características extraídas deben ser agrupadas bajo algun criterio en colecciones, llamadas primitivas de indexado. Ejemplos de primitivas de indexado incluyen a las colecciones de bordes o líneas extraídas, colecciones de características de esquinas o incluso de regiones homogéneas

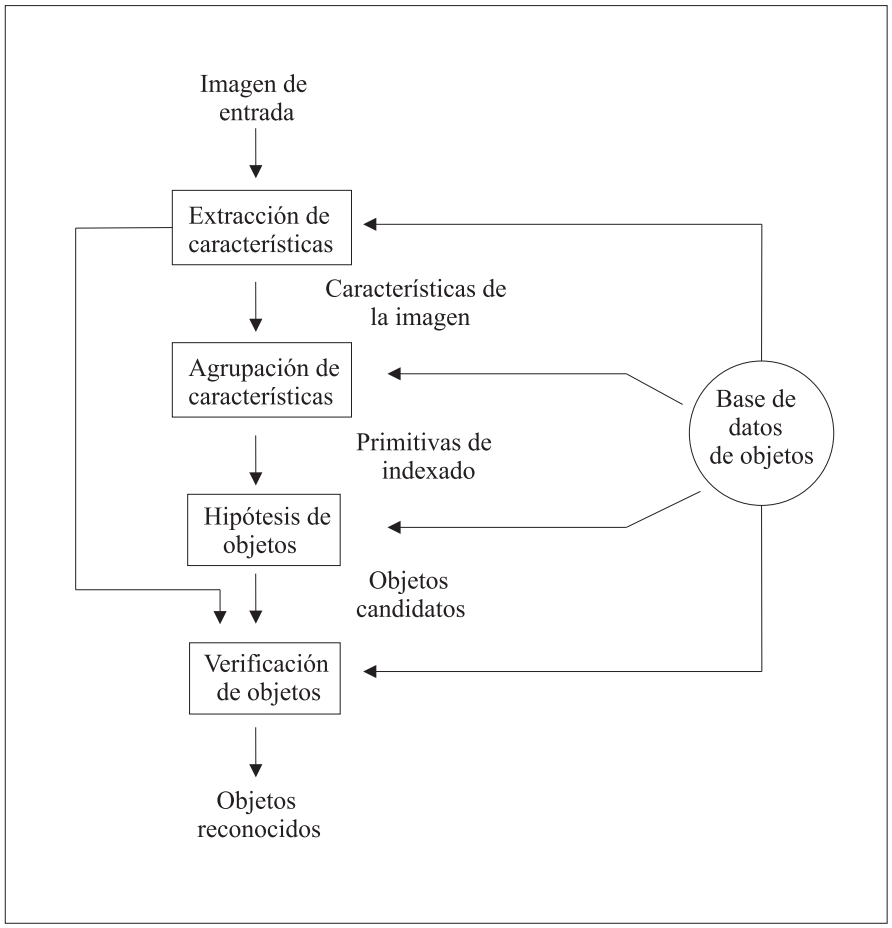


Figura 1.1: Los componentes de un sistema de reconocimiento de objetos

en la imagen. Una primitiva de indexado representa una petición a la base de datos del tipo: "Dame todos los objetos en la base de datos que tienen esta primitiva como componente". Al igual que en el caso de extracción de características, el contar con conocimiento del dominio de la imagen puede ayudar en el proceso de agrupamiento de características.

Una vez que se ha hecho la búsqueda, se usa un algoritmo de correspondencia que compara la primitiva de indexado con los modelos de los objetos que se encuentran en la base de datos, regresando un conjunto de objetos candidatos, en donde todos éstos contienen la primitiva de indexado. Al menos que la búsqueda regrese únicamente un modelo candidato, aún no hemos terminado, ya que debemos decidir cuál de los objetos candidatos es el que estamos buscando. De esta forma, el último componente del algoritmo de reconocimiento evalua (o verifica) a cada uno de los candidatos en términos de qué tanto aporta a los datos de la imagen. Generalmente se le asigna a cada candidato una puntuación, la cual sirve para elegir al mejor candidato que la mejor interpretación (o etiqueta) del objeto. Si existen otros objetos en la imagen, se puede repetir todo el proceso hasta que se hayan tomado en cuenta todas las características.

En la siguiente subsección se describen las principales aproximaciones al reconocimiento de objetos.

1.5. Aproximaciones

Históricamente se pueden identificar dos tendencias en los sistemas de reconocimiento de objetos, [55]. En un lado del espectro se encuentra la aproximación comúnmente conocida como *basada en geometría*. En esta aproximación el usuario provee al sistema con datos de la apariencia de un objeto por medio de un modelo de tipo CAD. Normalmente, este modelo describe sólo la forma 3D del objeto, y omite otras propiedades como el color y la textura. En el lado opuesto del espectro, se encuentra la aproximación en donde los métodos están basados en la apariencia, en éste caso no se requiere de un modelo geométrico que el usuario de explícitamente. Matas y Obdržálek presentaron en [55] una revisión del estado del arte de las diferentes aproximaciones al reconocimiento de objetos, analizaron sus fortalezas y debilidades y presentaron ejemplos de aplicaciones exitosas.

Las cuatro aproximaciones tradicionales al reconocimiento de objetos son (aunque la mayoría de los sistemas usa varias de estas técnicas):

- Métodos basados en la apariencia,
- correspondencia gramatical y de grafos,
- métodos basados en la geometría,
- basados en la correspondencia de características locales.

Un caso particular que ha tenido mucho auge en los últimos años es el reconocimiento como una correspondencia de características locales.

1.5.1. Reconocimiento de objetos basado en la apariencia

La idea central detrás de los métodos basados en la apariencia es la siguiente. Habiendo visto todas las posibles apariencias de un objeto, ¿se puede llevar a cabo el proceso de reconocimiento con el simple hecho de recordar eficientemente a todas éstas?, en otras palabras, ¿se puede implementar el reconocimiento como una memoria (pictórica) visual eficiente? La respuesta obviamente depende de qué entendamos por "todas las apariencias". Esta aproximación se ha demostrado que es éxitosa para escenas que contienen objetos sin oclusiones y que se encuentran sobre fondo negro [69]. Sin embargo, es actualmente imposible recordar todas las posibles apariencias que los objetos podrían tener en el caso de encontrarse sobre un fondo arbitrario, estar parcialmente ocultos o con cambios en la iluminación.

Los métodos basados en la apariencia [8, 103, 100, 41, 65, 69] normalmente están compuestos de dos fases:

1. En la primera etapa, se construye un modelo a partir de un conjunto de imágenes de referencia. El conjunto contiene imágenes de la apariencia del objeto bajo diferentes orientaciones, iluminaciones y múltiples instancias de una clase de objetos, por ejemplo, rostros.

2. La segunda etapa, conocida como "recordar", consiste en extraer partes de la imagen de entrada (subimágenes del mismo tamaño de las imágenes de entrenamiento), posiblemente por medio de técnicas de segmentación o por medio de una exhaustiva enumeración de ventanas en la imagen que juntas cubren a toda la imagen. Es entonces cuando el sistema de reconocimiento compara una parte extraída de la imagen de entrada con las imágenes de referencia.

Las principales limitantes de ésta aproximación son:

- Requieren que se separe todo el objeto de interés del fondo de la imagen y
- son sensibles a oclusiones.

Resumiendo, los métodos basados en la apariencia son atractivos ya que no requieren de la detección ni de la correspondencia de características en la imagen ni de primitivas geométricas. Sin embargo, sus limitaciones (por ejemplo, la necesitad de un muestreo denso de vistas de entrenamiento y su poca robustez a las oclusiones y a los fondos arbitrarios y ruidosos) hacen que éstas técnicas sean adecuadas principalmente para ciertas aplicaciones con variaciones limitadas o controladas en las condiciones de la formación de la imagen, por ejemplo, para inspección industrial.

Se han realizado varios intentos para tratar con el problema del reconocimiento de objetos con oclusiones parciales o con datos faltantes [67, 65, 95, 6, 41, 7, 96, 42, 35, 39]. A continuación se describen brevemente algunos métodos basados en la apariencia.

1.5.1.1. Correlación

Una de las primeras técnicas propuestas para detectar la presencia de un objeto conocido en una imagen, y en cuyo caso detectar su posición, está basada en la correlación entre la escena a ser analizada, $s(x, y)$, y el objeto a ser detectado, $r(x, y)$. La expresión matemática para la correlación esta definida por:

$$c(x, y) = s(x, y) \otimes r(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s(\xi, \eta) r^*(\xi - x, \eta - y) d\xi d\eta, \tag{1.1}$$

donde * denota el complejo conjugado y $\otimes$ denota la correlación. Esta función opera de tal manera que si existe una buena correspondencia entre un objeto en la imagen y el objeto a detectar, entonces aparece un pico alto en el plano de correlación $c(x, y)$ justo en la posición en donde se detecto el objeto en la imagen. La altura y lo puntiagudo del pico depende, entre otros factores, del grado de similitud entre las dos funciones correlacionadas.

La correlación descrita por la eq. 1.1 tiene serias deficiencias para reconocer objetos, entre las cuales destacan: problemas cuando existe un fondo ruidoso, no es tolerante a distorciones de la imagen, así como tampoco a cambios del objeto en rotaciones, escala e iluminación.

1.5.1.2. Correspondencia de histogramas

La familia de los métodos de reconocimiento de objetos basados en la apariencia incluyen a los métodos de correspondencia global de histogramas. En [98, 99], Swain y Ballard propusieron representar un objeto por medio de su histograma de color. De esta forma, se identifican a los objetos al hacer corresponder histogramas de regiones de la imagen con los histogramas de los modelos de las imágenes. Esta técnica es robusta a la orientación, escala y oclusión parcial de los objetos, sin embargo, es muy sensible a las condiciones de iluminación y no es muy adecuada para reconocer objetos que no puedan ser identificados únicamente por su color. Ésta aproximación fué después modificada por Healey y Slater [28] y por Funt y Finlayson [23] utilizando invariantes a la iluminación. Más recientemente, el concepto de correspondencia de histogramas fue generalizado por Schiele [86, 85, 84], quien en vez de usar los colores del pixel propuso utilizar las respuestas de varios filtros para construir los histogramas, llamados histogramas de campos receptivos (*receptive field histograms*).

1.5.2. Reconocimiento de objetos basado en la correspondencia gramatical y de grafos

En esta aproximación, el reconocimiento de objetos se lleva a cabo cuando un grupo de datos de la escena es estructuralmente idéntico a su correspondiente en el modelo, el cual normalmente expresa relaciones entre las características de la imagen, tales como bordes o regiones, pero también puede hacer referencia a relaciones entre los datos tridimensionales.

Para el caso de los objetos con características primitivas distinguibles que tengan relaciones fijas (geométricas o topológicas), se han desarrollado dos métodos generales. El primero es el método sintáctico (p.e [62] y [13]), en el cual las relaciones válidas se encapsulan en reglas gramaticales que después son usadas para realizar el reconocimiento. Este se lleva a cabo analizando sintácticamente los símbolos de los datos de acuerdo a estas reglas. Las principales aplicaciones de las técnicas gramaticales han sido en el reconocimiento de huellas [64], de diagramas de circuitos y en el análisis de texturas y cromosomas. Una variación de este método usa reglas para reconocer características específicas, por ejemplo, cierto tipo de vegetación en una imagen de satélite [68].

El segundo método general es la correspondencia de grafos, en donde la meta es encontrar un emparejamiento entre los grafos del objeto y del modelo. Una desventaja de esta aproximación es que los métodos de grafos tienden a ser NP-Completos y no son prácticos al menos que el tamaño de los grafos sea pequeño, lo cual hace que se requieran criterios heurísticos para comparar vértices y aristas, y para evaluar las correspondencias de subgrafos.

Dos técnicas clave son el isomorfismo de subgrafos y el problema de encontrar el máximo *clique* en grafos de asociación, que se describen en el siguiente capítulo.

Cuando el problema de reconocimiento de objetos se reduce al reconocimiento de patrones de líneas, el método más simple es el histograma geométrico de parejas [17]. Más recientemente se propuso una estrategia más sofisticada también basada en histogramas la cual usa un grafo de sus N-vecinos más cercanos [33]. En este trabajo se observó que el uso de estructuras relacionales ofrecen mejoras significativas en el desempeño cuando se trata de hacer corresponder patrones de líneas ruidosas.

Una aproximación más compleja consiste en la correspondencia de grafos relacionales con atributos. Cuando se logra esta abstracción surgen dos ingredientes computacionales clave que deben ser considerados. El primero consiste en encontrar una representación de los atributos que sea robusta al ruido y a la oclusión y además que sea invariante a cambios en la geometría del objeto provocacada por cambios en el punto de vista. El segundo ingrediente consiste en encontrar la manera de comparar los descriptores relacionales. Con respecto al primer punto, es decir, a una representación eficiente de los objetos, ha habido varias propuestas. Entre ellas se incluye el *hashing* geométrico y estructural [97]- [12], una variedad de invariantes [40] e histogramas geométricos de parejas [17]. Con respecto al segundo punto, de cómo comparar las representaciones es un problema viejo en la literatura de reconocimiento de patrones estructural. Por ejemplo, Shapiro y Haralick [91]-[92] han mostrado cómo se pueden comparar y emparejar descripciones relacionales inexactas contando posicionamientos consistentes de estructuras de aristas. Sanfeliu y Fu [78] proponen una medida más fina de medir la consistencia relacional, la cual usa una distancia de edición para comparar graficas relacionales. Wong y You [118] han desarrollado una aproximación basada en teoría de la información que usa la entropía para medir la similitud de grafos. Más recientemente, Wilson y Hancock [114] han reportado un esquema bayesiano que puede ser usado tanto para emparejamiento relacional

de grafos como para corregir errores estructurales por medio de la edición de grafos [115].

Los métodos convencionales de correspondencia estructural demandan demasiado tiempo de cómputo como para ser consideradas como máquinas de búsqueda. Esto lleva al surgimiento de muchas técnicas eficientes como el *hashing* estructural [97] y el etiquetado de relajación [14].

Finalmente, las aproximaciones basadas en la minimización de energía para resolver el problema de correspondencia de grafos se basan en la transformación del espacio de búsqueda discreto en uno continuo, en el cual se usan técnicas de optimización para encontrar una solución aproximada. Uno de los algoritmos representativos es el de *Softassign* [24], el cual optimiza una función de costo cuadrática. Lozano y Escolano [47] - [48] propusieron una mejora al algoritmo original al introducir los llamados *kernels de difusión*, los cuales son matrices de pesos y que de ahora en adelante se llamarán *kernels*. Los algoritmos detallados se presentan en el siguiente capítulo.

1.5.3. Reconocimiento de objetos basado en geometría

En los métodos basados en geometría, la información acerca de los objetos se representa explícitamente, de tal forma que el reconocimiento puede consistir en decidir cuando dada (una parte de) una imagen, ésta puede verse como una proyección del modelo (normalmente 3D) de un objeto conocido. Por lo general, se requieren dos representaciones: una para representar el modelo del objeto y otra para representar el contenido de la imagen. Para facilitar el procedimiento de encontrar una correspondencia entre el modelo y la imagen, las dos representaciones deben estar muy relacionadas. En el caso ideal existiría una relación simple entre las primitivas usadas para describir el modelo y las usadas para describir la imagen. Por ejemplo, si el objeto se describe por medio de un modelo de alambre, entonces una buena representación para la imagen sería en términos de bordes lineales. En éste caso, cada borde se puede hacer corresponder directamente con alguno de los modelos de alambre. Sin embargo, las representaciones del modelo y de la imagen por lo general tienen diferentes significados. El modelo podría estar describiendo la forma tridimensional de la forma de un objeto mientras que los bordes de la imagen podrían corresponder sólo a manifestaciones visibles de esa forma mezclada con bordes falsos por efectos de la iluminación (sombras) [55].

Para obtener invarianzas en la posición y en la iluminación, es preferible emplear primitivas como modelos que sean por lo menos un tanto invariantes con respecto a cambios en éstas condiciones. Se ha realizado un esfuerzo considerable en identificar primitivas que sean invariantes con respecto al cambio de punto de vista [66, 112].

Las principales desventajas de los métodos basados en geometría son:

- La dependencia en la extracción viable de primitivas geométricas (líneas, circulos, etc),

- la ambigüedad en la interpretación de las primitivas detectadas (presencia de primitivas que no están modeladas),
- las capacidades restringidas en el modelado de únicamente una clase de objetos que se caracterizan por estar compuestos de pocos elementos y que son fácilmente detectables, y
- la necesidad de construir los modelos manualmente.

Una forma de establecer la correspondencia entre el objeto en el modelo y el de la imagen es a través de una aproximación de alineación.

1.5.3.1. Aproximación de alineación

La idea básica de la aproximación de alineación consiste en encontrar la transformación que separa al objeto en la escena del objeto del modelo. Por ejemplo, el objeto en la escena y el del modelo pueden ser similares, excepto por una diferencia en tamaño. Escalar uno de ellos eliminaría esta discrepancia.

De manera general, la aproximación de alineación asume que para cada modelo M_i que represente un objeto y que este almacenado en la base de datos, existe un conjunto de transformaciones permitidas T_{ij} que el objeto podría sufrir, tales como cambios en escala, posición u orientación en el espacio. Así, el reconocimiento se puede ver como la búsqueda de un modelo particular y de una transformación en particular que maximize alguna medida de ajuste F entre el objeto de la escena y el del modelo. Si V es el objeto de la escena a reconocer entonces el reconocimiento consiste en la búsqueda del máximo de $F(V, (M_i, T_{ij}))$ sobre todos los modelos posibles M_i y sus transformaciones T_{ij}. Esto implica que se debe contar con un conjunto de características extraídas tanto del modelo como del objeto en la escena, las cuales pueden ser puntos, líneas o características invariantes, por ejemplo.

El uso de la alineación para el reconocimiento de objetos normalmente se restringe a cambios en posición, orientación y escala en el espacio bidimensional de la imagen. Su uso para el caso más general nos lleva a tener que lidiar con ciertas dificultades que deben ser consideradas. Por ejemplo, el conjunto de transformaciones que tienen que considerarse para el caso de objetos tridimensionales no se limita a las simples transformaciones mencionadas anteriormente. Cuando un objeto se mueve y rota en el espacio tridimensional, las transformaciones inducidas en la imagen son considerablemente más complejas. Otra dificultad es que los métodos usados para la normalización típicamente se basan en propiedades globales tales como el área aparente del objeto, la longitud del perímetro o el centro de masa, los cuales no se comportan bien cuando hay oclusiones parciales.

1.5.4. Reconocimiento de objetos basado en características locales

Los métodos basados en geometría y en la apariencia, no funcionan del todo bien con respecto a las características que se desearía que tuviera un sistema de

reconocimiento de objetos: generalidad, robustez y fácil aprendizaje. Las aproximaciones basadas en geometría requieren que el usuario especifique los modelos de los objetos y por lo general sólo puede trabajar con objetos que consisten de primitivas geométricas simples. Además, por lo general no soportan el fácil aprendizaje. Por el otro lado, los métodos basados en la apariencia requieren de un conjunto exhaustivo de imágenes de aprendizaje, tomadas desde puntos de vista y cambios en iluminación densamente distribuídos. Tal conjunto de imágenes, sólo puede ser obtenido cuando el objeto se puede observar en un ambiente controlado, por ejemplo, cuando se coloca sobre una mesa que gira. Estos métodos son además sensibles a las oclusiones parciales de los objetos y a los fondos desconocidos, lo cual hace que no sean robustos.

Como un intento para resolver las problemáticas arriba mencionadas, se han propuesto métodos basados en la correspondencia de características locales. En este caso, los objetos se representan por medio de un conjunto de características locales, las cuales se calculan automáticamente a partir de imágenes de entrenamiento. Después se organizan las características aprendidas en una base de datos. Finalmente, cuando se desea reconocer uno o varios objetos contenidos en una imagen, se extraen las características como en la etapa de entrenamiento, y se determina la presencia de un objeto conocido en términos del número de correspondencias entre las características locales. Ya que este procedimiento no requiere que correspondan todas las características locales, estas aproximaciones son robustas a oclusiones y a fondos desconocidos.

Para poder reconocer objetos desde diferentes puntos de vista, es necesario que se manejen todas las variaciones de la apariencia de los objetos. En general, las variaciones pueden ser complejas, pero en la escala de las características locales éstas pueden ser modeladas fácilmente, por ejemplo por medio de transformaciones afines. Por lo tanto, al permitir transformaciones simples en la escala local, se logra tener una invarianza al punto de vista significativa, incluso para objetos con formas complicadas. Como resultado de ésto, es posible obtener modelos de objetos a partir de sólo algunas pocas vistas, tomadas por ejemplo cada 90 grados.

Las principales ventajas de las aproximaciones basadas en la correspondencia de características locales se resumen a continuación:

- Aprendizaje. La construcción de los modelos internos de los objetos conocidos se realiza automáticamente a partir de las imágenes de éstos, lo cual implica que no se requiera de la intervención del usuario, excepto para proveer al sistema de las imágenes de entrenamiento.

- La representación local está basada en la apariencia. No hay necesidad de extraer primitivas geométricas (por ejemplo, líneas), las cuales son por lo general difíciles de detectar.

- No se requiere de una segmentación previa al reconocimiento, que separe a los objetos del fondo, y aún así los objetos que se encuentran sobre un fondo desconocido pueden ser reconocidos.

- Los objetos de interés se pueden reconocer incluso si se encuentran parcialmente ocultos por otros objetos en la escena.
- Las variaciones complejas en la apariencia de los objetos causadas por la variación del punto de vista y por las condiciones de iluminación, se aproximan usando transformaciones simples a una escala local.
- Las mediciones tanto en la base de datos como en las imágenes de los objetos a reconocer se obtienen y se representan de la misma manera.

Se han propuesto varias aproximaciones basadas en características locales, las cuales por lo general siguen cierta estructura en común que se resume a continuación.

1.5.4.1. Estructura general

1. **Detectores.** Primeramente, se detectan los elementos de "interés" en la imagen, los cuales servirán como posiciones de soporte, en donde se calcularán descriptores locales de apariencia. Un elemento de una imagen es de interés si éste representa una parte de un objeto que puede ser detectada repetidamente y localizada en imágenes tomadas sobre una amplia gama de condiciones. El reto se convierte entonces en encontrar tales puntos de interés. La alternativa de implementar detectores de fuerza bruta consiste en generar descriptores locales en cada punto de la imagen. Esto, por supuesto, no es viable debido a su complejidad computacional.
2. **Descriptores.** Una vez que se encuentran los elementos de interés, se codifica la apariencia en su vecindad local de la imagen, de tal forma que ésta permita la búsqueda de elementos similares. Cuando se diseña un descriptor (también llamado vector de características), se deben tomar en cuenta varios factores. Primero, los descriptores deben ser lo suficientemente discriminativos como para distinguir entre las características de los objetos almacenados en la base de datos. Otro aspecto del diseño de un descriptor es que éste debe ser invariante, o al menos robusto en cierta medida, a variaciones en la apariancia de los objetos que no estan reflejadas por el detector. Si por ejemplo, el detector detecta regiones circulares o elípticas sin asignarles una orientación, entonces el descriptor debe hacerse invariante a la orientación (invariantes a la rotación). O si el detector es impreciso en localizar los elementos de interés, por ejemplo, al tener poca tolerancia en pixeles, entonces el descriptor debe ser insensible a estas pequeñas desalineaciones. Tal descriptor podría estar basado por ejemplo en momentos de color (estadísticas integrales sobre toda la región), o sobre histogramas locales.
3. **Indexado.** Durante el aprendizaje de los modelos de los objetos, se almacenan en la base de datos los descriptores de las apariencias locales. En la fase de reconocimiento, se calculan los descriptores para la imágen que contiene los objetos a ser reconocidos y se buscan descriptores similares en la base de

datos (correspondecias muy probables). Es por ésto que la base de datos debe ser organizada (indexada) de tal manera que permita que se obtengan los descriptores similares de manera eficiente. El significado de "indexado adecuado" depende generalmente de las propiedades de los descriptores (por ejemplo, su dimensionalidad) y de la medida de distancia usada para determinar cúales son los más similares (por ejemplo, la distancia euclidiana). Generalmente, para obtener un rendimiento óptimo del índice (tiempos rápidos de obtención), se debe buscar una buena combinación del descriptor y la medida de distancia, que minimize la proporción de las distancias a las correspondencias correctas y a las incorrectas. La elección del esquema de indexado tiene un gran impacto en la velocidad del proceso de reconocimiento, especialmente cuando se tienen bases de datos muy grandes.

4. **Correspondencia.** Cuando se desea reconocer algún objeto, primero se calculan las características locales de la misma manera en que se hizo para las imágenes de los objetos almacenados en la base de datos. Entonces, se establecen cero o varias correspondencias tentativas para cada característica detectada en la imagen con los objetos a reconocer. Después se realiza una búsqueda en la base de datos utilizando alguna medida de similitud (por ejemplo la euclidiana o la de mahalanobis) entre las características de los objetos desconocidos con los almacenados en la base de datos, quedándose únicamente con aquellas correspondencias que se encontraron lo suficientemente cerca (bajo un cierto criterio). Éstas correspondencias tentativas están basadas únicamente en la similitud de los descriptores. A un objeto de la base de datos que tuvo un alto número de correspondencias con los descriptores de la imagen de entrada, se le considera como una correspondencia candidata.

5. **Verificación.** La similitud de los descriptores, por si misma, no es una medida lo suficientemente confiable como para garantizar que se ha establecido una correspondencia correcta. Es por ésto que como paso final del proceso de reconocimiento, se agrega la etapa de verificación de la presencia del modelo en la imagen de entrada. Una forma muy común de hacerlo es estimando de manera robusta una transformación global que conecta las imágenes, por ejemplo, usando el algoritmo RANSAC (Random Sample Consensus) [20]. Como se mencionó anteriormente, si un detector no puede recobrar ciertos parámetros de las transformaciones de la imagen, los descriptores se deben diseñar de tal forma que sean invariantes a éstas. Sin embargo, es preferible contar con un detector covariante que con un descriptor invariante, ya que ésto permite una consistencia global más poderosa en la verificación. Si por ejemplo, el detector no provee de las orientaciones de los elementos de la imagen, entonces se deben emplear descriptores invariantes a la rotación. En éste caso, es imposible verificar que todas las correspondencias de los elementos coinciden en sus orientaciones. Finalmente, las correspondencias tentativas que no son consistentes con la transformación global estimada se descartan, y sólo las correspondencias

restantes se usan para estimar la puntuación final de la correspondencia.

Las principales aproximaciones al reconocimiento de objetos basadas en la correspondencia de características locales se presentan en las siguientes subsecciones.

1.5.4.2. La aproximación de David Lowe

David Lowe desarrolló un sistema de reconocimiento de objetos [4, 50, 51, 9, 10, 49] en el cual puso énfasis en la eficiencia, con lo cual logró tener un sistema de reconocimiento en tiempo real. Su método consiste en detectar puntos de interés que sean invariantes a las transformaciones de escala, rotación y translación. Debido a que las zonas locales sufren de transformaciones más complicadas, se propuso un descriptor basado en histogramas locales, el cual es robusto a las imprecisiones en la alineación de las zonas.

1. **Detector.** La detección de las regiones de interés se describe a continuación:

   a) Detección de picos en el espacio escala [116] y [117]. Se detectan a todas las escalas y en todas las posiciones en la imagen, las regiones circulares con respuesta máxima al filtro de diferencia de gaussianas (DoG). Una implementación eficiente utiliza la pirámide del espacio escala. El proceso inicia aplicando una convolución con un filtro gaussiano a la imagen inicial. Se sigue aplicando el filtro a las imágenes resultantes, obteniendo finalmente un conjunto de imagenes espacio escala. Las imágenes adyacentes de éste conjunto se restan para obtener un conjunto de imágenes DoG. Enseguida se detectan los mínimos y máximos locales de éstas últimas imágenes, tanto en el dominio espacial como en el dominio de la escala. El resultado de esta primera etapa consiste de un conjunto de tripletas x, y y σ, de posiciones en la imagen y escalas características.

   b) Se refinan las posiciones de los puntos detectados. Ésto se lleva a cabo ajustando localmente las respuestas DoG usando una función cuadrátrica 3D, y se determinan con una exactitud de subpixeles las posiciones y las escalas características de las regiones circulares. El refinamiento es necesario, ya que a niveles altos de la pirámide, el desplazamiento de un sólo pixel podría producir un desplazamiento muy grande en el dominio de la imagen. Enseguida, las regiones inestables se descartan, bajo el criterio de que la estabilidad esta dada por la magnitud de la respuesta DoG. De esta forma, se eliminan las regiones con una respuesta menor a un umbral predefinido. También se descartan las regiones que se encuentran sobre un borde recto, las cuales, apesar de tener una respuesta DoG alta, son inestables en su localización en una dirección.

   c) Se asignan una o mas orientaciones a cada region. Para ésto, se forman histogramas locales de orientaciones de gradiente, en donde los picos en el histograma determinan las orientaciones características.

2. **Descriptor (SIFT).** Se miden los gradientes locales de la imagen en la escala característica de la región, tomando la distancia apartir del centro de la región y combinándolos en un conjunto de histogramas de orientación. Usando los histogramas, los pequeños errores de alineación en la localización no afectan la descripción final. La construcción de los descriptores permite tener rotaciones 3D de aproximadamente 20 grados, antes de que el modelo de similitud falle. Finalmente, cada región detectada se representa por medio de un vector de dimensión 128.

3. **Indexado.** Para que el sistema de reconocimiento soporte una obtención rápida de los vectores en la base de datos, se adoptó una modificación al algoritmo del árbol *kD*, llamado BBF (*best bin first*). Este algoritmo da una respuesta aproximada, en el sentido de que éste regresa el vecino más cercano con una alta probabilidad, o en otro caso, regresa otro punto que se encuentre muy cerca en distancia al vecino más cercano. El algoritmo BBF modifica el algoritmo del árbol *kD* para buscar *bins* en el espacio de características ordenando bajo el criterio de la distancia más cercana de la posición de la búsqueda, en vez de ordenar los elementos utilizando la jerarquía del árbol.

4. **Verificación.** La transformación de Hough se usa para identificar los grupos de correspondencias tentativas con una transformación geométrica consistente. Ya que la transformación se aproxima por una similitud, el acumulador de Hough es de dimensión 4 y se particiona para obtener más entradas de rangos inequívocos. De ésta forma sólo se consideran aquellos agrupamientos con al menos 3 votos en una entrada de rango. Cada uno de éstos agrupamientos se somete a un proceso de verificación geométrica, en el cual se usa un procedimiento iterativo de ajuste de mínimos cuadrados para encontrar la mejor proyección afín que relacione la imagen a reconocer con las imágenes de la base de datos.

La Figura 1.2 muestra un ejemplo de los resultados obtenidos usando SIFT.

1.5.4.3. La aproximación de Mikolajczyk y Schmid

La proximación de Mikolajczyk y Schmid se describe en [77, 59, 87, 88, 89, 90, 60, 16]. Utilizaron un método de detección basado en una generalización afin del detector de esquinas de Harris, en donde los puntos de interés se detectan y se describen por medio de derivadas Gausianas de las imágenes de intensidad en vecindades elípticas.

1. **Detector.** En su trabajo implementaron una adaptación afin del detector de puntos de Harris. Propusieron una solución la cual busca iterativamente una adaptación en forma afin en vecindades de puntos detectados en el espacio escala uniforme. Para la inicialización, se extraen posiciones y escalas aproximadas de puntos de interés utilizando el detector estándar multiescala de Harris. Estos puntos no son invariantes afines, debido a que se usa el kernel

Figura 1.2: Ejemplo del reconocimiento de los objetos con oclusiones, usando SIFT. Imagen tomada de [50].

Gaussiano uniforme. Dada la solución inicial aproximada, su algoritmo modifica iterativamente la forma, la escala y la posición espacial de la vecindad para cada punto y converge a puntos de interés afines. Para mas detalles ver [59].

2. **Descriptores y Correspondencia.** Los descriptores se componen de derivadas gaussianas calculadas sobre las regiones normalizadas en la forma. La invarianza a la rotación se obtiene al "guiar" las derivadas en la dirección del gradiente. Usando derivadas de a lo más cuarto grado, se obtienen descriptores de dimensión 12. La similitud entre los descriptores se mide en su primera aproximación usando la distancia de Mahalanobis. Después se confirman o se rechazan las correspondencias obtenidas que son más prometedoras. Ésto lo hacen usando una medida de correlación cruzada que se calcula sobre ventanas normalizadas en las vecindades.

3. **Verificación.** Una vez que se obtienen las correspondencias punto a punto, se calcula una estimación robusta de la transformación geométrica entre las dos imágenes. Esta estimación usa el algoritmo RANSAC (Random Sample Consensus) [20]. Más recientemente, Dorko y Schmid [16] extendieron la aproximación hacia la categorización de objetos. En esta nueva aproximación se detectan y se describen parches locales en la imagen, de la misma manera como se describio anteriormente. Se colectan varios parches de diferentes objetos de una cierta categoría (por ejemplo, carros) y se entrena a un clasificador para distinguir los parches de las diferentes categorías.

   La Fig. 1.3 muestra algunos de los resultados obtenidos con éste método.

1.5.4.4. La aproximación de Tuytelaars, Ferrari y van Gool

Luc van Gool y sus colaboradores desarrollaron una aproximación basada en la correspondencia de características locales [106, 108, 18, 105, 104, 107, 102].

1. **Detector.** Propusieron dos métodos para la extracción de regiones invariantes afines, basados en geometría y en intensidades. Las regiones son covariantes afines, éstas adaptan su forma a la plantilla de intensidades, con la finalidad de seguir representando la misma parte física de un objeto. Además de la invarianza geométrica, se utiliza la invarianza fotométrica, la cual permite tener escalas independientes para cada uno de los tres canales de color. La extracción de la región siempre inicia detectando los puntos de interés que son estables. Los puntos de interés son también puntos de Harris [27], o extremos locales de la imagen de intensidad. Apesar de que la detección de los puntos de Harris no es realmente invariante afin, debido a que es circular el conjunto de soporte sobre el cual se calcula la respuesta, los puntos son aún bastante estables a los cambios de punto de vista y pueden ser localizados con precisión. Los extremos en intensidad, por otro lado, son invariantes a cualquier transformación geométrica continua y a cualquier transformación monotónica de la intensidad, pero no son

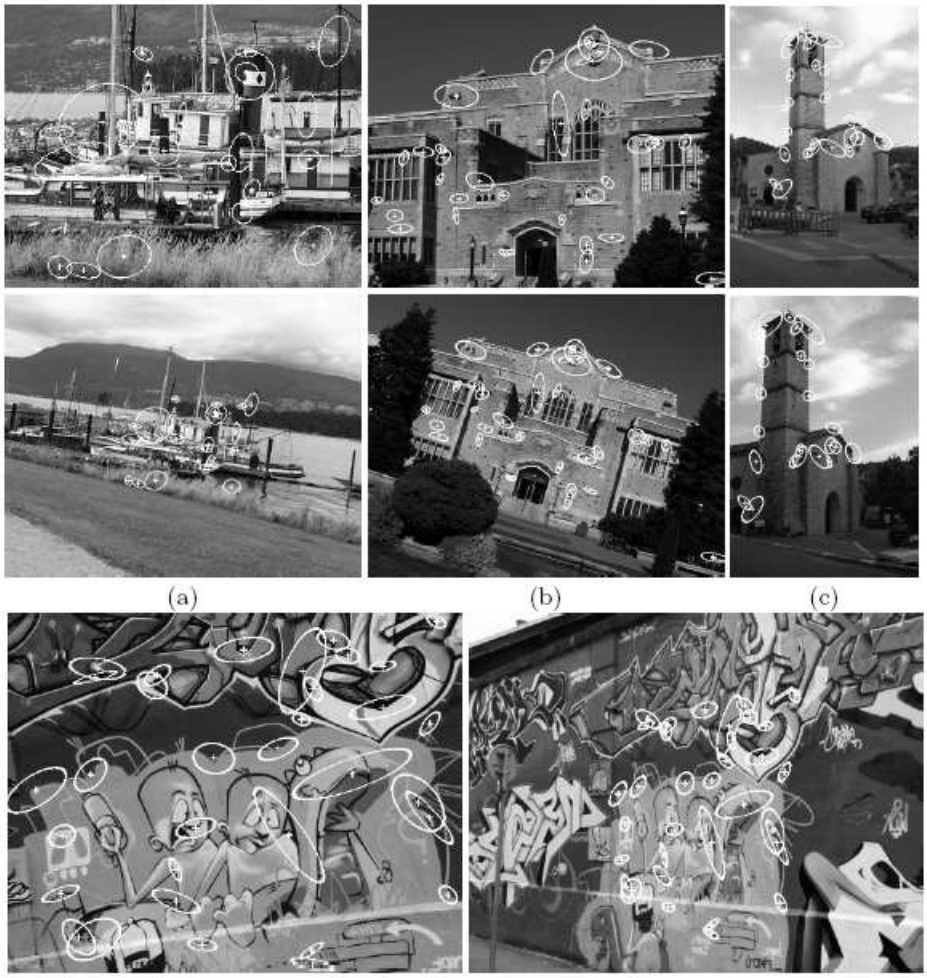


Figura 1.3: Algunos resultados presentados por Mikolajczyk y Schmid. Parejas de imágenes que muestran la imágen de consulta y la imágen más similar que se encontró en la base de datos. Se encontraron 22 correspondencias para la primer pareja, 32 para la segunda, 22 para la tercera y 33 para la cuarta. Todas las correspondencias son correctas. Imagen tomada de [59].

localizables con mucha precisión. En las imágenes a color, la detección se lleva a cabo tres veces, una para cada banda de color.

2. **Descriptores y Correspondencia.** En el caso de regiones basadas en geometría, cada una de las regiones se describe por medio de un vector de 18 momentos de color generalizados [63], invariantes a transformaciones fotométricas. Para las regiones basadas en intensidad, se usan 9 momentos de color generalizados invariantes a la rotación. La similitud entre los descriptores esta dada por la distancia de Mahalanobis, las correspondencias entre las dos imágenes se forman a partir de las regiones con distancia más pequeña. Una vez que se encuentran las regiones correspondientes, se calcula como una verificación final la correlación cruzada entre éstas. En el caso de las regiones basadas en intensidad, en donde la rotación es desconocida, se maximiza la correlación cruzada sobre todas las rotaciones.

3. **Verificación.** El conjunto de correspondencias tentativas se reduce usando restricciones geométricas y fotométricas. La restricción geométrica básicamente rechaza las correspondencias que contradicen la geometría epipolar. La restricción fotométrica asume que siempre existe un grupo de regiones correspondientes que experimentan la misma transformación de intensidades. Se rechazan las correspondencias que tienen una transformación fotométrica singular.

La Fig. 1.4 muestra algunos de los resultados obtenidos.

1.5.4.5. La aproximación LAF de Matas et al.

La aproximación de Matas et al. [56, 70, 57, 71] inicia con la detección de regiones extremas de estabilidad máxima. Después se establecen sistemas de coordenadas locales covariantes afines (llamados Local Affine Frames, LAF's) y se toman medidas relativas a éstas para describir las regiones.

1. **Detector.** Las regiones extremas de estabilidad máxima (MSERs) se introdujeron en [56]. Las propiedades atractivas de las MSERs son: 1) invarianza a las transformaciones afines de coordenadas de imágenes, 2) invarianza a la transformación monotónica de intensidad, 3) su complejidad computacional es casi lineal en el número de pixeles y consecuentemente muy cercano a tiempos de ejecución en tiempo real, y 4) ya que no incluye un procesamiento de suavizado, se pueden detectar estructuras en la imagen que sean muy finas o gruesas. Iniciando con los contornos de la región detectada, se construyen marcos locales (sistemas de coordenadas) de varias maneras que sean covariantes afines. Los marcos locales afines facilitan la normalización de las regiones de la imagen en un marco canónico que permite la comparación directa de valores de intensidad de fotometría normalizada, eliminando con ésto la necesidad de invariantes.

Figura 1.4: Algunos resultados presentados por Tuytelaars, Ferrari y Van Gool. A la izquierda se muestran tres vistas de la misma escena. A la derecha, se muestra el seguimiento de la foto de la mujer. Imagen tomada de [108].

2. **Descriptor.** Se usaron tres diferentes descriptores. El primero consiste en usar directamente las intensidades de las regiones locales [70, 57, 71]. Las intensidades se discretizan en barridos de 15 x 15 x 3, obteniendo así descriptores de dimensión 675. El tamaño es lo suficientemente discriminativo como para distinguir entre una gran cantidad de objetos en la base de datos. El segundo tipo de descriptor usa la transformación discreta coseno, la cual se aplica a las regiones discretizadas [72]. El número de coeficientes de frecuencia baja DCT que se guardan en la base de datos, se usa para adaptar la preferencia de la discriminación en los descriptores contra la tolerancia de localización. Finalmente, el tercer descriptor usa invariantes a la rotación [56].

3. **Verificación.** Las correspondencias se verifican al seleccionar de manera robusta solo aquellas que cumplan con la restricción de la geometría epipolar. Para el reconocimiento de objetos es muy típico que sea suficiente con aproximar la transformación geométrica global por medio de una homografía con tolerancia flexible que se incrementa hacia los bordes del objeto.

1.5.4.6. La aproximación de Zisserman et al.

A. Zisserman y sus colaboradores desarrollaron estrategias para la correspondencia de características locales principalmente en el contexto del problema de correspondencia estereo [75, 74, 81, 79, 80]. Recientemente, presentaron un trabajo muy interesante que relaciona el problema de recuperación de imágenes con el problema de recuperación de texto [93, 82, 83]. Introdujeron un sistema de recuperación de imágenes, llamado VideoGoogle, el cual es capaz de procesar e indexar películas enteras.

1. **Detectores y Descriptores.** Emplearon dos tipos de detectores de elementos locales en la imagen. Uno, es el de regiones elípticas de forma adaptable de Mikolajczyk and Schmid, como se describe en la subsección 1.5.4.3. El segundo que usaron es el de Regiones Extremas de Estabilidad máxima explicada en la subsección 1.5.4.5. La representación de la apariencia local se realiza usando los descriptores SIFT introducidos por David Lowe. Con el conocimiento apriori de que se va a procesar una secuencia de video, se pueden entonces eliminar las regiones inestables y las ruidosas. Las regiones que se detectaron en todos los cuadros del video se rastrean usando un modelo dinámico simple de velocidad constante y correlación. Cualquier región que no sobreviva por más de tres cuadros se elimina. El estimado del descriptor para una región se calcula tomando el promedio de los descriptores através de la secuencia.

2. **Indexado y Correspondencia.** Los descriptores se agrupan en diferentes clases de acuerdo a su similitud. Al igual que en la recuperación de texto, se ignoran las palabras comunes como por ejemplo “el”, en el caso de imágenes se eliminan las clases muy grandes. Cuando se observa una nueva imagen, cada uno de los descriptores de la nueva imagen se hace corresponder únicamente

contra representantes de cada clase. La selección de la clase más cercana genera inmediatamente correspondencias para todos los cuadros de la clase, para toda la película. De esta manera se evita la comparación exhaustiva de cada descriptor de cada cuadro. La medida de simulitud usada tanto para la generación de las clases como para la determinación de la clase más cercana, esta dada por la distancia Mahalanobis de los descriptores.

3. **Verificación.** Los cuadros de video se obtienen primeramente usando la frecuencia de descriptores con correspondencia, después se vuelven a evaluar basandose en una medida de consistencia espacial de las correspondencias. Las regiones con correspondencia proveen una transformación afin entre la imagen observada y la recuperada, de tal forma que se puede usar una correspondencia local punto a punto. Se define después un área de búsqueda para cada correspondencia por medio de algunos vecinos cercanos. Otras regiones que también corresponden dentro de esta área le dan un voto a ese cuadro. Finalmente se eliminan las correspondencias sin soporte y el valor final del cuadro esta determinado por la suma total de los votos.

1.6. Síntesis

En este capítulo se presentaron cuatro diferentes aproximaciones al reconocimiento de objetos. ¿Cuál es el correcto o cual es el mejor? La experiencia dice que cada uno de ellos cuenta con sus características particulares, lo cual hace que bajo ciertas condiciones en específico alguno sea mejor que otro y con ésto el que la combinación de dos o más resulte en mejores sistemas de reconocimiento de objetos.

En esta tesis se toma la combinación de características locales invariantes, las cuales han mostrado dar muy buenos resultados en los últimos años en parte debido a que permiten realizar el reconocimiento bajo diferentes puntos de vista y con oclusiones parciales [61], junto con un método propuesto de correspondencia estructural para la fase de verificación.

Con respecto al primer componente, el extractor de características locales invariantes, en este capítulo se presentaron seis propuestas recientes. En el siguiente capítulo se presenta una visión general de las diferentes alternativas que existen al problema de encontrar las correspondencias correctas entre las características detectadas en el modelo y las de la escena, haciendo énfasis en aquellas basadas en grafos.

Capítulo 2

Algoritmos de correspondencia de características

El problema de correspondencia de características es uno de los problemas centrales en el campo del análisis de imágenes, ya que éste constituye un paso fundamental en muchas de las aplicaciones de visión por computadora tales como la recuperación de estructuras tridimensionales, la estimación de movimiento y por supuesto, el reconocimiento de objetos. El problema consiste en encontrar los puntos (en ambas imágenes) que corresponden al mismo elemento de la escena.

En literatura reciente, ha surgido el siguiente esquema como el método de elección preferido para realizar una correspondencia efectiva [110]:

1. Detección de puntos característicos, por ejemplo [49], [59] y [106].
2. Aplicación de una correlación entre los puntos característicos seguida de un proceso de umbral para obtener un primer conjunto de correspondencias candidatas.
3. Uso de un método robusto, tal como RANSAC [20] o LMedS (*Least Median of Squares*), para estimar la geometría epipolar o trifocal del sistema de la cámara.
4. Rechazo de las parejas candidatas que son incompatibles con la geometría estimada de la cámara.
5. Realización de un proceso de correspondencia guiado que hace uso de la geometría estimada para encontrar más correspondencias.

La eficiencia y precisión de este esquema depende en gran medida de la calidad del conjunto de las correspondencias candidatas que se obtienen en el paso 2, y a pesar de que normalmente éstos métodos son robustos a cierta cantidad de ruido, el estimador que se usa en el paso 3 requiere de un conjunto de entrada con un número suficiente de correspondencias correctas para encontrar una solución correcta y de

una baja proporción de correspondencias erróneas para ejecutarse eficientemente. Por estas razones, normalmente se agrega otro paso entre los pasos 2 y 3 que se encarga de filtrar el conjunto inicial de correspondencias. Esto se lleva a cabo generalmente introduciendo algunas restricciones básicas que permitan eliminar correspondencias que se sospecha son incorrectas. Estas restricciones adicionales son básicas en el sentido de que en la etapa en la que se aplican, la geometría epipolar o trifocal de la cámara es aún desconocida y por lo tanto el proceso de correspondencia guiada no es posible en este punto.

Vincent and Laganière [110] presentaron una revisión de las diferentes estrategias de correspondencia y propuesieron una evaluación empírica de su desempeño. Ellos evaluan el desempeño en términos de su habilidad para reducir el número de correspondencias erróneas de un conjunto dado de correspondencias, y preservar las buenas. Lowe [49] utilizó una transformada de Hough para identificar cúmulos de correspondencias tentativas con una transformación geométrica aproximada. Solo los cúmulos con al menos tres entradas en un intervalo se consideran para la siguiente etapa. En seguida, cada cúmulo se somete a un proceso de verificación geométrico en el cual se aplica un algoritmo de ajuste de mínimos cuadrados. Mikokajczyk y Schmid [59] realizaron una estimación robusta de la transformación entre las dos imágenes usando un algoritmo de *RANSAC*. Tuytelaars et al [104] aplicaron dos restricciones para la eliminación de correspondencias erróneas, una geométrica y otra fotométrica. La restricción geométrica básicamente consiste en rechazar aquellas correspondencias que contradicen la geometría epipolar. La restricción fotométrica asume que siempre existe un grupo de regiones correspondientes que sufren de la misma transformación de intensidades, de tal forma que se rechazan aquellas correspondencias que tienen una transformación fotométrica singular.

Alternativamente a esta aproximación de alineación, existen otros trabajos que proponen realizar la correspondencia de las características a través de un enfoque estructural aplicando algoritmos de grafos. En las siguientes secciones se presenta una visión general de éstos, haciendo énfasis en el algoritmo de *Softassign* debido a que éste es usado como algoritmo de comparación contra el algoritmo propuesto en esta tesis.

2.1. Correspondencia de características basada en grafos

Los grafos permiten codificar información significativa estructural y relacional de patrones proviendo de una invarianza útil para el reconocimiento de objetos. Dado un grafo de entrada que represente al objeto, su reconocimiento basado en grafos normalmente se pone en términos de encontrar el grafo del modelo más similar (estructuralmente compatible) en la base de datos, incluso si existe un grado de distorción significativo entre ambos grafos. Los grafos pueden representar partes del objeto o características invariantes de éstos. Además cuentan con propiedades de invarianza importantes. Por ejemplo, si un grafo (que se dibuja en papel) se translada, se rota o se transforma en su imagen espejo, éste es aún el mismo grafo en

el sentido matemático. Estas propiedades de invarianza junto con el hecho de que los grafos son muy adecuados para modelar objetos complejos, los hacen muy atractivos para varias aplicaciones.

Actualmente existe una amplia gama de propuestas para resolver este problema, entre ellas destacan las aproximaciones basadas en una distacia de edición del grafo, las basadas en encontrar el máximo *clique* y las de correspondencia de grafos. Estas tres aproximaciones se describen brevemente a continuación.

2.1.1. Correspondencia basada en una distancia de edición

La correspondencia de grafos basada en una distancia de edición es una generalización de la correspondencia de cadenas, o de la distancia de edición de cadenas. Para medir la similitud de dos grafos se introducen operaciones de edición, por ejemplo, la eliminación, inserción y sustitución de nodos y aristas en donde comunmente se les asigna un cierto costo a cada operación. Los costos dependen de la aplicación y reflejan el parecido de las distorsiones de los grafos. Muchos de estos algoritmos están basados en la búsqueda en árboles que usan una heurística de evaluación tipo A^* para podar el espacio de búsqueda [91] [58] [78], o utilizan un relajado probabilístico [14], redes neuronales [120], recocido simulado [29] y algoritmos genéticos [111].

En esta aproximación de correspondencia de grafos, la función de costo, es decir, el costo asignado a las operaciones de edición, tiene una gran influencia en los resultados, en otras palabras, dos grafos que bajo una función de costo son muy parecidos bajo otra función pueden ser muy diferentes. De manera similar, la correspondencia óptima de los nodos y aristas de dos grafos puede cambiar significativamente si la función de costo cambia.

2.1.2. Correspondencia basada en encontrar el máximo clique

Una forma de representación estructural para encapsular tanto las características a hacer corresponder como las relaciones entre éstas, es a través de los grafos relacionales. Asi, se puede construir un grafo para cada imagen a ser correspondida, representando las características como nodos y sus relaciones (los valores invariantes) codificados en las aristas. De esta manera, la correspondencia se puede formular como un problema de isomorfismo de subgrafos entre dos grafos relacionales. Formalmente, dados dos grafos $G' = (V', E')$ y $G'' = (V'', E'')$ con el mismo orden y tamaño, se define un isomorfismo entre éstos dando una biyección $\phi : V' \rightarrow V''$ tal que $(i, j) \in E' \iff (\phi(i), \phi(j)) \in E''$, para toda i, $j \in V'$. Se dice entonces que dos grafos son isomorfos si existe un isomorfismo entre ellos. La meta se convierte entonces en determinar, uno o más cúmulos de nodos que sean mutuamente compatibles de acuerdo a la similitud de las relaciones invariantes codificadas en las aristas, y así cada cúmulo representará a un objeto a ser reconocido.

Dado un grafo arbitrario no dirigido $G = (V, E)$ y un subconjunto de vértices C es llamado *clique* si todos sus vértices son mutuamente adyacentes, es decir, si para toda $i, j \in E$. Se dice que que el *clique* es máximo si es el *clique* más grande del grafo

y maximal si no existe otro conjunto que lo contenga. Ahora, la meta es equivalente a determinar todos los *cliques* maximales presentes en el grafo.

Por lo general los algoritmos para encontrar los *cliques* maximales se han basado en técnicas de búsqueda recursivas, las cuales exploran todos los mapeos potenciales entre dos grafos relacionales [31]. Estas técnicas recursivas dan una solución correcta, sin embargo requieren de mucho tiempo para explorar todas las posibles soluciones. Existen también métodos que aproximan la solución a través de la minimización de una función de costo dada ([14],[44],[114]), los cuales a pesar de que proveen solo una aproximación, convergen en tiempos polinomiales a una solución óptima lo cual los hace viables para su uso en el reconocimiento de objetos.

2.1.3. Correspondencia de grafos

En esta aproximación, una vez que se han representado la imagen del modelo y la de la escena por medio de grafos ($G_D = (V_D, E_D)$ y $G_S = (V_S, E_S)$, respectivamente), debemos ser capaces de compararlos para averiguar si pertenecen a un mismo objeto o no. Para poder hacer esto, antes se debe establecer un emparejamiento entre los nodos de ambos grafos. La representación de este emparejamiento de nodos es la matriz de correspondencias M (de tamaño m x n) en la que el valor de cada elemento esta dado por:

$$M_{ai} = \begin{cases} 1 & \text{si } a \in V_D \text{ se empareja con } i \in V_S \\ 0 & \text{de otra forma} \end{cases}$$

La matriz de correspondencias M deberá cumplir con la restricción de tener a lo sumo 1 en cada fila y columna, ya que un nodo de un grafo puede estar emparejado con a lo más un nodo del otro grafo.

$$\forall a \sum_{i=1}^{m} M_{ai} \leq 1, \forall a_i \; M_{ai} \in 0, 1 \tag{2.1}$$

En la Fig. 2.1 se muestra un posible emparejamiento entre dos grafos.

Dada esta matriz de correspondencias, podemos evaluar el parecido de ambos grafos contando el número de aristas comunes entre nodos emparejados.

$$F(G_D, G_S; M) = \sum_{a=1}^{|V_D|} \sum_{i=1}^{|V_S|} \sum_{b=1}^{|V_D|} \sum_{j=1}^{|V_S|} M_{ai} M_{bj} Caibj \tag{2.2}$$

siendo,

$$C_{aibj} = D_{ab} S_{ij} \tag{2.3}$$

donde D es la matriz de adyacencia del grafo G_D y S es la matriz de adyacencia de G_S.

En otros términos, lo que se esta valorando en esta función de costo es que si un nodo $a \in V_D$ se empareja con otro nodo $i \in V_S$, entonces los nodos $b \in V_D$ adyacentes

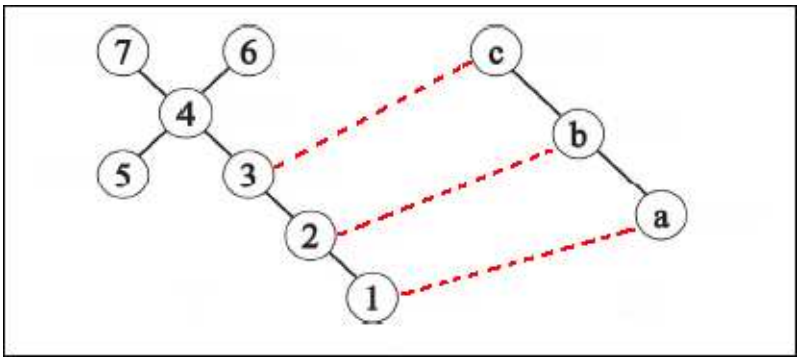


Figura 2.1: Posible emparejamiento entre dos grafos.

a a en G_D también deberían emparejarse con los nodos $j \in V_S$ adyacentes a i en G_S. Esta es mejor conocida como la regla del rectángulo (en términos de maximización se desea obtener el mayor número de rectángulos posible).

2.1.4. Algoritmo Softassign

Obtener la correspondencia óptima M entre dos grafos es un problema NP-completo. El algoritmo $Softassign$ [24] esta basado en el descenso por gradiente, para obtener este emparejamiento entre ambos grafos. Lo primero que hace es transformar las desigualdades 2.1 en las restricciones de la matriz M en las siguientes desigualdades:

$$\sum_{i=1}^{m+1} M_{ai} = 1, \forall a \text{ y } \sum_{a=1}^{n+1} M_{ai} = 1, \forall i \tag{2.4}$$

Para hacer esto añade una fila y una columna a la matriz M, las cuales representan dos variables ficticias. La nueva matriz se denota por M'. De esta forma, cuando un nodo de un grafo no se empareje con ningún nodo de otro grafo, se evitará poner un 0 en su correspondiente fila o columna. En su lugar, lo que se hará será poner un 1 en la variable ficticia de dicha fila o columna. Esto permite manejar ruido, tolerando de esta forma las características espurias que puedan aparecer en los grafos, lo cual mejorará la robustez del método de correspondencia.

Se actualiza cada elemento M_{ai} de la matriz M en la dirección del gradiente. Para ello se deriva parcialmente la función de costo respecto a cada elemento M_{ai}:

$$Q_{ai} = \frac{\delta F(G_D, G_S; M)}{\delta M_{ai}} = \sum_{b=1}^{m} \sum_{j=1}^{n} M_{bj} C_{aibj} \tag{2.5}$$

El valor de cada derivada parcial Q_{ai} será proporcional al beneficio que obtendríamos si emparejaramos el nodo a de G_D con el nodo i de G_S, es decir, si estableciéramos $M_{ai} = 1$. Por lo tanto, el algoritmo se basará en estos valores para

construir la matriz M. Cada elemento de la matriz M_{ai} será actualizado utilizando la exponencial del valor de Q_{ai}, atenuada por una variable β:

$$M_{ai}^0 = e^{\beta Q_{ai}} \tag{2.6}$$

La variable β se va incrementando a medida que itera el algoritmo, de tal forma que al comienzo habrá menor diferencia entre los distintos elementos M_{ai}, centrándose el algoritmo en la búsqueda, mientras que en las últimas iteraciones se centrarán en seleccionar los mejores emparejamientos, dándole un valor alto a los elementos que dan un mejor costo y prácticamente 0 al resto de los elementos.

Sin embargo, esta asignación de valores a los elementos de la matriz M' no garantiza que se vayan a cumplir las restricciones de la eq. 2.4. Para hacer que la matriz cumpla estas restricciones se utiliza el proceso de *Sinkhorn* [94], que consiste en una normalización cruzada de filas y columnas. Se normalizan de forma alternativa las filas y las columnas de la matriz M' hasta que ésta converja. Este proceso de *Sinkhorn* se realiza en el bloque C del algoritmo, tal como se muestra en el listado 2.1.

Listing 2.1: Algoritmo de *Softassign* para obtener el emparejamiento entre dos grafos

```
/* Inicializa */
β = β_0
M'_ai = (1 + ε)
/* Inicio iteraciones */
Comienzo A: Repetir A hasta que  β ≥ β_f
        Comienzo B: Repetir B hasta que M converja o
                    el num. de iteraciones > I_0
                Q_ai = Σ_{b=1}^{m} Σ_{j=1}^{n} M_bj C_aibj
                M^0_ai = exp{β Q_ai}
                Comienzo C: Repetir hasta que M' converja o
                            el num. de iteraciones > I_1
                        M'^1_ai = M'^0_ai / Σ_{j=1}^{m+1} M'^0_aj
                        M'^0_ai = M'^1_ai / Σ_{b=1}^{n+1} M'^1_bi
                Fin C
        Fin B
        β = β_r β
Fin A
Heuristica de limpieza
Devuelve M
```

Una vez finalizado el algoritmo, no se asegura que los elementos de la matriz M valgan 0 o 1, por lo que se debe utilizar una heurística de limpieza para hacer que esta restricción se cumpla. Se puede poner a 1 el mayor elemento de cada fila y columna, y dejar el resto a 0.

El algoritmo tiene una serie de parámetros que se deben ajustar. Para controlar los valores que tomará la variable β, se tienen los parámetros β_0, β_r y β_f. β_0 será el

valor inicial que tome β, β_f será su valor final y β_r será el factor que se utilizará para incrementar β en cada iteración. Por otro lado, para controlar el número de iteraciones máximo, en caso de que no converja antes, se tienen los parámetros I_0 e I_1. I_0 es el número de iteraciones máximo para el ciclo B, mientras que I_1 será el número de iteraciones máximo para el proceso de *Sinkhorn* (ciclo C). Los autores propusieron un posible valor para estos parámetros de $\beta_0 = 0{,}5$, $\beta_f = 10$, $\beta_r = 1{,}075$, $I_0 = 4$ e $I_1 = 30$.

Finalmente se dirá que la matriz converge cuando la diferencia entre las dos matrices generadas en iteraciones consecutivas caiga por debajo de un cierto umbral:

$$\sum_{a=1}^{A}\sum_{i=1}^{I} |M_{ai}^0 - M_{ai}| < \epsilon \tag{2.7}$$

El valor del umbral ϵ dependerá del ciclo para el que queramos comprobar la convergencia. En el caso del ciclo B se puede utilizar un valor $\epsilon = 0{,}5$, mientras que para el proceso de *Sinkhorn* (ciclo C) se puede utilizar $\epsilon = 0{,}05$ (propuestos por los autores).

2.1.4.1. Softassign con kernels

Lozano M. y Escolano F. presentaron en [47] y [48] una versión con *kernels* del algoritmo original de *Softassign*. Esta propuesta mejora significativamente el desempeño del algoritmo de correspondencia de grafos original, transformando el problema de correspondencia de grafos no pesados a la correspondencia de grafos pesados, los cuales se basan en las entropías de las distribuciones de probabilidad asociadas a los vértices después de calcular los *kernels*. Mostraron en sus experimentos que el agregar pesos a la función cuadrática original resulta en una mejora notable del desempeño de la correspondencia, incluso con condiciones de ruido. Su propuesta consiste en redefinir la eq. 2.3 como

$$C_{aibj}^K = D_{ab}S_{ij}exp - [(H_a^{K_D} - H_i^{K_S})^2 + (H_b^{K_D} - H_j^{K_S})^2], \tag{2.8}$$

en donde las entropías H^{K_D} y H^{K_S} están asociadas con los *kernels*

$$K_D = e^{-\frac{\beta}{m}L_D} \text{ y } K_S = e^{-\frac{\beta}{n}L_S},$$

L_D y L_S corresponden a los laplacianos de los grafos. El laplaciano de un grafo se define como

$$L_{ij} = T_{ij} - A_{ij} = \begin{cases} -1 & \text{si } (i,j) \in E \\ T_{ij} & \text{si } i = j \\ 0 & \text{de otra forma} \end{cases}$$

en donde A es la matriz de adyacencia del grafo y T es un a matriz diagonal definida por:

$$T_{ij} = \begin{cases} \sum_{j=1}^{n} A_{ij} & \text{si } i = j \\ 0 & \text{de otra forma} \end{cases}$$

Finalmente, las entropías H_{K_x} y H_{K_y} se definen como

$$H_i^K = -\sum_{j=1}^{m} K_{ij} log K_{ij}. \tag{2.9}$$

La Fig. 2.2 ilustra los kernels y la entropía de dos grafos. La Fig. 2.3 muestra las correspondencias encontradas por el algoritmo de *Softaasign* con *kernels* entre los grafos asociados a imágenes de fachadas. En el renglón superior se muestran las correspondencias entre los grafos asociados a cuadros consecutivos de cambios de punto de vista. El renglón de en medio muestra también las correspondencias pero ahora con grafos más grandes. Finalmente, en el renglón inferior se muestran del lado izquierdo los resultados de las correspondencias entre cuadros lejanos de una misma fachada y del lado derecho las correspondencias entre dos fachadas diferentes.

2.2. Síntesis

En este capítulo se presentó una visión general de los algoritmos de correspondencia de características para el reconocimiento de objetos. Se presentó un esquema general de emparejamiento que se ha convertido en el preferido en los últimos años, el cual está basado en obtener los puntos característicos, realizar un emparejamiento inicial tentativo, estimar la geometría epipolar o trifocal del sistema de la cámara a través del uso de un método robusto como RANSAC o LmedS, rechazar las parejas candidatas incompatibles con la geometría estimada y finalmente aplicar un proceso de correspondencia guiado que hace uso de la geometría estimada para encontrar más correspondencias.

Alternativamente a este esquema, existen otros trabajos, entre los cuales están aquellos que proponen realizar la correspondencia de las características a través de un enfoque estructural aplicando algoritmos de grafos. En este capítulo se presentó un resumen de las aproximaciones más populares haciendo énfasis en el algoritmo de *Softassign* debido a que éste será tomado como punto de comparación del algoritmo propuesto en esta tesis.

En el siguiente capítulo se presenta el algoritmo de emparejamiento basado en la transformación de grafos (el cual es la principal aportación de esta tesis) así como el esquema general del reconocimiento de objetos en el cual es usado.

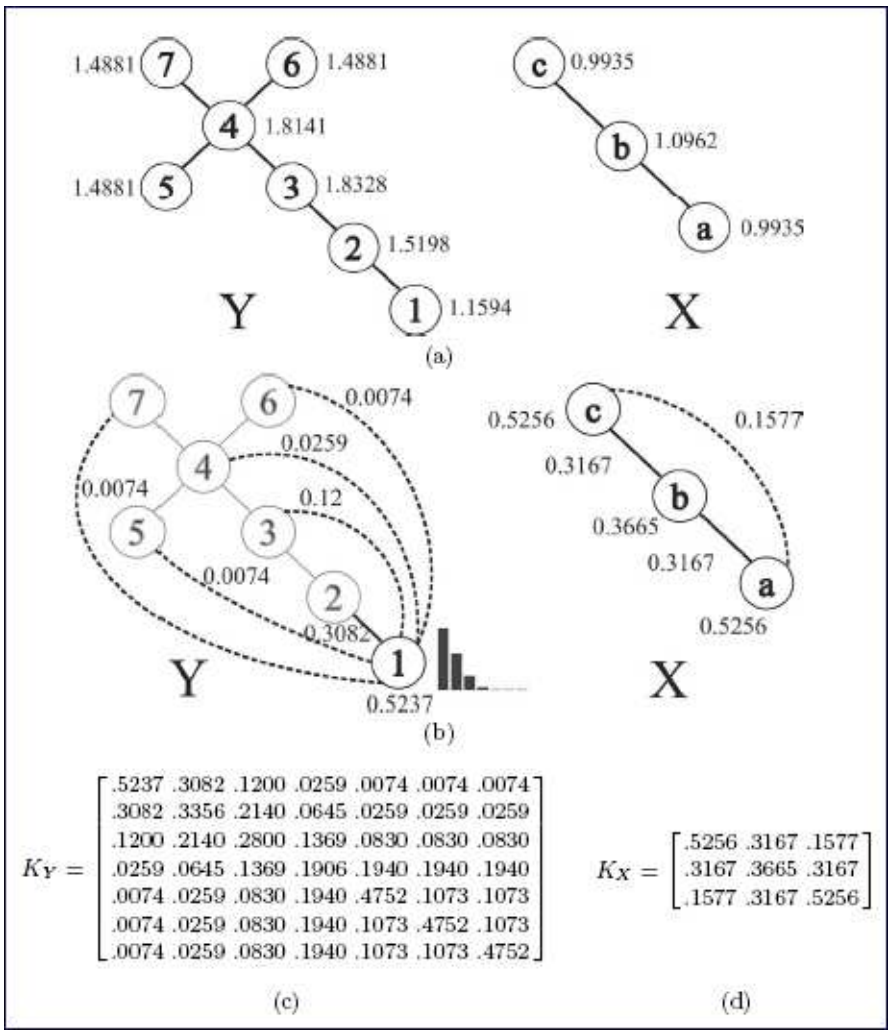


Figura 2.2: Ilustra los *kernels* y la entropía. (a) Ejemplo de dos grafos X y Y en donde los nodos están etiquetados con sus entropías, (b) los valores del *kernel* y su distribución para el vértice 1 del grafo Y así como los valores del *kernel* para todos los vértices del grafo X, (c) *kernel* K_Y y (d) *kernel* K_X. Figura tomada de [47].

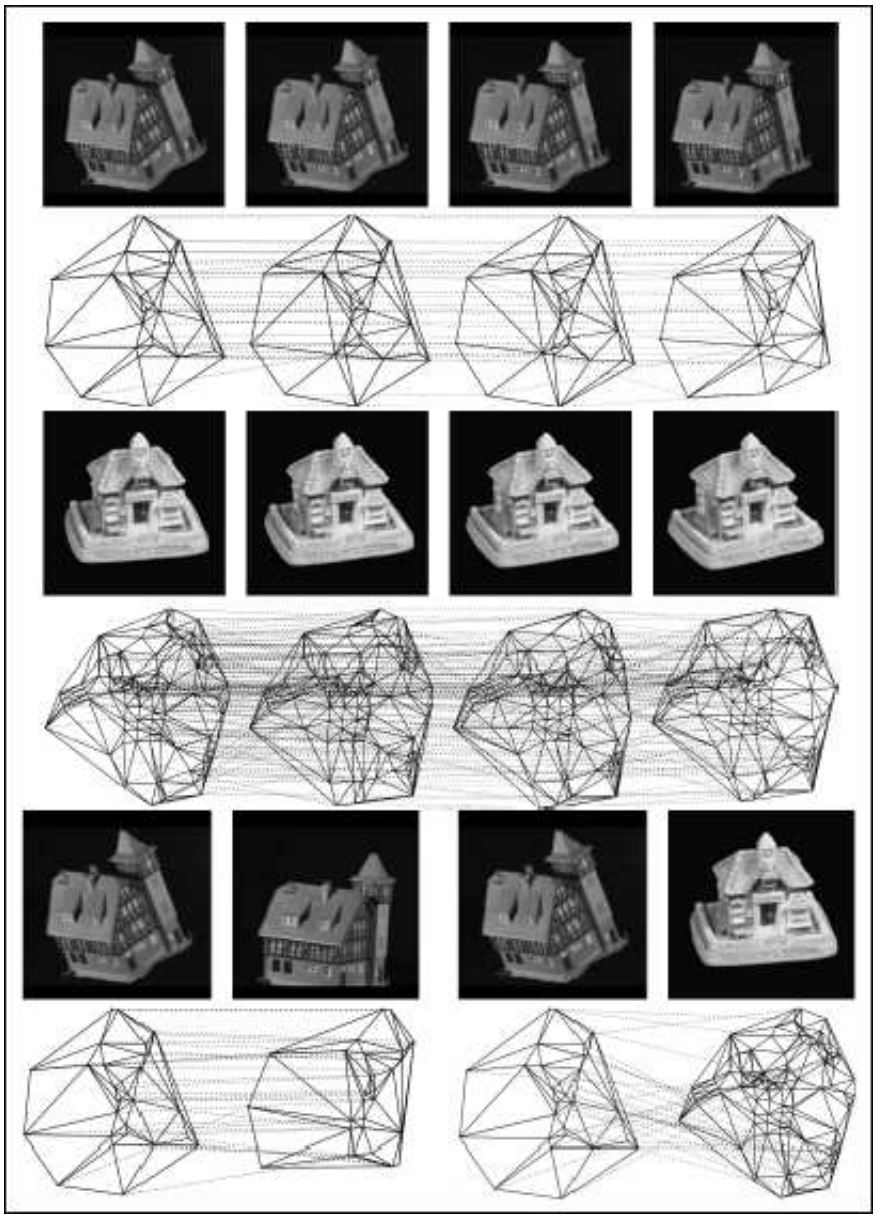


Figura 2.3: Resultados de la correspondencias encontradas por el algoritmo de *Soft-assign* con *kernels* para imágenes de fachadas. Figura tomada de [47].

Capítulo 3

Algoritmo de *Transformación de Grafos* para la correspondencia de características locales

En esta tesis se propone realizar el reconocimiento de objetos a través del emparejamiento estructural de características locales, el cual consiste en extraer características tanto de la imagen de la escena actual como de la imagen del objeto a reconocer (previamente almacenado en la base de datos) y realizar un emparejamiento estructural entre las características de ambas imágenes. En la Fig. 3.1 se muestra el diagrama de bloques del sistema propuesto. En la fase de extracción de características se puede utilizar cualquier extractor, que puede ir desde uno muy simple basado en gradiente hasta uno mucho más sofisticado como *SIFT* (descrito en la sección 1.5.4.2). Algunas de estas posibilidades se explican en el capítulo 1. Una vez que se obtienen las características de ambas imágenes, se les aplica un algoritmo que realiza una comparación de los descriptores y obtiene un emparejamiento inicial (si es que lo hay). Este algoritmo puede ser una simple correlación o el algoritmo de *BBF (Best Bin First)*, descrito en la sección 1.5.4.2, por mencionar algunos. Si este algoritmo nos da un emparejamiento inicial posible, entonces el objeto en cuestion se convierte en un objeto candidato que debe ser verificado. Es importante esta fase, ya que la similitud de los descriptores, por sí misma, no es una medida lo suficientemente confiable como para garantizar que se ha establecido una correspondencia correcta. Es por ésto que como paso final del proceso de reconocimiento de objetos, se agrega la etapa de verificación de la presencia del modelo en la imagen de entrada. La forma más común de realizar ésto es estimando de manera robusta una transformación global que conecte las imágenes, sin embargo, en esta tesis se propone realizar la verificación encontrando aquellas características que sean estructuralmente compatibles en ambas imágenes, para lo cual se diseñó un algoritmo al que se le ha llamado *Transformación de Grafos* y que es la aportación principal de esta tesis.

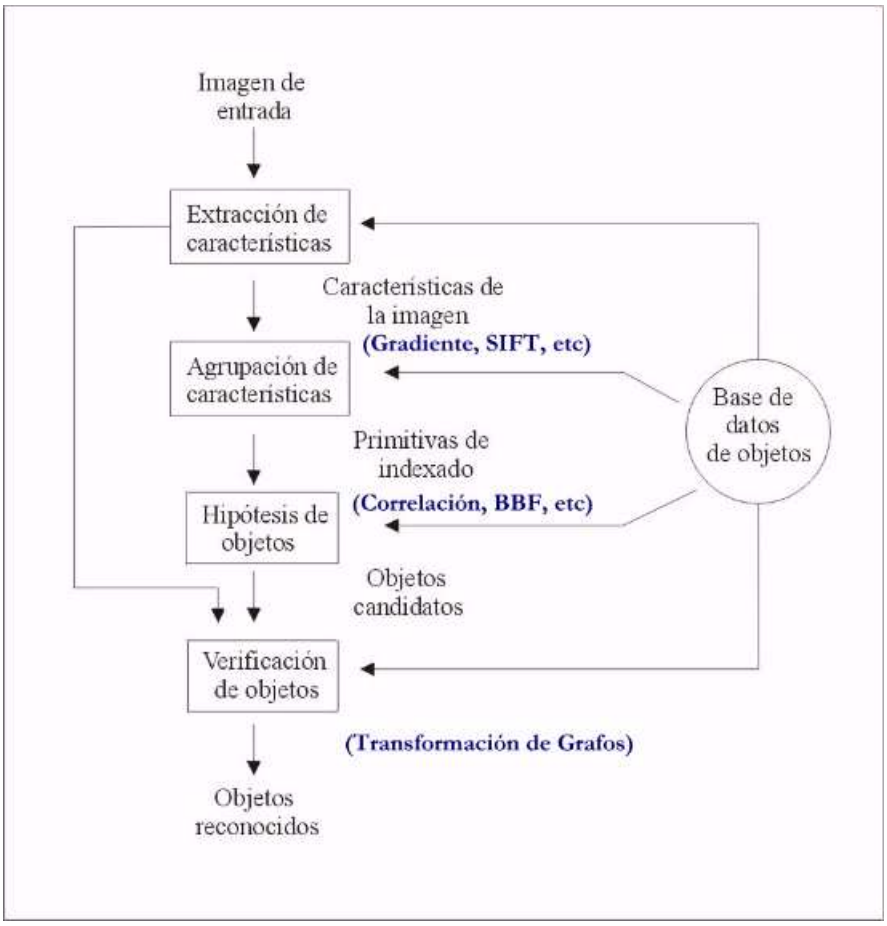


Figura 3.1: Diagrama de bloques del sistema de reconocimiento de objetos

En las siguientes secciones se describe el método de *Transformación de Grafos*, se presenta el algoritmo de fuerza bruta, se analiza su complejidad y se presenta una versión optimizada.

3.1. Transformación de Grafos

Consideremos un emparejamiento inicial entre las características del modelo $F = f_1, f_2, ..., f_n$ y las de la escena $F' = f'_1, f'_2, ..., f'_n$, en donde f_i se empareja con f'_i. Así mismo, sean (x_i, y_i) y (x'_i, y'_i) las coordenadas de pixel correspondientes a f_i y a f'_i. La etapa de verificación consiste en tomar este emparejamiento inicial y eliminar las correspondencias erróneas utilizando una aproximación estructural iterativa.

Dadas las coordenadas de pixel del emparejamiento inicial se construye un grafo para el modelo y otro grafo para la escena de la siguiente manera.

Sea $G_F = (V_F, E_F)$ un grafo no dirigido y no pesado, donde V_F es el conjunto de

vértices (correspondiendo a las características F) de tamaño n y E_F es el conjunto de aristas que se definen como sigue.

Sean u_i y u_j dos elementos de V_F, con coordenadas $P_i = (x_i, y_i)$ y $P_j = (x_j, y_j)$ respectivamente. Se forma una arista si se cumplen las siguientes dos condiciones:

1. P_j es uno de los k vecinos más cercanos de P_i.
2. $dist(P_i, P_j) \leq mediana_F$

donde, *dist* es la distancia euclidiana y la $mediana_F$ se calcula con las distancias entre todas las coordenadas de las características de F. La primera restricción indica que un vértice puede validar la estructura sólo de sus vértices cercanos. La segunda restricción indica ¿qué tan cercanos?, a lo mas la distancia de la mediana entre todos los vértices del grafo. Se eligió la mediana porque ésta representa el centro de la distribución de todas las distancias y no es tan sensible a los puntos extremos como la media.

Si un cierto punto P_i no tiene k vecinos que cumplan con las dos condiciones anteriores, entonces se elimina y se borran todas las aristas que había a él.

Sea A_F la correspondiente matriz de adyacencia de G_F

$$A_{F_{ij}} = \begin{cases} 1 & \text{si } (u_i, u_j) \in E_F \\ 0 & \text{de otra forma} \end{cases}$$

De la misma manera se definen $G_{F'}$ y $A_{F'}$, pero ahora para las características de la escena, de tal forma que (u_i, u_i'), $u_i \in V_F$ y $u_i' \in V_{F'}$, se emparejan.

La Fig. 3.2 muestra un ejemplo de un emparejamiento inicial entre las características de la imagen del modelo (izquierda) y las de la escena (derecha), en donde u_i se empareja con u_i'. Sus grafos correspondientes creados conectando cada uno de sus vértices con sus k vecinos más cercanos, para $k = 2$, se presentan en la Fig. 3.3 y sus matrices A_F y $A_{F'}$ son:

$$A_F = \begin{array}{c} \\ u1 \\ u2 \\ u3 \\ u4 \\ u5 \\ u6 \end{array} \begin{array}{c} \begin{array}{cccccc} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}, A_{F'} = \begin{array}{c} \\ u_1' \\ u_2' \\ u_3' \\ u_4' \\ u_5' \\ u_6' \end{array} \begin{array}{c} \begin{array}{cccccc} u_1' & u_2' & u_3' & u_4' & u_5' & u_6' \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Nótese que el grado de los vértices no es el mismo que el valor elegido para k, como podría suponerse. Esto es debido a que la grafica es no dirigida.

Figura 3.2: Ejemplo de un emparejamiento inicial entre las características de la imagen del modelo (izquierda) y las de la escena (derecha).

Figura 3.3: Ejemplo de los grafos de un emparejamiento inicial entre las características de la imagen del modelo (izquierda) y las de la escena (derecha), con $k = 2$.

Definición 3.1 Una correspondencia (f_i, f_i') con vértices (u_i, u_i') es errónea si es la correspondencia que más rompe con la estructura del grafo, es decir, causa que existan más aristas que no comparten ambos grafos. Formalmente,

$$\text{argmin}_{1 \leq i \leq n} |\{j | A_{F_{ij}} = 1 \text{ y } A_{F'_{ij}} = 1, 1 \leq j \leq n\}|$$

Para nuestro ejemplo, solo existe una correspondencia errónea, (u_6, u_6').

3.1.1. Algoritmo de fuerza bruta

Listing 3.1: Algoritmo de fuerza bruta para la *Transformación de Grafos*

```
/*
Entrada:
        FM      conjunto de caracteristicas del modelo.
        FS      conjunto de caracteristicas de la escena.
Salida:
        FM      conjunto de caracteristicas correctas del modelo.
        FS      conjunto de caracteristicas correctas de la escena.
Nota: FM se empareja con FS.
*/

Dist_FM  = calculaMatrizDistancias(FM);
Dist_FS  = calculaMatrizDistancias(FS);

mediana_FM = obtenMediana(Dist_FM);
mediana_FS = obtenMediana(Dist_FS);

AFM = hacergrafoKVecinosMenoresA(Dist_FM,K,mediana_FM);
AFS = hacergrafoKVecinosMenoresA(Dist_FS,K,mediana_FS);


while(AFM != AFS) {
        comun = abs(AFM - AFS);

        v = vertice con mayor cant de unos en su columna
        Elimina de Dist_FM el vertice v
        Elimina de Dist_FS el vertice v

        Elimina de FM el vertice v
        Elimina de FS el vertice v

        AFM = hacergrafoKVecinosMenoresA(Dist_FM,K,mediana_FM);
        AFS = hacergrafoKVecinosMenoresA(Dist_FS,K,mediana_FS);
}

eliminaVerticesDesconectados(AFM, AFS);
```

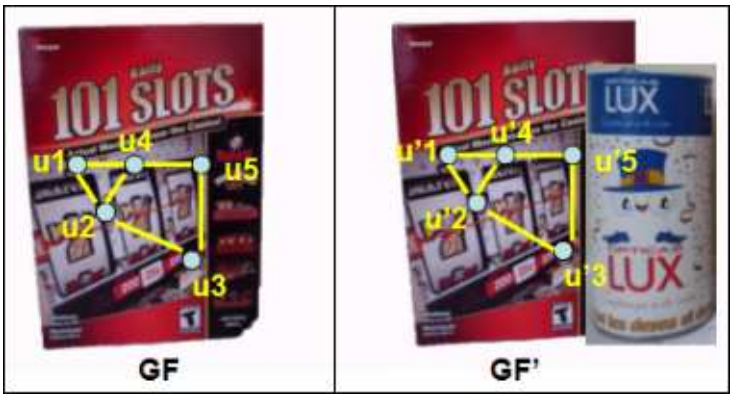


Figura 3.4: Grafos resultantes de eliminar la correspondencia (u_6, u'_6)).

El algoritmo propuesto para la eliminación de las correspondencias erróneas transforma iterativamente los grafos G_F y $G_{F'}$, eliminando en cada iteración la pareja de vértices (u_i, u'_i) de la correspondencia que cumple con la definición 3.1. Una vez eliminado, se vuelven a crear los grafos como en un inicio y se vuelve a iterar hasta que $A_F = A_{F'}$. Una forma sencilla de detectar la correspondencia a eliminar es realizando $Abs(A_F - A_{F'})$. La columna que contenga un mayor número de 1's será la correspondencia a eliminar. El Listado 3.1 muestra el pseudocódigo del algoritmo de fuerza bruta, la cual es una implementación directa y fácil de entender.

En nuestro ejemplo para determinar la primera correspondencia errónea se realiza la resta de las matrices de adyacencia:

$$Abs(A_F - A_{F'}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Aquellas posiciones en donde el resultado dió un 1, indican que esa es una arista que no comparten ambos grafos, y con esto de acuerdo a la definición 3.1, aquel vértice con el mayor número de $1's$ en su columna es la correspondencia que esta causando mayor ruptura estructural. En el ejemplo, es la correspondencia (u_6, u'_6). Siguiendo el algoritmo, se le considera correspondencia errónea, se elimina y se vuelve a construir el grafo repitiendo el proceso. Los nuevos grafos se muestran en la Fig. 3.4.

Ahora las matrices de adyacencia son idénticas:

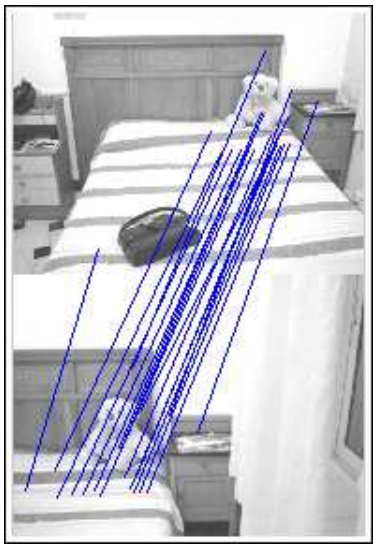


Figura 3.5: Ejemplo de un emparejamiento inicial entre las características de la imagen del modelo (imagen superior) y las de la escena (imagen inferior).

$$A_F = A_{F'} = \begin{array}{c} \\ u1 \\ u2 \\ u3 \\ u4 \\ u5 \end{array} \begin{array}{c} \begin{array}{ccccc} u_1 & u_2 & u_3 & u_4 & u_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

y termina el algoritmo.

La Fig. 3.6 muestra un ejemplo de la transformación de grafos en cada iteración para el emparejamiento inicial presentado en la Fig. 3.5. Así mismo, la Fig. 3.7 muestra la correspondencia final obtenida.

Nótese que por su construcción, los grafos G_S y G_M pueden ser disconexos, en cuyo caso el algoritmo verifica la correctez de las correspondencias locales únicamente. Por la naturaleza del problema del reconocimiento de objetos y de acuerdo a los resultados de los experimentos realizados, es suficiente la verificación local para realizar el reconocimiento. Sin embargo, si se desea realizar una verificación global se tendrá que modificar el algoritmo de tal forma que los grafos no sean disconexos. Una forma sencilla de hacer esto es conectar cada cúmulo con su cúmulo más cercano a través de sus k vértices más cercanos.

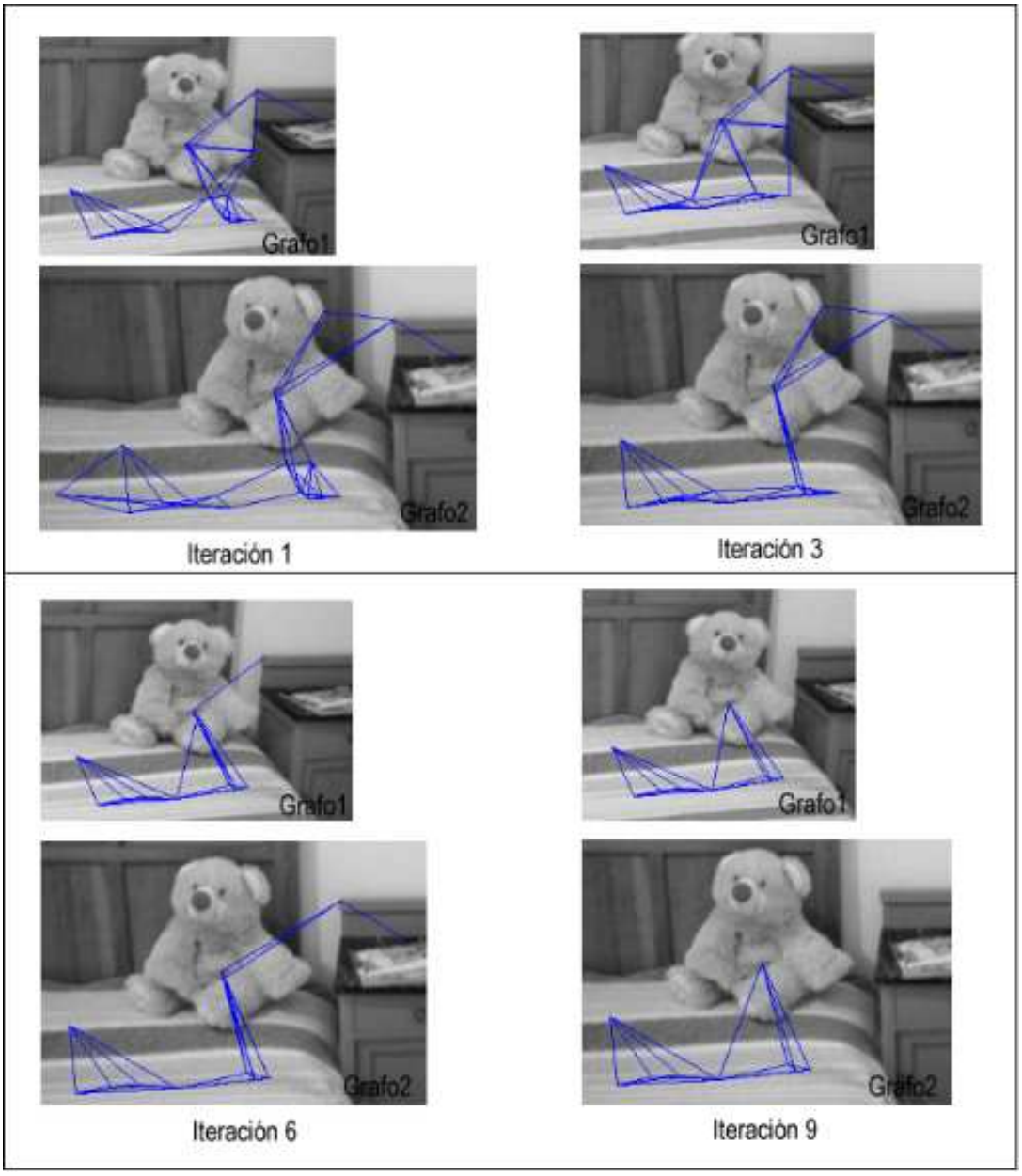


Figura 3.6: Transformación de los grafos durante la ejecución del algoritmo en sus iteraciones 1, 3, 6 y 9.

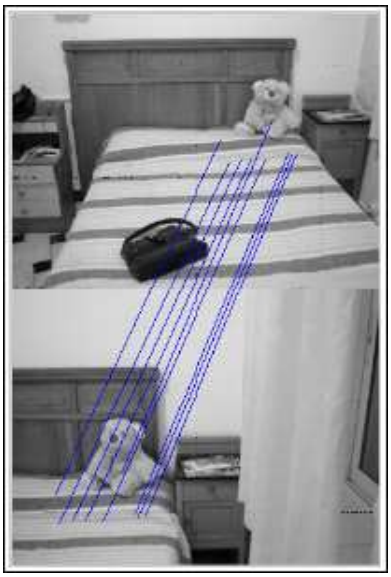


Figura 3.7: Ejemplo del emparejamiento final entre las características de la imagen del modelo y las de la escena.

3.1.1.1. Complejidad

La complejidad en tiempo del algoritmo que calcula la matriz de distancias es $O(n^2)$, en donde n es el número de vértices. Obtener la mediana es $O(n^2 Log(n))$, ya que los elementos se deben ordenar. Para el caso del algoritmo de la creación de los grafos es de $O(n^2 Log(n))$, ya que se deben de encontrar nuevamente los k vecinos más cercanos. Finalmente, la diferencia de matrices es de $O(n^2)$. De esta forma, el algoritmo que elimina las correspondencias erróneas por medio de la transformación de los grafos es de $O(n^3 Log(n))$, ya que a lo más iterará el número de vértices menos k.

3.1.2. Algoritmo optimizado

El algoritmo de fuerza bruta resuelve el problema de la *Transformación de Grafos* de manera directa y sencilla, sin embargo, el orden del algoritmo es cúbico causado principalemente por la necesidad de reconstruir ambos grafos en cada iteración, lo cual como se vió es de orden cuadrático. La fase de inicialización, en la cual se calculan las matrices de distancias, se obtienen las medianas y se construyen por primera vez los dos grafos no es posible optimizarla, por lo cual se mantendrá en $O(n^2 Log(n))$. La parte del algoritmo que sí es posible optimizar corresponde a la búsqueda de la correspondencia errónea y a la reconstrucción del grafo en cada iteración.

En esta propuesta se sustituyen las matrices de adyacencia A_F y $A_{F'}$ por tres nuevas estructuras: una matriz O_F de nxn en la cual los renglones representan las aristas de salida de cada vértice (ordenadas por distancias menor a la mediana y en

donde las primeras k localidades representan las aristas de salida actuales para ese vértice), un arreglo de listas ligadas I_F de dimensión n que contiene las aristas de entrada de cada vértice y un arreglo N_F de dimensión n que guarda el índice del siguiente vértice a conectar en O_F. Asi mismo se definen $O_{F'}$, $I_{F'}$ y $N_{F'}$ pero ahora para las características de la escena.

Siguiendo con el ejemplo de la Fig. 3.3, las nuevas estructuras serían:

$$O_F = \begin{array}{c} \\ u1 \\ u2 \\ u3 \\ u4 \\ u5 \\ u6 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} \mathbf{2} & \mathbf{4} & 5 & 6 & 3 & -2 \\ \mathbf{4} & \mathbf{1} & 3 & 5 & 6 & -2 \\ \mathbf{2} & \mathbf{5} & 4 & 6 & 1 & -2 \\ \mathbf{1} & \mathbf{2} & 5 & 6 & 3 & -2 \\ \mathbf{6} & \mathbf{4} & 3 & 2 & 1 & -2 \\ \mathbf{5} & \mathbf{4} & 1 & 2 & 3 & -2 \end{pmatrix} \end{array} \qquad N_F = \begin{array}{c} \\ u1 \\ u2 \\ u3 \\ u4 \\ u5 \\ u6 \end{array} \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix}$$

$$\boxed{u_1} \longrightarrow 2 \longrightarrow 4 \longrightarrow NULL$$

$$\boxed{u_2} \longrightarrow 1 \longrightarrow 3 \longrightarrow 4 \longrightarrow NULL$$

$$\boxed{u_3} \longrightarrow NULL$$

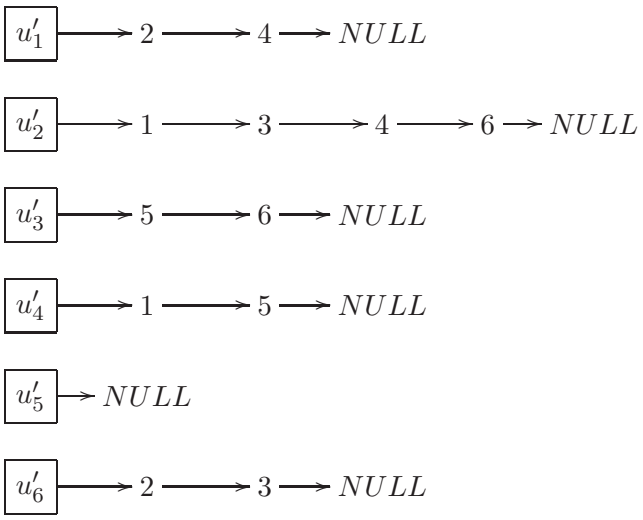
$$\boxed{u_4} \longrightarrow 1 \longrightarrow 2 \longrightarrow 5 \longrightarrow 6 \longrightarrow NULL$$

$$\boxed{u_5} \longrightarrow 3 \longrightarrow 6 \longrightarrow NULL$$

$$\boxed{u_6} \longrightarrow 5 \longrightarrow NULL$$

$$I_F$$

$$O_{F'} = \begin{array}{c} \\ u'1 \\ u'2 \\ u'3 \\ u'4 \\ u'5 \\ u'6 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} \mathbf{2} & \mathbf{4} & 6 & 5 & 3 & -2 \\ \mathbf{6} & \mathbf{1} & 4 & 3 & 5 & -2 \\ \mathbf{6} & \mathbf{2} & 5 & 4 & 1 & -2 \\ \mathbf{1} & \mathbf{2} & 5 & 6 & 3 & -2 \\ \mathbf{4} & \mathbf{3} & 2 & 1 & 6 & -2 \\ \mathbf{2} & \mathbf{3} & 1 & 4 & 5 & -2 \end{pmatrix} \end{array} \qquad N_{F'} = \begin{array}{c} \\ u'1 \\ u'2 \\ u'3 \\ u'4 \\ u'5 \\ u'6 \end{array} \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 3 \\ 3 \end{pmatrix}$$



$I_{F'}$

En las matrices O_F y $O_{F'}$ se marca el final de los siguientes posibles vértices a conectar con el valor -2. La unión de las aristas de salida definidas en las primeras k columnas de O_F con las aristas de entrada definidas en I_F dan como resultado la misma matriz A_F que se utilizó en el algoritmo de fuerza bruta. De la misma manera sucede con $A_{F'}$. Para encontrar la correspondencia errónea a eliminar (equivalente a la resta de matrices del algoritmo de fuerza bruta) simplemente se unen las k aristas de la iésima entrada de O_F con las aristas definidas en la iésima entrada de I_F y se comparan con la unión de sus respectivas correspondencias en $O_{F'}$ e $I_{F'}$. La correspondencia errónea a eliminar será aquella que comparta menos aristas. En nuestro ejemplo se obtienen los siguientes conjuntos provenientes de las uniones:

Aristas de $u_1 =$	{2,4}	Aristas de $u'_1 =$	{2,4}	difieren en **0** aristas
Aristas de $u_2 =$	{1,3,4}	Aristas de $u'_2 =$	{1,3,4,6}	difieren en **1** aristas
Aristas de $u_3 =$	{2,5}	Aristas de $u'_3 =$	{2,5,6}	difieren en **1** aristas
Aristas de $u_4 =$	{1,2,5,6}	Aristas de $u'_4 =$	{1,2,5}	difieren en **1** aristas
Aristas de $u_5 =$	{3,4,6}	Aristas de $u'_5 =$	{3,4}	difieren en **1** aristas
Aristas de $u_6 =$	{4,5}	Aristas de $u'_6 =$	{2,3}	difieren en **4** aristas

Debido a que la correspondencia (u_6, u'_6) tiene más aristas en las que difiere, entonces se le considera como la correspondencia errónea a eliminar.

Para eliminar un vértice u_i y volver a crear el grafo con los k vecinos más cercanos, se tienen que realizar básicamente los siguientes 3 pasos:

1. Eliminar de I_F todas las ocurrencias de u_i.
2. Eliminar definitivamente el vértice u_i de O_F.

3. Para cada ocurrencia de u_i en O_F (en sus k primeras columnas), eliminar la arista y reconectarlo con el siguiente vértice más cercano (actualizando las entradas respectivas en I_F).

El paso 1 corresponde a eliminar todas las aristas que llegaban de otros vértices a u_i, el paso 2 corresponde a eliminar todas las aristas que salían de u_i así como a eliminarlo definitivamente del grafo y el paso 3 corresponde a volver a crear todo el grafo con los k vecinos más cercanos. Se realiza lo mismo pero ahora para las estructuras de datos correspondientes a la escena.

El Listado 3.2 presenta el algoritmo optimizado que incorpora estos tres pasos.

Listing 3.2: Algoritmo optimizado para la *Transformación de Grafos*

```
/*
Entrada:
        FM      conjunto de caracteristicas del modelo.
        FS      conjunto de caracteristicas de la escena.
Salida:
        FM      conjunto de caracteristicas correctas del modelo.
        FS      conjunto de caracteristicas correctas de la escena.
Nota: FM se empareja con FS.
*/

Dist_FM  = calculaMatrizDistancias(FM);
Dist_FS  = calculaMatrizDistancias(FS);

mediana_FM = obtenMediana(Dist_FM);
mediana_FS = obtenMediana(Dist_FS);

OFM = hacerGrafoOutKVecinosMenoresA(Dist_FM, K, mediana_FM);
OFS = hacerGrafoOutKVecinosMenoresA(Dist_FS, K, mediana_FS);

IFM = hacerGrafoInKVecinosMenoresA(OFM, K);
IFS = hacerGrafoInKVecinosMenoresA(OFS, K);

NM = hacerArregloSiguientesIndices();
NS = hacerArregloSiguientesIndices();

num_dif_aristas = hacerArregloAristasDiferentes();
sum_dif_aristas = sumaAristasDiferentes(num_dif_aristas);


while(sum_dif_aristas != 0) {
    corresp_erronea = obtenIndiceValorMaximo(num_dif_aristas);

    EliminaOcurrenciasEnI(corresp_erronea, OFM, IFM, K);
    EliminaOcurrenciasEnI(corresp_erronea, OFS, IFS, K);
```

```
        OFM[corresp_erronea][1] = -1;
        OFS[corresp_erronea][1] = -1;

        ReconectaElGrafo(corresp_erronea, OFM, IFM, K, NM);
        ReconectaElGrafo(corresp_erronea, OFS, IFS, K, NS);

        num_dif_aristas = hacerArregloAristasDiferentes();
        sum_dif_aristas = sumaAristasDiferentes(num_dif_aristas);
}

eliminaVerticesDesconectados(OFM, OFS);
FM = obtenCaracteristicasResultantes(OFM);
FS = obtenCaracteristicasResultantes(OFS);
```

El Listado 3.3 y el Listado 3.4 presentan los algoritmos correspondientes a las funciones auxiliares *EliminaOcurrenciasEnI* y *ReconectaElGrafo*. La función auxiliar *obtenCaracteristicasResultantes* lo único que hace es eliminar de FM y FS aquellas características que tengan el valor de -1 en su entrada correspondiente en OFM y OFS, respectivamente.

Listing 3.3: Función auxiliar *EliminaOcurrenciasEnI*

```
/*
Entrada:
        corresp_erronea indice del vertice considerado como
                        correspondencia erronea
        OF  conjunto de aristas de salida del grafo

        IF  conjunto de aristas de entrada del grafo
        K   conectividad
Salida:
        IF  conjunto de caracteristicas correctas del modelo
        sin referencias a corresp_erronea
*/

for(i=1; i<=K; i++) {
  if(OF[corresp_erronea][i]>=0)
    eliminaDeInVertice(corresp_erronea,IF[OF[corresp_erronea][i]]);
}
```

La función auxiliar *eliminaDeInVertice* básicamente consiste en recorrer la lista ligada de la entrada indicada en IF y eliminar el nodo con el valor de la correspondencia errónea.

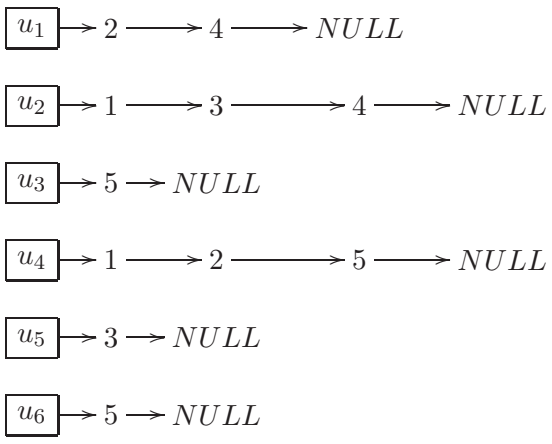
La función auxiliar *desconectaVerticesMarcados* consiste en desconectar por completo a los vértices que se encuentran en la lista vertices_desconectar. Esto lo hace eliminando todas las ocurrencias del vértice a desconectar en IF y en las primeras K columnas de OF. En OF se desconecta el vértice colocando el valor de -3 en cada una de sus ocurrencias.

Listing 3.4: Función auxiliar *ReconectaElGrafo*

```
/*
Entrada:
        corresp_erronea indice del vertice considerado como
                        correspondencia erronea
        OF  conjunto de aristas de salida del grafo

        IF  conjunto de aristas de entrada del grafo
        K   conectividad
        N   arreglo con indices a OF del siguiente vertice
            mas cercano
Salida:
        IF  conjunto de caracteristicas correctas del modelo
        sin referencias a corresp_erronea
*/


vertices_desconectar = lista ligada vacia;

Por cada vertice en la lista IF de la entrada IF[corresp_erronea] {
    vertice = vertice actual en la lista ligada
    indice = indice en OF[vertice] que tiene el valor de
             corresp_erronea (dentro de sus primeras k columnas)
    // Busca el siguiente vertice mas cercano
    j = N[vertice];
    while(OF[vertice][j]!= -2 && OF[OF[vertice][j]][1] == -1)
      j++;
    if(OF[vertice][j]==-2)  //Ya no hay k vertices mas cercanos
      agregar en vertices_desconectar el valor vertice
    else {
      // Actualizo IF para la nueva arista
      agrega un nuevo nodo con el valor vertice en la
      lista IF[OF[vertice][j]];

      // Reconecta
      OF[vertice][indice] = OF[vertice][j];
      N[vertice] = j+1;
    }
}

desconectaVerticesMarcados(vertices_eliminar, OF, OI, K);
```

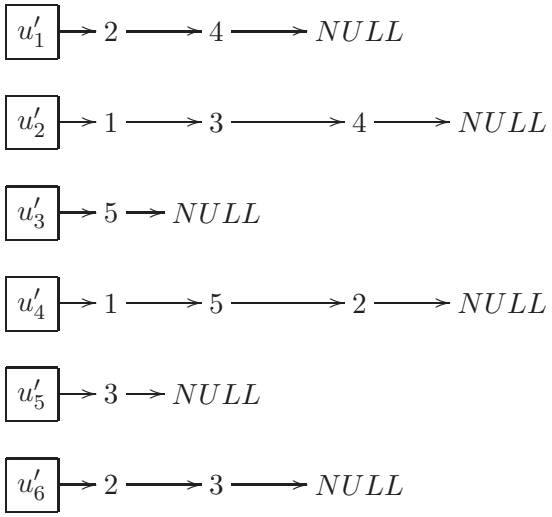
Después de realizar una iteración del algoritmo, las estructuras de datos de ambos grafos quedan como sigue:

$$O_F = \begin{array}{c} \\ u1 \\ u2 \\ u3 \\ u4 \\ u5 \\ u6 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} \mathbf{2} & \mathbf{4} & 5 & 6 & 3 & -2 \\ \mathbf{4} & \mathbf{1} & 3 & 5 & 6 & -2 \\ \mathbf{2} & \mathbf{5} & 4 & 6 & 1 & -2 \\ \mathbf{1} & \mathbf{2} & 5 & 6 & 3 & -2 \\ \mathbf{3} & \mathbf{4} & 3 & 2 & 1 & -2 \\ \mathbf{-1} & \mathbf{4} & 1 & 2 & 3 & -2 \end{pmatrix} \end{array} \qquad N_F = \begin{array}{c} \\ u1 \\ u2 \\ u3 \\ u4 \\ u5 \\ u6 \end{array} \begin{array}{c} \\ \begin{pmatrix} 3 \\ 3 \\ 3 \\ 3 \\ 4 \\ 3 \end{pmatrix} \end{array}$$



I_F

$$O_{F'} = \begin{array}{c} \\ u'1 \\ u'2 \\ u'3 \\ u'4 \\ u'5 \\ u'6 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathbf{1} & \mathbf{2} & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} \mathbf{2} & \mathbf{4} & 6 & 5 & 3 & -2 \\ \mathbf{4} & \mathbf{1} & 4 & 3 & 5 & -2 \\ \mathbf{5} & \mathbf{2} & 5 & 4 & 1 & -2 \\ \mathbf{1} & \mathbf{2} & 5 & 6 & 3 & -2 \\ \mathbf{4} & \mathbf{3} & 2 & 1 & 6 & -2 \\ \mathbf{-1} & \mathbf{3} & 1 & 4 & 5 & -2 \end{pmatrix} \end{array} \qquad N_{F'} = \begin{array}{c} \\ u'1 \\ u'2 \\ u'3 \\ u'4 \\ u'5 \\ u'6 \end{array} \begin{array}{c} \\ \begin{pmatrix} 3 \\ 4 \\ 4 \\ 3 \\ 3 \\ 3 \end{pmatrix} \end{array}$$



$I_{F'}$

Ahora los grafos son idénticos y el algoritmo termina. Las aristas de ambos grafos son:

Aristas de $u_1 =$	{2,4}	Aristas de $u'_1 =$	{2,4}	difieren en **0** aristas
Aristas de $u_2 =$	{1,3,4}	Aristas de $u'_2 =$	{1,3,4}	difieren en **0** aristas
Aristas de $u_3 =$	{2,5}	Aristas de $u'_3 =$	{2,5}	difieren en **0** aristas
Aristas de $u_4 =$	{1,2,5}	Aristas de $u'_4 =$	{1,2,5}	difieren en **0** aristas
Aristas de $u_5 =$	{3,4}	Aristas de $u'_5 =$	{3,4}	difieren en **0** aristas

Los resultados obtenidos de ésta optimización se presentan en el siguiente capítulo.

3.2. Síntesis

En este capítulo se presentó y se analizó la complejidad del algoritmo de *Transformación de Grafos*, el cual es la principal aportación de esta tesis. El algoritmo consiste en ir transformando simultáneamente el grafo correspondiente al modelo y el correspondiente a la escena para quedarse finalmente solo con las correspondencias correctas. La complejidad del algoritmo de fuerza bruta es $O(n^3 Log(n))$. Al final del capítulo se presentó un versión optimizada del algoritmo de fuerza bruta.

En el siguiente capítulo se presentan los resultados de las pruebas realizadas.

Capítulo 4

Resultados

Se diseñaron cuatro experimentos diferentes para poner a prueba el método de *Transformación de Grafos*. El primero tiene como objetivo comparar su eficacia y eficiencia contra otro método estructural llamado *Softassign* [24]. El segundo hace una comparación contra un método de alineación llamado *RANSAcing Thin-plate Splines*, el cual está basado en los *Thin-plate Splines* presentados en [21]. El tercero consiste en probar el método para reconocer objetos y escenas bajo efectos de oclusión parcial. Finalmente, el cuarto realiza pruebas de eficiencia de dos implementaciones del algoritmo de fuerza bruta (en *Matlab* y en *C*) presentado en la sección 3.1.1 contra la implementación en *C* del algoritmo optimizado presentado en la sección 3.1.2.

En las últimas dos secciones se presentan dos casos en los que se detecto que el algoritmo no funciona correctamente y una galería de imágenes de otros resultados.

Las imágenes de prueba tienen una resolución de $640X480$ pixeles. Una parte de las imágenes se tomo con una cámara digital a objetos y escenas. Otro conjunto de imágenes se tomo de una base de datos pública presentada en [22]. Finalmente, el conjunto de imágenes de fondo de ojo fué proporcionada por la Dra. María Elena Martínez [1]. Es importante notar que las imágenes seleccionadas para las pruebas son imágenes a ser utilizadas en diferentes aplicaciones: reconocimiento de objetos, reconocimiento de escenas, construcción de mosaicos con imágenes de fondo de ojo y estimación de movimiento. En las siguientes secciones de éste capítulo se detallan los experimentos y se muestran los resultados obtenidos, en donde las tablas comparativas muestran el número de correspondencias correctas (iniciales y finales) que se verificaron visualmente una a una y en donde los tiempos reportados corresponden a implementaciones en *Matlab* de los diferentes algoritmos, ésto para que los resultados sean comparables.

4.1. Transformación de Grafos vs SoftAssign

Se realizaron experimentos para comparar su eficacia y eficiencia con respecto a otro método estructural llamado *Softassign* en sus versiones que usan *kernels* y *costos*,

[1] Investigadora del IIMAS (Instituto de Matemáticas Aplicadas y Sistemas), UNAM

el cual se describió en la sección 2.1.4.1. El valor del costo de un emparejamiento esta dado por:

$$e^{-dist/\sigma}$$

en donde, *dist* es la distancia euclidiana en el espacio de dimensión 128 entre las características del modelo y de la escena y σ es una constante de escalamiento que en todos los experimentos tomó el valor 800, ya que se observó experimentalmente que este valor daba buenos resultados.

En estos experimentos se utilizó el extractor de características de *SIFT* y el emparejamiento inicial es resultado del algoritmo *BBF*, ambos descritos en la sección 1.5.4.2. La implementación de los tres algoritmos se realizó en *Matlab*.

Se probó con 12 parejas de imágenes, 8 de las cuales representaban imágenes de escenas y 4 de objetos, dando resultados similares en todos los casos. Tres de éstos se reportan a continuación.

4.1.1. Objeto *Tasa*

La primer pareja de imágenes esta formada por una tasa que fue tomada de frente (el modelo) y una escena en la que aparece únicamente la misma tasa pero con un cambio de punto de vista de 45 grados. La Fig. 4.1 muestra las correspondencias iniciales resultantes del algoritmo *BBF*. La Fig. 4.2 muestra las correspondencias encontradas por el algoritmo de *Softassign* usando *kernels* (a), *Softassign* usando *kernels* y *costos* (b) y *Transformación de Grafos* (c). Finalmente, la Fig. 4.3 muestra el grafo encontrado en ambas imágenes.

El Cuadro 4.1 muestra un comparativo de los resultados obtenidos. En sus columnas muestra el nombre del algoritmo, el número de correspondencias iniciales (CI), el número de correspondencias iniciales correctas (CIC), el número de correspondencias iniciales erróneas (CIE), el número de correspondencias finales (CF), el número de correspondencias finales correctas (CFC), el número de correspondencias finales erróneas (CFE), el número de iteraciones y el tiempo en segundos.

4.1.2. Objeto *Poster*

Esta pareja de imágenes consiste en la imagen de un poster y una escena en donde aparece el mismo poster pero desde otro punto de vista arbitrario. Estas imágenes son interesantes, ya que al lado del poster que se desea reconocer se encuentra otro muy similar en sus letreros y en su logotipo. El reto consistía en que el método no se confundiera y emparejara algunas características con el primer poster y otras con el segundo.

La Fig. 4.4 muestra las correspondencias iniciales resultantes del algoritmo *BBF*. La Fig. 4.5 muestra las correspondencias encontradas por el algoritmo de *Softassign* usando *kernels* (a), *Softassign* usando *kernels* y *costos* (b) y *Transformación de Grafos* (c). Finalmente, la Fig. 4.6 muestra el grafo encontrado en ambas imágenes.

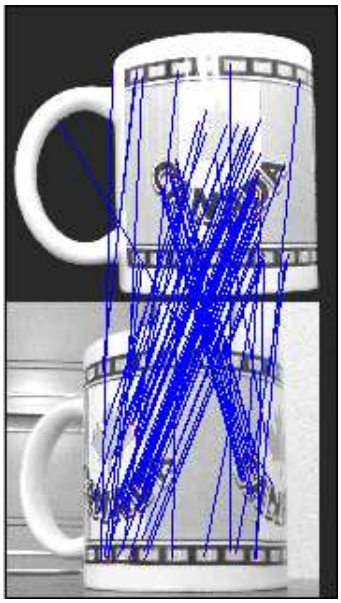


Figura 4.1: Emparejamiento inicial obtenido *BBF*, del objeto *tasa* entre su modelo (imagen superior) y la escena (imagen inferior).

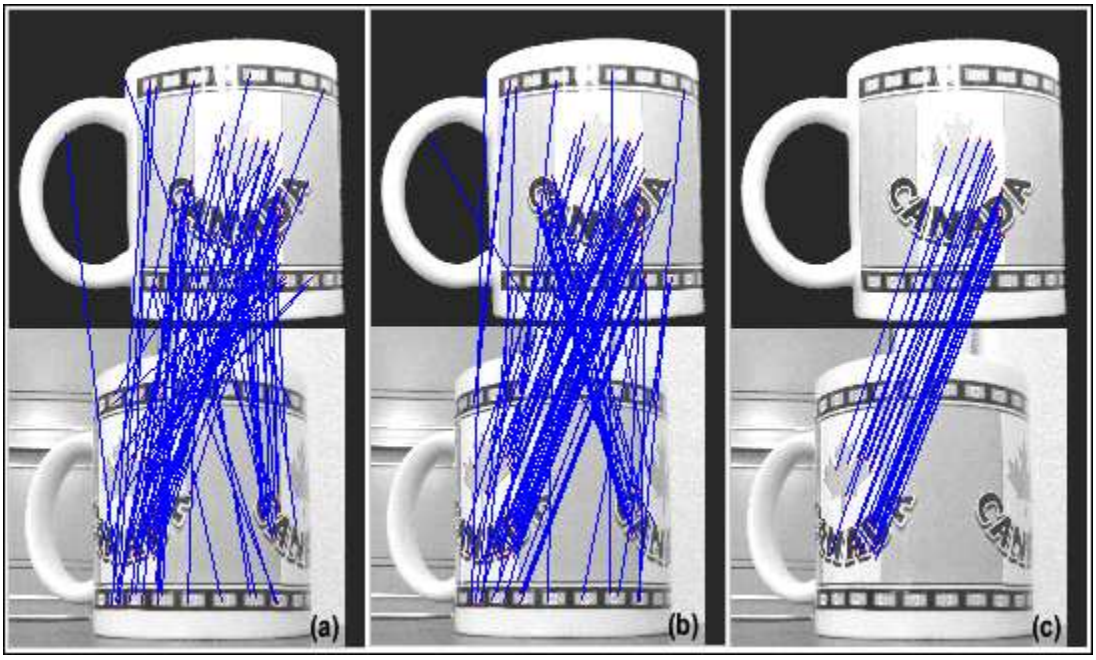


Figura 4.2: Emparejamientos resultantes del objeto *tasa* con los métodos: (a)*Softassign* con *kernels*, (b) *Softassign* con *kernels* + *costos* y (c) *Transformación de Grafos*.

Algoritmo	**CI**	**CIC**	**CIE**	**CF**	**CFC**	**CFE**	**Iter.**	**Seg**
Softassign + kernel	74	36	38	74	3	71	242	5.57
Softassign + kernel + costos	74	36	38	74	36	38	85	.4
Transformacion de Grafos	74	36	38	28	28	0	47	.012

Cuadro 4.1: Comparativo de métodos estructurales aplicados al objeto *tasa*: algoritmo, número de correspondencias iniciales obtenidas con *BBF* (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg.).

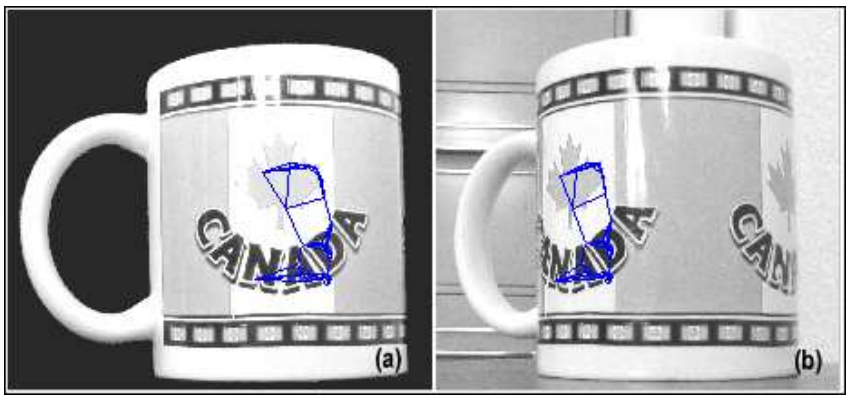


Figura 4.3: Grafos resultantes del objeto *tasa*: (a)Imagen del modelo, (b)Imagen de la escena.

El Cuadro 4.2 muestra un comparativo de los resultados obtenidos. En sus columnas muestra el nombre del algoritmo, el número de correspondencias iniciales (CI), el número de correspondencias iniciales correctas (CIC), el número de correspondencias iniciales erróneas (CIE), el número de correspondencias finales (CF), el número de correspondencias finales correctas (CFC), el número de correspondencias finales erróneas (CFE), el número de iteraciones y el tiempo en segundos.

4.1.3. Escena *Recámara*

Los dos experimentos anteriores se trataron del reconocimiento de objetos, en este caso se experimentó con imágenes a ser usadas en el reconocimiento de escenas.

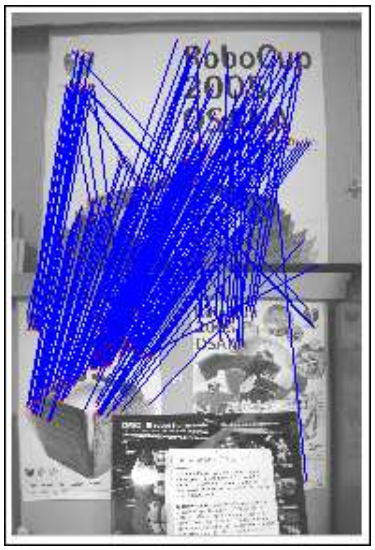


Figura 4.4: Emparejamiento inicial obtenido con *BBF*), del objeto *poster1* entre su modelo (imagen superior) y la escena (imagen inferior).

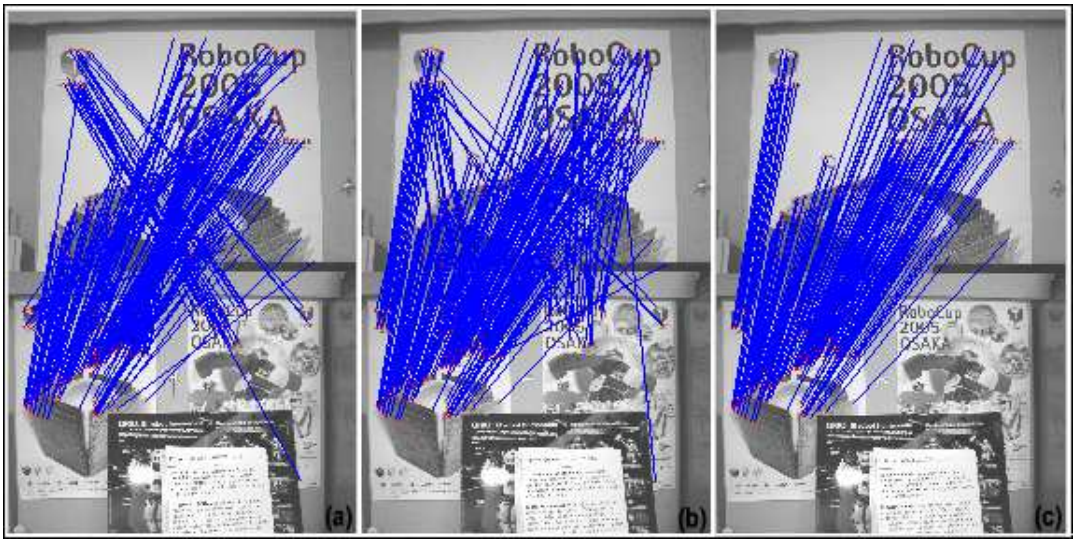


Figura 4.5: Emparejamientos resultantes del objeto *poster1* con los métodos: (a)*Softassign* con *kernels*, (b) *Softassign* con *kernels* + *costos* y (c) *Transformación de Grafos*.

Algoritmo	CI	CIC	CIE	CF	CFC	CFE	Iter.	Seg
Softassign + kernel	217	177	40	217	76	141	250	41.40
Softassign + kernel + costos	217	177	40	217	170	47	85	10.46
Transformacion de Grafos	217	177	40	170	170	0	48	1.15

Cuadro 4.2: Comparativo de métodos estructurales aplicados al objeto *poster*: algoritmo, número de correspondencias iniciales obtenidas con *BBF* (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg.).

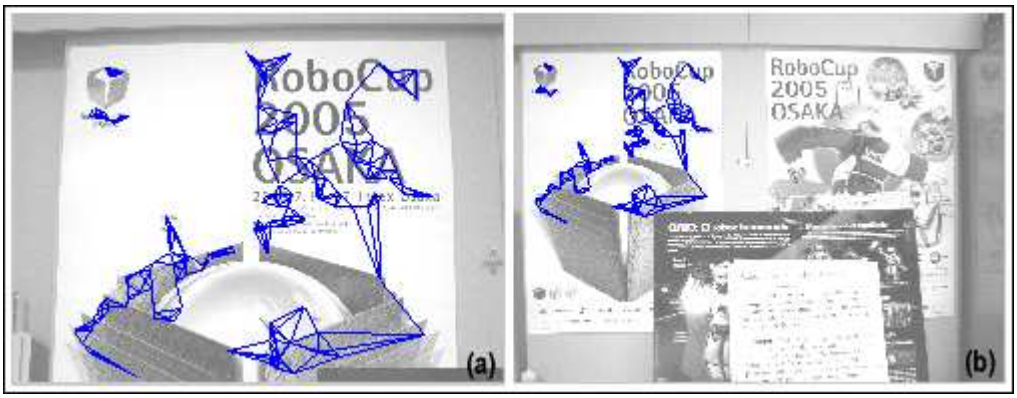


Figura 4.6: Grafos resultantes del objeto *poster1*: (a)Imagen del modelo, (b)Imagen de la escena.

La escena del modelo consiste en una toma frontal de una recámara. La nueva escena es una toma de la misma recámara pero desde un punto de vista diferente.

La Fig. 4.7 muestra las correspondencias iniciales resultantes del algoritmo *BBF*. La Fig. 4.8 muestra las correspondencias encontradas por el algoritmo de *Softassign* usando *kernels* (a), *Softassign* usando *kernels* y *costos* (b) y *Transformación de Grafos* (c). Finalmente, la Fig. 4.9 muestra el grafo encontrado en ambas imágenes.

El Cuadro 4.3 muestra un comparativo de los resultados obtenidos. En sus columnas muestra el nombre del algoritmo, el número de correspondencias iniciales (CI), el número de correspondencias iniciales correctas (CIC), el número de correspondencias iniciales erróneas (CIE), el número de correspondencias finales (CF), el número de correspondencias finales correctas (CFC), el número de correspondencias finales erróneas (CFE), el número de iteraciones y el tiempo en segundos.

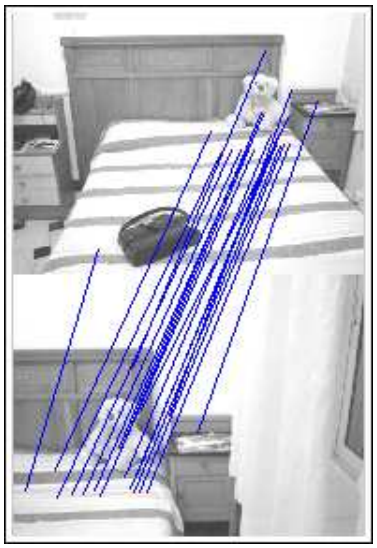


Figura 4.7: Emparejamiento inicial obtenido con *BBF*, de la escena *recámara* entre su modelo (imagen superior) y la escena (imagen inferior).

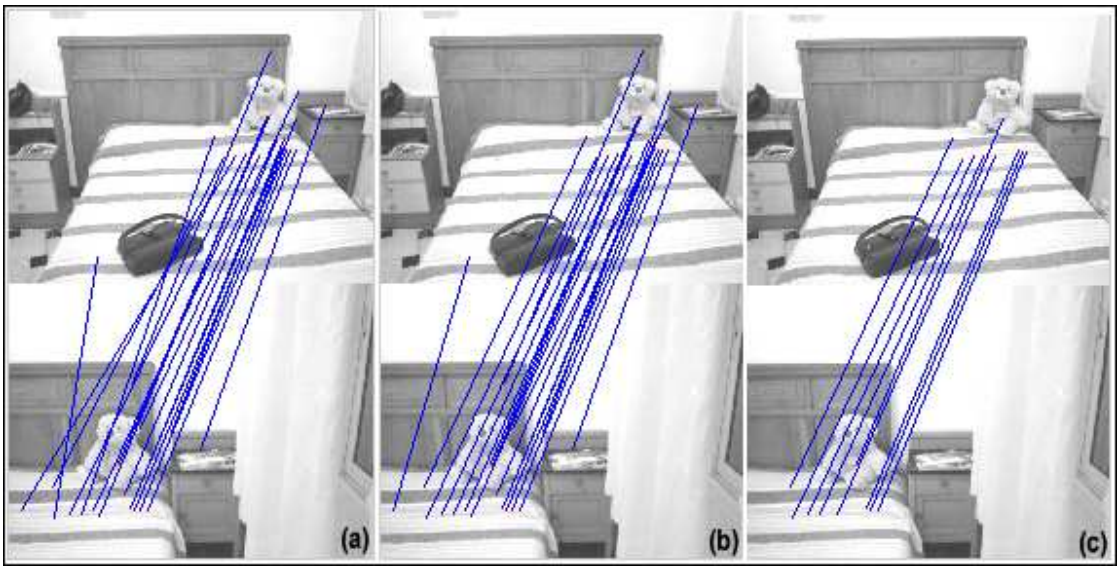


Figura 4.8: Emparejamientos resultantes de la escena *recámara* con los métodos: (a)*Softassign* con *kernels*, (b) *Softassign* con *kernels* + *costos* y (c) *Transformación de Grafos*.

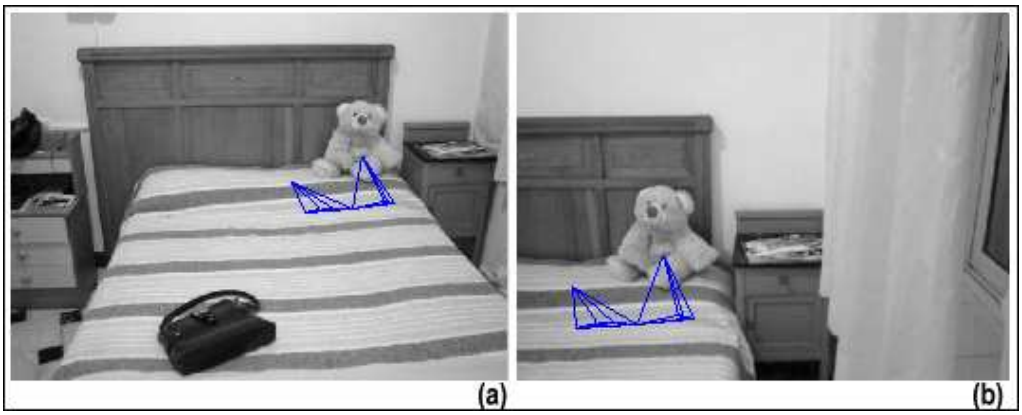


Figura 4.9: Grafos resultantes de la escena *recámara*: (a)Imagen del modelo, (b)Imagen de la escena.

Algoritmo	**CI**	**CIC**	**CIE**	**CF**	**CFC**	**CFE**	**Iter.**	**Seg**
Softassign + kernel	19	17	2	19	12	7	144	.28
Softassign + kernel + costos	19	17	2	19	17	2	85	0.15
Transformacion de Grafos	19	17	2	9	9	0	9	.01

Cuadro 4.3: Comparativo de métodos estructurales aplicados al objeto *recámara*: algoritmo, número de correspondencias iniciales obtenidas con *BBF* (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg.).

4.1.4. Discusión

Los resultados mostraron que el algoritmo de *Transformación de Grafos* es superior en eficacia y eficiencia a *Softassign*, esto posiblemente gracias a que supone que el emparejamiento inicial contiene algunas correspondencias erróneas que debe encontrar y eliminar. En cambio, el algoritmo de *Softassign* trata de maximizar el número de rectángulos sin tomar en cuenta el emparejamiento inicial, lo cual mostró no dar resultado para este tipo de imágenes.

Nótese las diferencias en cuanto al número de iteraciones realizadas y al tiempo. En todos los casos le costo menos iteraciones y tiempo al algoritmo de *Transformación de Grafos* el encontrar una solución, la cual siempre fue correcta a diferencia de *Softassign* que en todos los casos dió resultados muy malos.

4.2. Transformación de Grafos vs RANSAcing Thin-plate Splines

Este experimento consistió en comparar el algoritmo estructural de *Transformación de Grafos* contra un método de alineación. El método de alineación elegido fué el llamado RANSAcing Thin-plate Splines. Este algoritmo esta basado en los Thin-plate Splines presentados en [21]. Consiste en iniciar con un conjunto de posibles correspondencias, que probablemente contenga un número desconocido de correspondencias erróneas (que éste método las define como correspondencias que rompen la suavidad de la transformación no rígida). El algoritmo consiste en ir eliminando las correspondencias erróneas incrementalmente usando una aproximación tipo RANSAC. Se dice que es de tipo incremental porque consiste de varias etapas, en las cuales el tamaño de la muestra se va incrementando hasta que se encuentra el tamaño óptimo. Se utilizaron para ambos algoritmos implementaciones en *Matlab*.

Para este experimento se utilizó un detector de características muy simple basado en el gradiente (puntos que son máximos locales de gradiente de intensidad en una ventana de radio 10) y una correlación para obtener el emparejamiento inicial. Al resultado de la correlación se le aplicó un filtro al que se le llamo *filtro calidad correlación* y que se explica más adelante con detalle en la sección **??**. Se realizaron pruebas con 10 parejas de imágenes a ser usadas en estimación de movimiento, de las cuales se reportan los resultados de 3 de ellas a continuación.

4.2.1. Pareja *DOS*

La primer pareja de imágenes consiste de un movimiento de acercamiento más una pequeña translación. La Fig. 4.10 muestra la imagen del movimiento, en (a) las correspondencias iniciales obtenidas con correlación, (b) las correspondencias finales del algoritmo *RANSAcing Thin-plate Splines* y (c) las correspondencias finales del algoritmo de *Transformación de Grafos*. La imagen de movimiento se genera como la suma de la imagen en el tiempo t más la imagen en el tiempo $t+1$. Las líneas se dibujan desde la coordenada del pixel del punto característico en el tiempo t hasta la coordenada en la que quedo ese mismo pixel (según el algoritmo) en el tiempo $t+1$. La Fig. 4.11 muestra los grafos resultantes. El Cuadro 4.4 presenta el comparativo de los resultados de ambos algoritmos. En sus columnas muestra: nombre del algoritmo, número de correspondencias iniciales (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos.

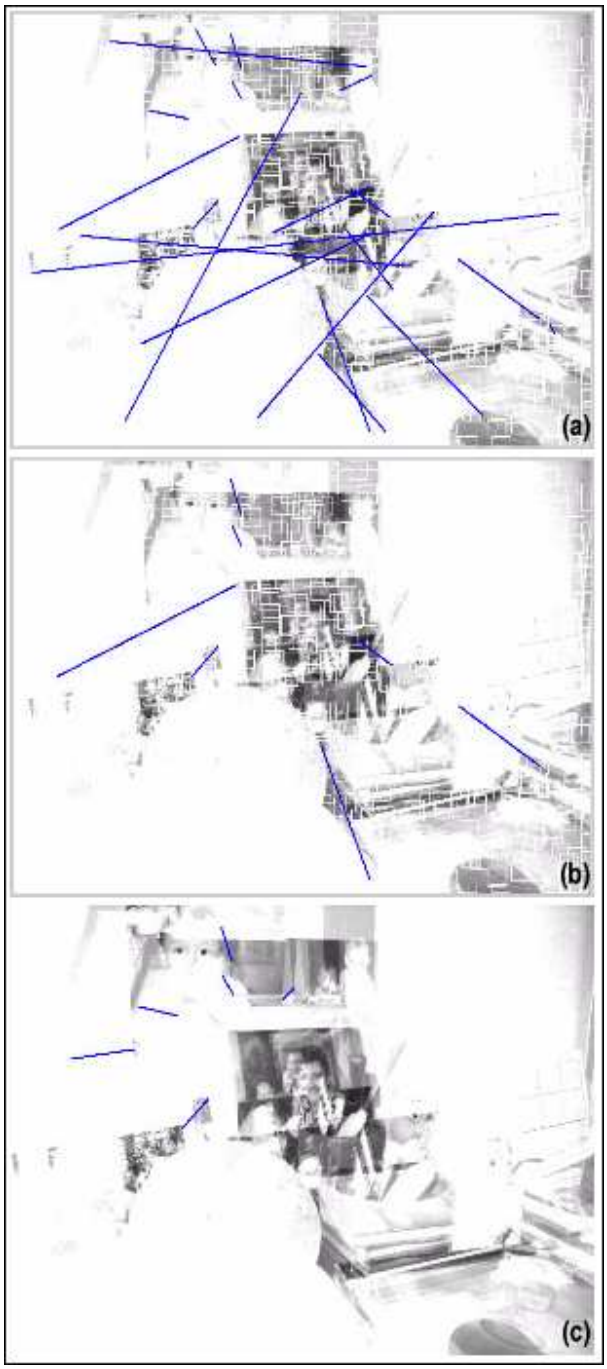


Figura 4.10: Pareja *DOS*, imágenes de movimiento. (a) Entrada obtenida con correlación, (b) Resultado del algoritmo de *RANSAcing Thin-plate Splines* y (c) Resultado del algoritmo de *Transformación de Grafos*.

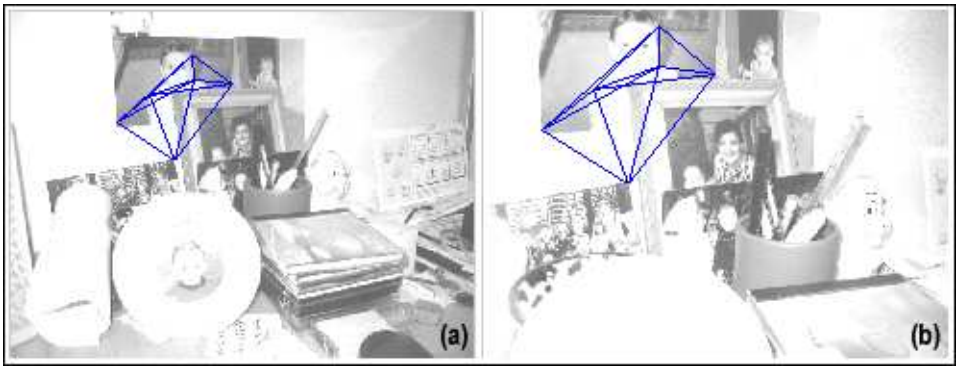


Figura 4.11: Grafos de la pareja *DOS*. (a) Grafo sobre la escena en el tiempo t y (b) Grafo sobre la escena en el tiempo $t+1$

Algoritmo	**CI**	**CIC**	**CIE**	**CF**	**CFC**	**CFE**	**Iter.**	**Seg**
RANSAcing Thin-plate Splines	24	12	12	7	0	7	100	.84
Transformación de grafos	24	12	12	6	6	0	16	.12

Cuadro 4.4: Comparativo de algoritmo estructural vs de alineación con la pareja *DOS*: algoritmo, número de correspondencias iniciales obtenidas con correlación (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg.).

4.2.2. Pareja *CINCO*

La segunda pareja de imágenes consiste de un movimiento de translación en diagonal. La Fig. 4.12 muestra la imagen del movimiento, en (a) las correspondencias iniciales obtenidas con correlación, (b) las correspondencias finales del algoritmo *RANSAcing Thin-plate Splines* y (c) las correspondencias finales del algoritmo de *Transformación de Grafos*. La Fig. 4.13 muestra los grafos resultantes. El Cuadro 4.5 presenta el comparativo de los resultados de ambos algoritmos. En sus columnas muestra: nombre del algoritmo, número de correspondencias iniciales (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos.

Algoritmo	**CI**	**CIC**	**CIE**	**CF**	**CFC**	**CFE**	**Iter.**	**Seg**
RANSAcing Thin-plate Splines	42	33	9	13	12	1	100	1.37
Transformación de Grafos	42	33	9	26	23	3	17	.01

Cuadro 4.5: Comparativo de algoritmo estructural vs de alineación con la pareja *CINCO*: algoritmo, número de correspondencias iniciales obtenidas con correlación (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg.).

4.2.3. Pareja *NUEVE*

Finalmente, la tercer pareja de imágenes seleccionadas consiste de un movimiento de acercamiento con una pequeña translación. La Fig. 4.14 muestra la imagen del movimiento, en (a) las correspondencias iniciales obtenidas con correlación, (b) las correspondencias finales del algoritmo *RANSAcing Thin-plate Splines* y (c) las correspondencias finales del algoritmo de *Transformación de Grafos*. La Fig. 4.15 muestra los grafos resultantes. El Cuadro 4.6 presenta el comparativo de los resultados de ambos algoritmos. En sus columnas muestra: nombre del algoritmo, número de correspondencias iniciales (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos.

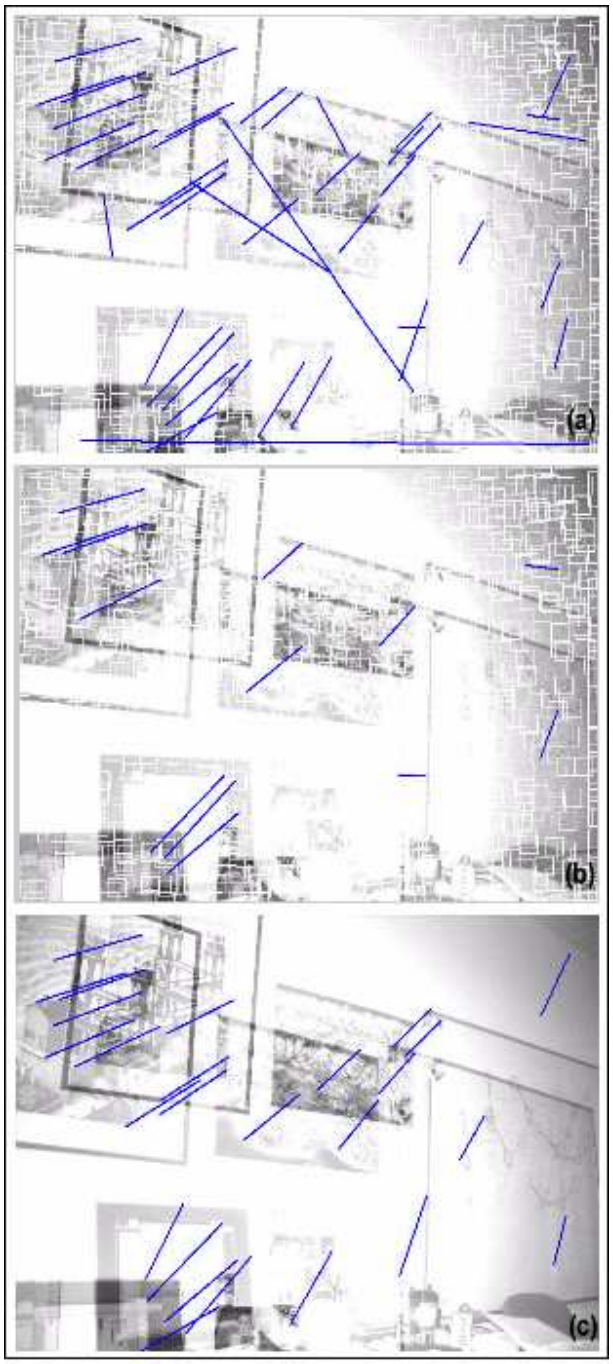


Figura 4.12: Pareja *CINCO*, imágenes de movimiento. (a) Entrada obtenida con correlación, (b) Resultado del algoritmo de *RANSAcing Thin-plate Splines* y (c) Resultado del algoritmo de *Transformación de Grafos*.

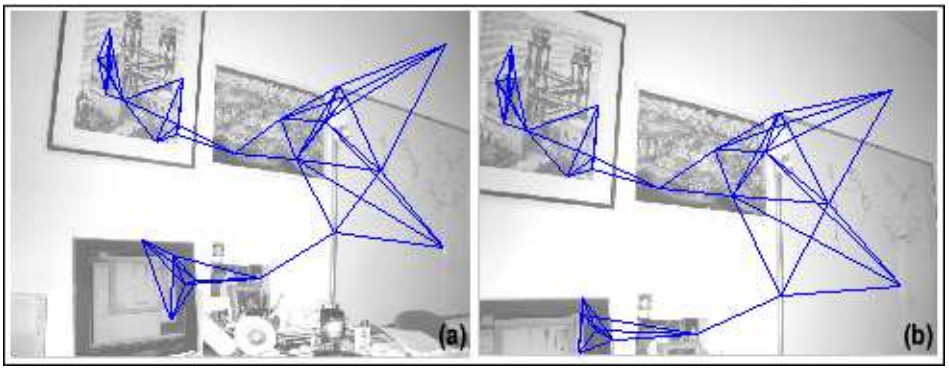


Figura 4.13: Grafos de la pareja *CINCO*. (a) Grafo sobre la escena en el tiempo t y (b) Grafo sobre la escena en el tiempo $t+1$

Algoritmo	**CI**	**CIC**	**CIE**	**CF**	**CFC**	**CFE**	**Iter.**	**Seg**
RANSAcing Thin-plate Splines	37	32	5	12	12	0	100	1.23
Transformación de Grafos	37	32	5	29	29	0	9	.01

Cuadro 4.6: Comparativo de algoritmo estructural vs de alineación con la pareja *NUEVE*: algoritmo, número de correspondencias iniciales obtenidas con correlación (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg.).

4.2.4. Discusión

Los resultados mostraron que el algoritmo de *Transformación de Grafos* resolvió (en la mayoría de los casos) mejor el problema de correspondencias que el algoritmo de *RANSAcing Thin-plate Splines*. En dos de los casos presentados, resolvió sin ningún error mientras que el algoritmo de *RANSAcing Thin-Plate Splines* tuvo 7 y 1 errores. En el caso de la pareja CINCO, el algoritmo de alineación tuvo un sólo error mientras que el estructural tuvo 3. Sin embargo, en todos los casos el algoritmo estructural logró encontrar más correspondencias correctas que el algoritmo de alineación, logrando encontrar para la pareja *NUEVE* un 90 % de las correspondencias correctas sin nungún error. El porcentaje menor se presentó para la pareja *DOS* en donde encontro el 50 % de las correspondencias correctas. Por el otro lado,

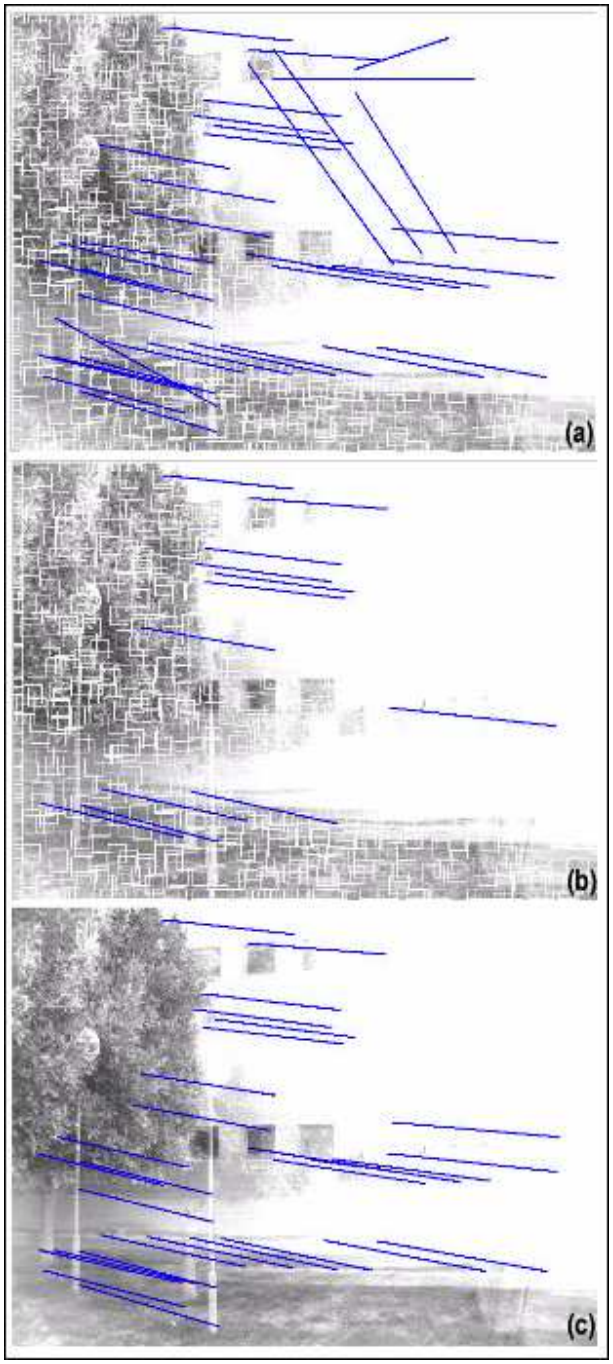


Figura 4.14: Pareja *NUEVE*, imágenes de movimiento. (a) Entrada obtenida con correlación, (b) Resultado del algoritmo de *RANSAcing Thin-plate Splines* y (c) Resultado del algoritmo de *Transformación de Grafos*.

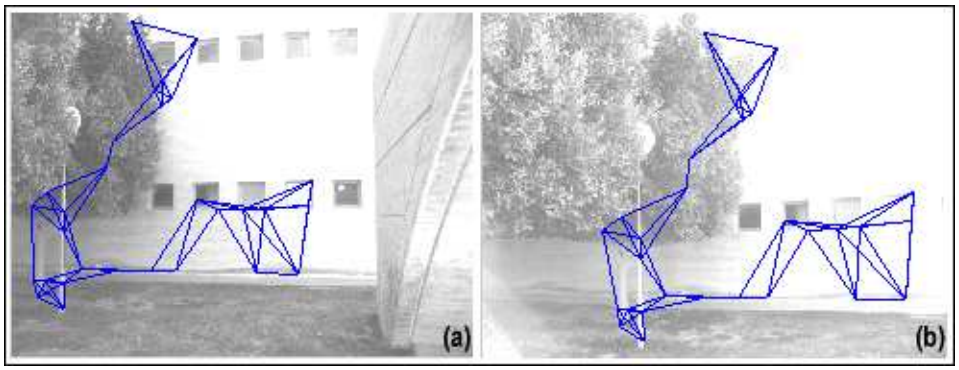


Figura 4.15: Grafos de la pareja *NUEVE*. (a) Grafo sobre la escena en el tiempo t y (b) Grafo sobre la escena en el tiempo $t + 1$

el algoritmo de alineación logro encontrar a lo más el 39 % de las correspondencias correctas e incluso no encontrar ninguna para el caso de la pareja *DOS*.

Otro aspecto importante a resaltar es el número de iteraciones. El algoritmo de alineación itera 100 veces para encontrar el resultado, mientras que el estructural a lo más le tomo 17 iteraciones. Para el primer algoritmo el número de iteraciones es fijo, contrariamente para el del segundo algoritmo el número de iteraciones depende de la cantidad de correspondencias erróneas y de la distribución espacial de las correspondencias.

Finalmente, el algoritmo de *Transformación de Grafos* resulto ser mas eficiente en tiempo que el algoritmo de *RANSAcing Thin-plate Splines*, lo cual resulta sumamente importante sobre todo para aplicaciones que requieran tiempo real.

4.3. Pruebas con oclusiones parciales

En esta prueba se utilizaron dos tipos de imágenes, uno en donde aparecía el objeto a reconocer con una cierta oclusión parcial y otro en donde la escena a reconocer se veía parcialmente ocluída por alguna persona u objeto. En total se aplico a 10 parejas de imágenes del primer tipo y a 5 del segundo. Se utilizó el extractor de características de SIFT y las correspondencias iniciales fueron la salida del algoritmo *BBF*. En todos los casos se observaron resultados similares. Tres ejemplos se presentan a continuación.

La primer pareja de imágenes consiste en reconocer el objeto *carro* dentro de una imagen que lo contiene a una diferente escala y con una oclusión parcial de aproximadamente el 50 %. La Fig. 4.16 muestra las correspondencias iniciales (a) y las correspondencias finales encontradas (b). La Fig. 4.17 muestra los grafos encontrados en el modelo (a) y en la escena (b).

La segunda pareja de imágenes está formada por el objeto *agenda* y una escena en donde se encuentra la misma *agenda* pero a una diferente escala y con una oclusión parcial que obstruye muchas de las características distintivas del objeto. La Fig. 4.18

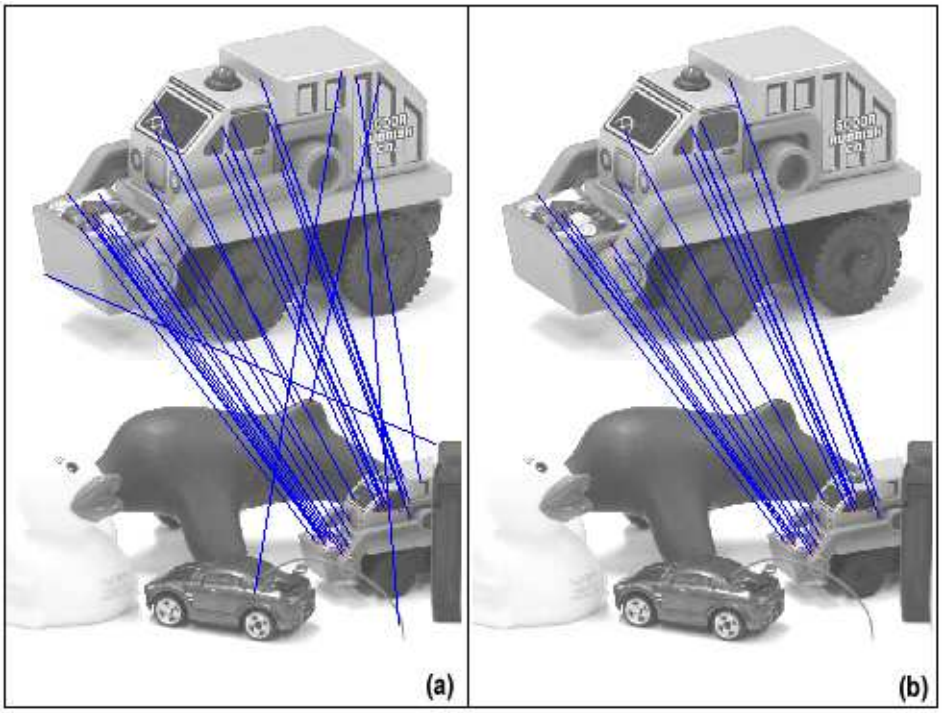


Figura 4.16: Emparejamiento inicial obtenido con el algoritmo *BBF* (a), y final obtenido con el algoritmo de *Transformación de Grafos* (b) del objeto *carro* bajo el efecto de oclusión parcial.

muestra las correspondencias iniciales (a) y las correspondencias finales encontradas (b). La Fig. 4.19 muestra los grafos encontrados en el modelo (a) y en la escena (b).

La tercera y última pareja de imágenes son escenas del *laboratorio1*. Una de ellas es la imagen de una vista representativa del *laboratorio1* mientras que la segunda es una escena desde un punto de vista ligeramente diferente y con la oclusión parcial de una persona. La Fig. 4.20 muestra las correspondencias iniciales (a) y las correspondencias finales (b). La Fig. 4.21 muestra los grafos encontrados en el modelo (a) y en la escena (b).

El Cuadro 4.7 presenta los resultados de aplicar el algoritmo de *Transformación de Grafos* a estas tres parejas de imágenes. En sus columnas muestra el nombre de la imagen, el número de correspondencias iniciales (CI), el número de correspondencias finales (CF), y de éstas el número de correspondencias erróneas (CE), el número de iteraciones (Iter) y el tiempo en segundos (Seg).

4.3.1. Discusión

Los experimentos mostraron que el algoritmo de *Transformación de Grafos* funciona correctamente cuando el objeto o escena a reconocer se encuentra parcialmente oculta. Una de las virtudes del algoritmo es que permite que el grafo sea disconexo y

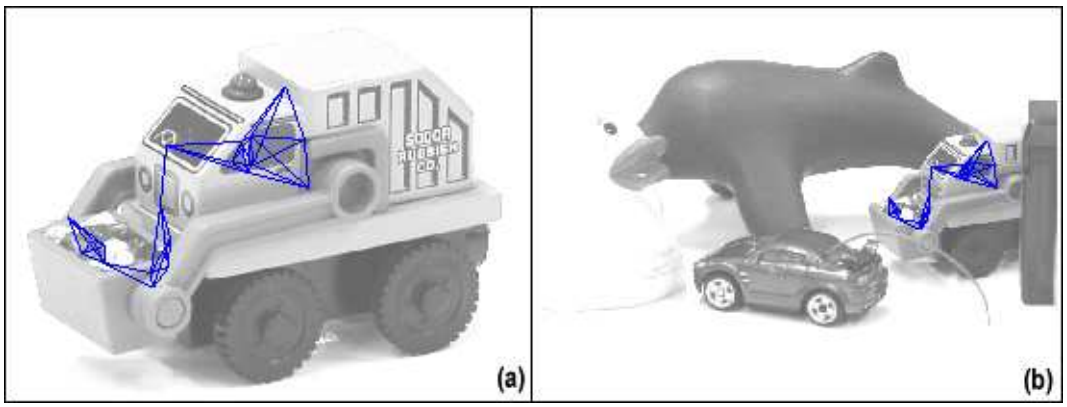


Figura 4.17: Grafos finales del objeto *carro* bajo el efecto de oclusión parcial.

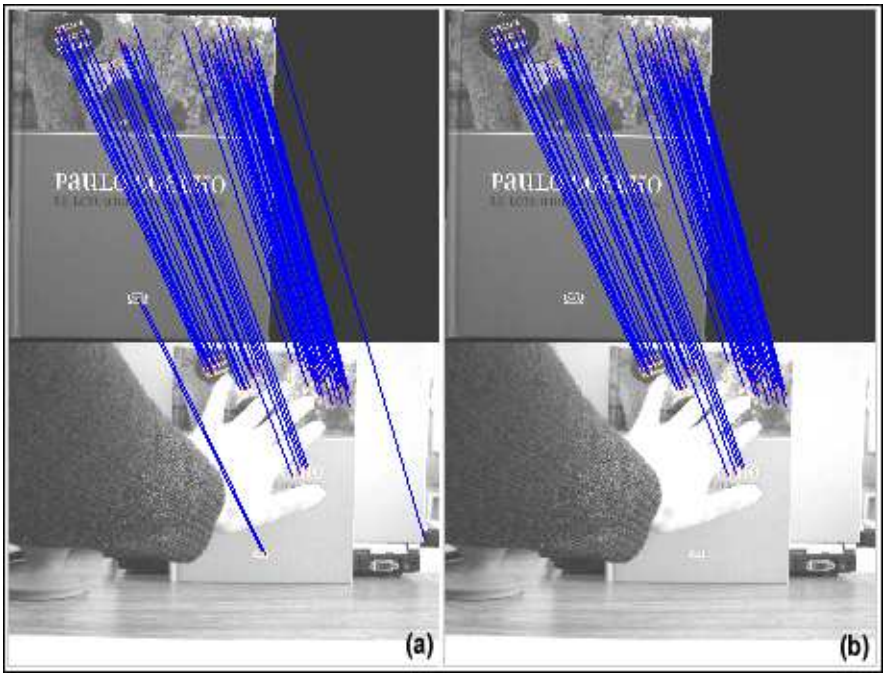


Figura 4.18: Emparejamiento inicial obtenido con el algoritmo *BBF* (a) y final obtenido con el algoritmo de *Transformación de Grafos* (b) del objeto *agenda* bajo el efecto de oclusión parcial.

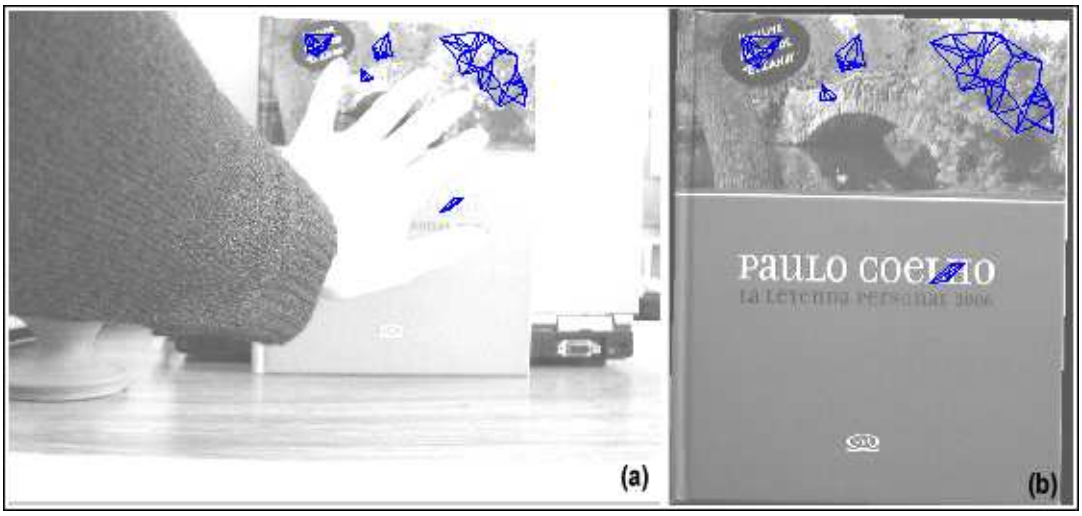


Figura 4.19: Grafos finales del objeto *agenda* bajo el efecto de oclusión parcial.

Figura 4.20: Emparejamiento inicial obtenido con el algoritmo *BBF* (a) y final obtenido con el algoritmo de *Transformación de grafos* (b) de la escena *laboratorio1* bajo el efecto de oclusión parcial.

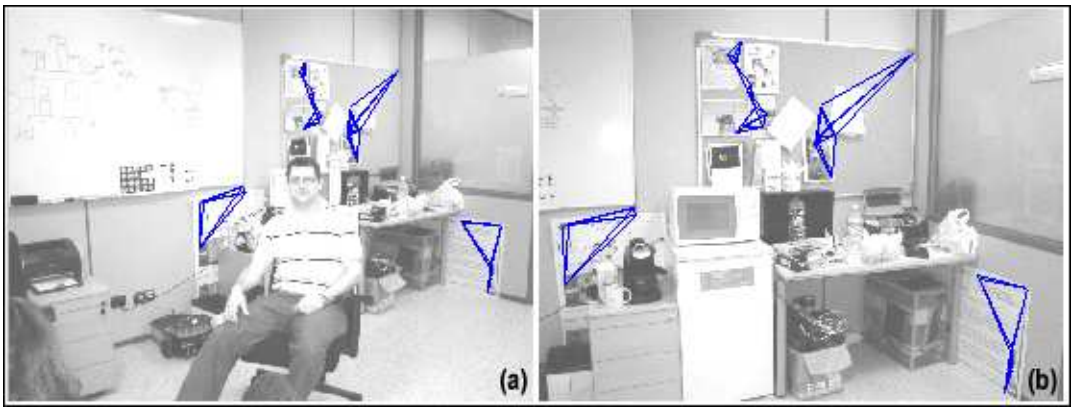


Figura 4.21: Grafos finales de la escena *laboratorio1* bajo el efecto de oclusión parcial.

Imagen	CI	CIC	CIE	CF	CFC	CFE	Iter.	Seg
Carro	30	24	6	19	19	0	11	.03
Agenda	65	63	2	59	59	0	5	.06
Laboratorio1	49	43	6	35	35	0	15	.06

Cuadro 4.7: Resultados de aplicar el algoritmo de *Transformación de Grafos* a imágenes con oclusiones parciales: imagen, número de correspondencias iniciales obtenidas con *BBF* (CI), número de correspondencias finales (CF), número de correspondencias erróneas (CE), número de iteraciones (Iter) y tiempo en segundos (Seg.).

con eso el que las características se encuentren acumuladas en grupos espacialmente lejanos. Esta virtud ayuda en los casos en que la oclusión divida espacialmente a las características, como es el caso de objeto *agenda* y de la escena *laboratorio1*. Es importante resaltar que el reconocimiento de los objetos fue exitoso debido en gran parte a que el detector de características SIFT fue capaz de detectar las mismas características en ambas imágenes apesar de que había cambios de escala, de puntos de vista e incluso un poco de variación en la iluminación. La otra parte del éxito se debió a que el algoritmo eliminó correctamente las correspondencias erróneas.

4.4. Algoritmo de fuerza bruta vs algoritmo optimizado

Se realizaron tres implementaciones del algoritmo de *Transformación de Grafos*. Las dos primeras hacen una implementación del algoritmo de fuerza bruta (AFB) presentado en la sección 3.1.1, una escrita en *Matlab* y otra escrita en *C*. La tercera es una implementación en *C* del algoritmo optimizado presentado en la sección 3.1.2. La complejidad del algoritmo de fuerza bruta (AFB) es de $O(n^3 Log(n))$, en donde cada iteración es de $O(n^2 Log(n))$ causada principalmente por la reconstrucción de

los grafos. Por otro lado, la versión optimizada del algoritmo de fuerza bruta realiza una optimización en la reconstrucción de los grafos.

Los cuadros 4.8 y 4.9 presentan un comparativo de los tiempos reportados por las tres implementaciones para todas las imágenes presentadas en éste y en el siguiente capítulo.

Imagen	**CI**	**CF**	**Iter.**	**Matlab (AFB)**	**C (AFB)**	**C Optim.**
tasa	74	28	47	0.01	0.09	0.01
poster	217	170	48	1.15	0.46	0.2
recámara	19	9	9	0.01	0.0	0.0
DOS (grad. filtro calidad correlación)	24	6	16	0.12	0.0	0.0
CINCO(grad. filtro calidad correlación)	42	26	17	0.01	0.01	0.01
NUEVE (grad. filtro calidad correlación)	37	29	9	0.01	0.0	0.0
carro	30	19	11	0.03	0.01	0.01
agenda	65	59	5	0.06	0.03	0.02
laboratorio1	49	35	15	0.06	0.04	0.02
UNO (grad. sin filtros)	478	46	433	23.09	9.87	1.06
UNO (grad. filtro 1-1)	120	42	79	0.42	0.29	0.04
UNO (grad. filtro calidad correlación)	62	38	23	0.09	0.06	0.01
UNO (SIFT sin filtros)	225	192	34	1.03	0.51	0.18
UNO (SIFT. filtro 1-1)	194	151	44	0.89	0.4	0.12

Cuadro 4.8: Comparativo 1 de tiempos del algoritmo de *Transformación de Grafos* implementado en *Matlab*, en *C* y su versión optimizada en *C*: imagen, número de correspondencias iniciales obtenidas con *BBF* (CI), número de correspondencias finales (CF), número de iteraciones(Iter.), tiempo en segundos de la implementación del algoritmo de fuerza bruta en *Matlab*, tiempo en segundos de la implementación del algoritmo de fuerza bruta en *C* y tiempo en segundos de la implementación del algoritmo optimizado en *C*.

También se realizo un estudio del comportamiento del desempeño de las imple-

Imagen	CI	CF	Iter.	Matlab (AFB)	C (AFB)	C Optim.
SEIS (grad. sin filtros)	559	42	518	36.6	21.14	1.68
SEIS (grad. filtro 1-1)	87	19	68	0.18	0.14	0.03
SEIS (grad. filtro calidad correlación)	24	20	3	0.01	0.0	0.0
SEIS (SIFT sin filtros)	96	86	11	0.09	0.04	0.03
SEIS (SIFT. filtro 1-1)	77	71	7	0.04	0.04	0.01
OCHO (grad. sin filtros)	574	78	497	39.07	55.6	1.76
OCHO (grad. filtro 1-1)	143	76	68	0.58	0.25	0.07
OCHO (grad. filtro calidad correlación)	107	75	33	0.25	0.09	0.03
OCHO (SIFT sin filtros)	1545	1334	212	278.1	89.79	9.4
OCHO (SIFT. filtro 1-1)	1361	1165	197	195.6	64.92	7.29
NUEVE (grad. sin filtros)	520	25	494	29.1	15.2	1.6
NUEVE (grad. filtro 1-1)	93	20	74	0.29	0.17	0.03

Cuadro 4.9: Comparativo 2 de tiempos del algoritmo de *Transformación de Grafos* implementado en *Matlab*, en *C* y su versión optimizada en *C*: imagen, número de correspondencias iniciales obtenidas con *BBF* (CI), número de correspondencias finales (CF), número de iteraciones(Iter.), tiempo en segundos de la implementación del algoritmo de fuerza bruta en *Matlab*, tiempo en segundos de la implementación del algoritmo de fuerza bruta en *C* y tiempo en segundos de la implementación del algoritmo optimizado en *C*.

mentaciones, notando primeramente que el tiempo requerido por el algoritmo de *Transformación de Grafos* depende de 2 variables: del número de correspondencias iniciales y del número de iteraciones requeridas para converger. Esta última variable esta íntimamente relacionada con el número de correspondencias iniciales erróneas.

El primer experimento consistió en fijar el número de iteraciones a 40 e ir variando el número de correspondencias iniciales desde 50 hasta 1000 en incrementos de 50

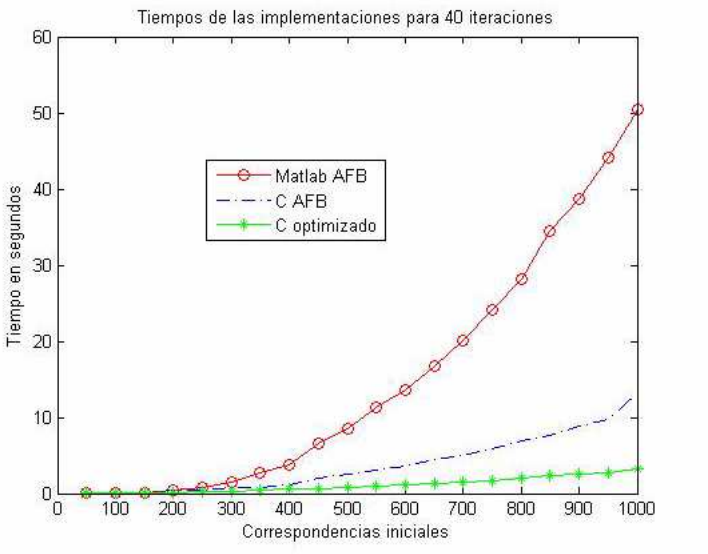


Figura 4.22: Gráficas de los tiempos en segundos reportados por las tres implementaciones del algoritmo de *Transformación de Grafos* para 40 iteraciones y un número variable de correspondencias iniciales que va de 50 a 1000 en incrementos de 50 correspondencias.

correspondencias. Se eligió el valor de 40 debido a que si se elegía un valor muy pequeño, entonces el desempeño de los tres algoritmos se vería muy similar debido a que la mayor cantidad del tiempo requerido sería utilizado en la fase de inicialización, la cual es $O(n^2 Log(n))$ para las tres implementaciones y si el valor era muy grande entonces la diferencia en el desempeño se vería muy afectada por el esfuerzo extra en la reconstrucción de los grafos en cada iteración del algoritmo de fuerza bruta. Para el caso de los valores elegidos para las correspondencias iniciales se tomo un rango que abarcara los dos extremos del número de correspondencias encontradas en imágenes típicas a ser usadas. La Fig. 4.22 y el Cuadro 4.10 muestran el desempeño de las tres implementaciones.

El segundo experimento consistió en variar ahora el segundo parámetro del que depende el tiempo requerido por el algoritmo: el número de iteraciones. Se fijo el número de correspondencias iniciales a 559 y se hizo variar el número de iteraciones de 25 a 500 en incrementos de 25 iteraciones. La Fig. 4.23 y el Cuadro 4.11 presentan los tiempos obtenidos con las tres implementaciones.

El tercer y último experimento consistió en probar el algoritmo con un número de correspondencias iniciales fijo, que fuera lo suficientemente grande para ser considerado como un número razonable de características extraídas en una imagen pero al mismo tiempo lo suficientemente pequeño para reportar tiempos adecuados. Se eligió fijar el número de correspondencias iniciales a 200 y variar el número de iteraciones de 10 a 190 en incrementos de 10 iteraciones. La Fig. 4.24 y el Cuadro 4.12 presentan los tiempos obtenidos de las tres implementaciones.

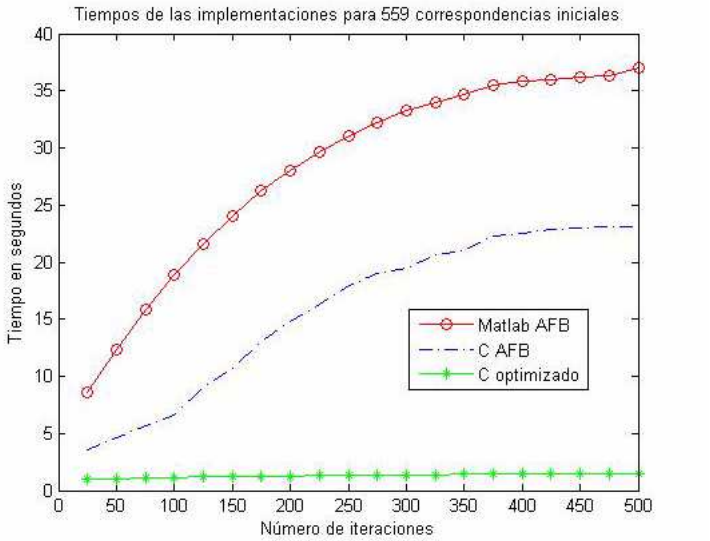


Figura 4.23: Gráficas de los tiempos en segundos reportados por las tres implementaciones del algoritmo de *Transformación de Grafos* para 559 correspondencias iniciales y un número variable de correspondencias iniciales que va de 25 a 500 en incrementos de 25 iteraciones.

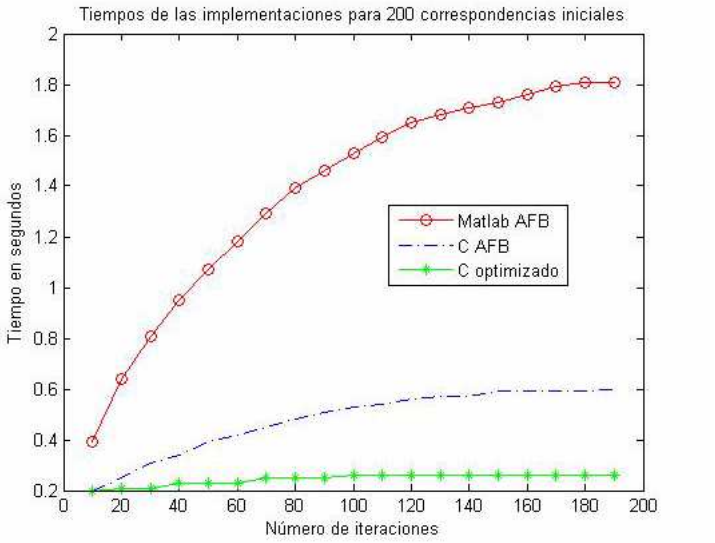


Figura 4.24: Gráficas de los tiempos en segundos reportados por las tres implementaciones del algoritmo de *Transformación de Grafos* para 200 correspondencias iniciales y un número variable de iteraciones que va de 10 a 190 en incrementos de 10 iteraciones.

CI	Matlab (AFB)	C (AFB)	C optimizado
50	0.01	0.016	0.016
100	0.06	0.047	0.047
150	0.20	0.12	0.07
200	0.45	0.37	0.14
250	0.78	0.51	0.29
300	1.48	0.71	0.34
350	2.68	0.81	0.39
400	3.76	1.06	0.59
450	6.64	2.0	0.68
500	8.54	2.53	0.84
550	11.32	3.04	0.92
600	13.53	3.62	1.12
650	16.84	4.43	1.29
700	20.12	5.04	1.48
750	24.06	5.89	1.71
800	28.18	6.87	2.01
850	34.48	7.67	2.29
900	38.7	8.78	2.5
950	44.09	9.68	2.8
1000	50.5	13.25	3.17

Cuadro 4.10: Comparativo de tiempos del algoritmo de *Transformación de Grafos* implementado en *Matlab*, en *C* y su versión optimizada en *C* para 40 iteraciones: número de correspondencias iniciales (CI), tiempo en segundos de la implementación del algoritmo de fuerza bruta en *Matlab*, tiempo en segundos de la implementación del algoritmo de fuerza bruta en *C* y tiempo en segundos de la implementación del algoritmo optimizado en *C*.

4.4.1. Discusión

El primer experiemento mostró que la diferencia en tiempo entre las implementaciones en *Matlab* y en *C* del algoritmo de fuerza bruta, se vuelve evidente cuando el número de correspondencias iniciales es mayor a 200 para un número pequeño de iteraciones como lo es 40 (aproximádamente un 20 % de correspondencias iniciales erróneas). La diferencia en tiempo llego a ser de hasta 37.25 seg en el caso de 1000 correspondencias iniciales.

Para analizar la diferencia entre el algoritmo de fuerza bruta y el optimizado para un número fijo de iteraciones, observemos las gráficas correspondientes a ambas implementaciones en *C* de la Fig. 4.22. De éstas, podemos concluir que el algoritmo optimizado es significativamente más eficiente que el de fuerza bruta para un número de correspondencias iniciales mayor a 200, llegando a existir una diferencia de hasta

Iteraciones	Matlab (AFB)	C (AFB)	C optimizado
25	8.64	3.57	0.98
50	12.3	4.67	1.03
75	15.79	5.68	1.07
100	18.87	6.62	1.12
125	21.62	9.03	1.17
150	24.06	10.76	1.20
175	26.20	13.01	1.25
200	28.07	14.85	1.28
225	29.68	16.31	1.31
250	31.06	17.90	1.34
275	32.23	19.03	1.35
300	33.23	19.51	1.39
325	34.00	20.68	1.40
350	34.64	20.96	1.42
375	35.51	22.26	1.43
400	35.79	22.53	1.45
425	36.00	22.89	1.45
450	36.23	23.03	1.46
475	36.28	23.07	1.46
500	37.03	23.10	1.48

Cuadro 4.11: Comparativo de tiempos del algoritmo de *Transformación de Grafos* implementado en *Matlab*, en *C* y su versión optimizada en *C* para 559 correspondencias iniciales: Número de correspondencias iniciales obtenidas con *BBF* (CI), tiempo en segundos de la implementación del algoritmo de fuerza bruta en *Matlab*, tiempo en segundos de la implementación del algoritmo de fuerza bruta en *C* y tiempo en segundos de la implementación del algoritmo optimizado en *C*.

10.08 seg para el caso de 1000 correspondencias iniciales. Recordemos que la fase de inicialización es la misma para ambos algoritmos y que la optimización se realizó principalmente al optimizar la reconstrución del grafo en cada iteración. Este tiempo de reconstrucción del grafo fué el que marcó la diferencia en el desempeño de ambos algoritmos implementados en *C*, ya que apesar de que 40 iteraciones es un número relativamente pequeño, resultó ser lo suficientemente grande como para que los tiempos difirieran en hasta 10.08 segundos.

El segundo experimento mostró que la implementación en *C* del algoritmo de fuerza bruta sigue siendo más eficiente que su implementación en *Matlab* hasta por 13.93 segundos para 500 iteraciones. Para el caso de las implementaciones en *C* del algoritmo de fuerza bruta y del optimizado, la diferencia en tiempos llego a ser de hasta 21.62 segundos para 500 iteraciones, resultando ser mucho más eficiente la versión optmizada. Algo interesante a notar es el comportamiento casi constante del

Iteraciones	Matlab (AFB)	C (AFB)	C optimizado
10	0.39	0.20	0.12
20	0.64	0.25	0.12
30	0.81	0.32	0.14
40	0.95	0.37	0.14
50	1.07	0.39	0.14
60	1.18	0.42	0.15
70	1.29	0.45	0.15
80	1.39	0.48	0.15
90	1.46	0.51	0.17
100	1.53	0.53	0.17
110	1.59	0.54	0.17
120	1.65	0.56	0.17
130	1.68	0.57	0.17
140	1.71	0.57	0.18
150	1.73	0.59	0.18
160	1.76	0.59	0.18
170	1.79	0.59	0.18
180	1.81	0.59	0.18
190	1.81	0.60	0.18

Cuadro 4.12: Comparativo de tiempos del algoritmo de *Transformación de Grafos* implementado en *Matlab*, en *C* y su versión optimizada en *C* para 200 correspondencias iniciales: número de correspondencias iniciales obtenidas con *BBF* (CI) , tiempo en segundos de la implementación del algoritmo de fuerza bruta en *Matlab*, tiempo en segundos de la implementación del algoritmo de fuerza bruta en *C* y tiempo en segundos de la implementación del algoritmo optimizado en *C*.

algoritmo optimizado, indicando que este algoritmo requerirá casi el mismo tiempo para procesar entradas con cualquier cantidad de correspondencias iniciales erróneas. En otras palabras, es prácticamente insensible a la cantidad iteraciones requeridas para converger. Cuantitativamente, la diferencia en segundos de realizar 25 iteraciones a realizar 500 iteraciones es de 0.5 segundos para el algoritmo optimizado contra 19.53 segundos del algoritmo de fuerza bruta.

Otro punto interesante a notar es que el tiempo requerido en cada iteración (del algoritmo de fuerza bruta) va disminuyendo conforme más iteraciones se hayan realizado, ésto debido a que recordemos que cada iteración implica la eliminación de un vértice y con ésto el que la reconstrucción sea menos costosa.

Finalmente, el tercer experimento mostró que para un número razonable de características iniciales (200), la implementación en *C* del algoritmo de fuerza bruta resultó nuevamente ser mucho más eficiente (aproximadamente la tercera parte del tiempo) que su implementación en *Matlab*. Más interesante, resulta notar que la

versión optimizada difiere en únicamente 0.06 segundos en realizar 10 iteraciones o realizar 190 iteraciones. En otras palabras, requiere prácticamente el mismo tiempo en resolver el problema para entradas con el 5 % de correspondencias iniciales erróneas que para el 95 %. Desde únicamente 10 iteraciones la diferencia en tiempo entre el algoritmo de fuerza bruta y el optimizado llega a ser significativa si ha de considerarse aspirar a realizar el reconocimiento de objetos en tiempos razonables, lo cual dependerá de la aplicación.

Estos experimentos sugieren que un número de 200 correspondencias iniciales es un límite adecuado para considerar realizar el reconocimiento de objetos en tiempos razonables (0.18 seg para la etapa de verificación). Además, las imágenes reportadas en este capítulo sugieren también que 200 es un número suficente de características detectadas en las imágenes para su reconocimiento.

4.5. Casos en los que no funciona

Se detectaron 2 casos en los que el algoritmo no funciona correctamente. El primer caso tiene que ver con el valor de la conectividad k. Se observó experimentalmente que $k = 4$ era el valor que resolvía mejor el emparejamiento en todos los casos. Esto implica que cuando se tengan (desde un inicio o en alguna iteración) únicamente 5 correspondencias, entonces el algoritmo siempre construirá dos grafos idénticos, sin importar si las correspondencias son correctas o erróneas. Las matrices de adyacencia de ambos grafos serán:

$$\begin{array}{c} \\ v1 \\ v2 \\ v3 \\ v4 \\ v5 \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & v_3 & v_4 & v_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Un ejemplo de este caso se muestra en la Fig. 4.25. Una sugerencia para poder resolver este problema de forma sencilla consiste en que como paso final del algoritmo, se eliminen aquellas correspondencias que se encuentren en subgrafos desconectados y que estén formados por únicamente 5 correspondencias.

El segundo caso se presenta cuando existe más de una instancia de alguna parte del objeto. Un ejemplo de esto se muestra en la Fig. 4.26 en donde se desea reconocer un poster que aparece en la escena junto a otro que comparte algunos elementos como los letros e íconos. Debido a que en las correspondencias iniciales hay un agrupamiento de ellas que cumplen con que su distancia es menor o igual a la mediana y que son mas de 6, entonces aparea erróneamente el ícono del poster a reconocer con el del otro poster. Si se modificara el algoritmo para que hiciera una verificación global de las correspondencias se podría esperar que este tipo de casos particulares se resolvieran.

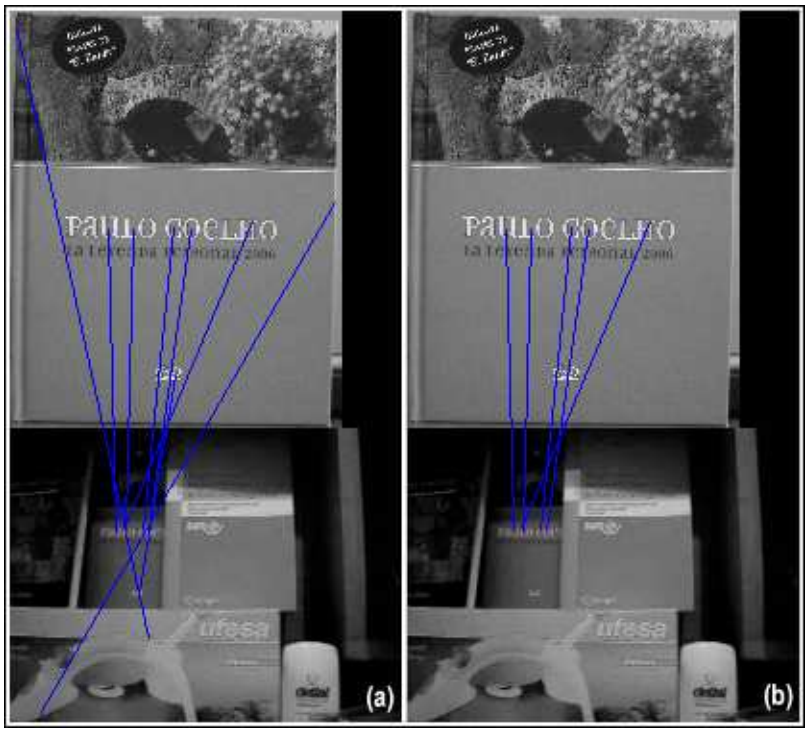


Figura 4.25: Ejemplo 1 de un caso en el que el algoritmo no funciona.

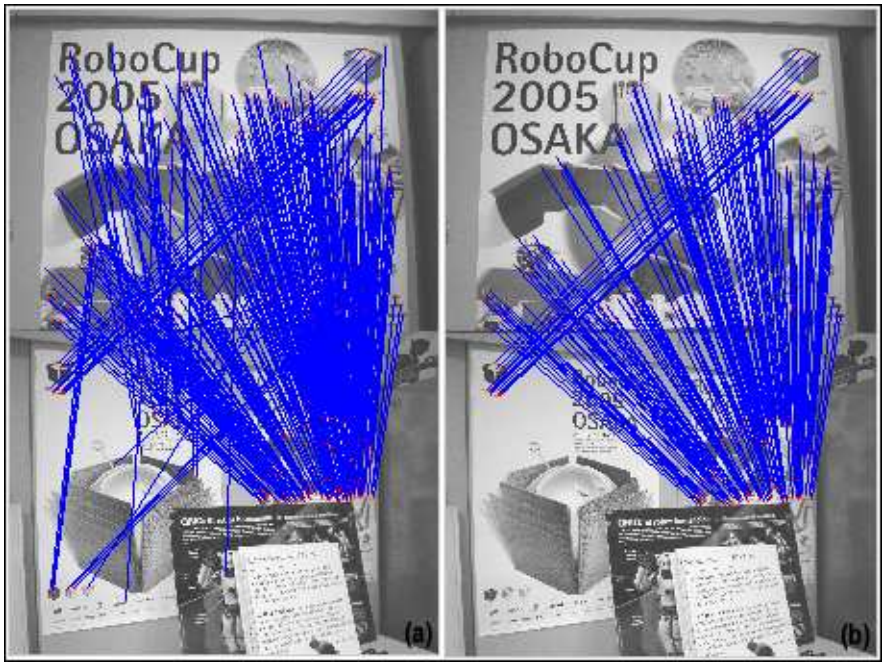


Figura 4.26: Ejemplo 2 de un caso en el que el algoritmo no funciona.

4.6. Otros resultados

El algoritmo también se probo con imágenes de fondo de ojo que se desean hacer corresponder para formar mosaicos y lograr reconstrucciones en 2 y 3 dimensiones. Para éstas imágenes se utilizó el detector de características descrito en [54] y se utilizó una correlación para encontrar el emparejamiento inicial. Los resultados de dos de los experimentos de muestran en la Fig. 4.27 y en la Fig. 4.28. Las Figuras 4.29 y 4.30 muestran los grafos encontrados en otras parejas de imágenes a ser usadas en reconocimiento de escenas y en reconocimiento de objetos. En todos los casos el número de correspondencias finales erróneas fué de 0.

4.7. Síntesis

En este capítulo se presentaron los resultados obtenidos de las pruebas realizadas al algoritmo de *Transformación de Grafos*. Las dos primeras pruebas consistieron en compararlo contra dos algoritmos, uno de enfoque estructural llamado *Softassign* y el segundo de enfoque de alineación llamado *RANSACing Thin-Plate Splines*. En ambos casos el algoritmo de *Transformación de Grafos* resulto ser superior en tiempo, en el número de correspondencias correctas encontradas y en el número de errores cometidos.

La siguiente prueba consistió en probarlo con objetos que sufrían de oclusiones parciales, con las cuales funcionó correctamente también.

Enseguida se reportaron pruebas del desempeño de las implementaciones del algoritmo de fuerza bruta en *C* y en *Matlab* y del algoritmo optimizado implementado en *C*. Estas pruebas mostraron que la implementación del algoritmo de fuerza bruta en *C* es mucho más eficiente que en *Matlab* hasta por 37.25 segundos para el caso de 1000 correspondencias iniciales y 40 iteraciones. Mejor aún, el algoritmo optimizado implementado en *C* generó gráficas muy parecidas a las de una constante indicando que éste es muy poco sensible en tiempo al número de iteraciones requeridas para converger y con ésto al número de correspondencias iniciales erróneas.

Como conclusión final, se llego a que para considerar aspirar a realizar el reconociento de objetos en tiempos razonables se tome como límite 200 características iniciales.

Finalmente, se reportaron dos casos en los que el algoritmo no funciona correctamente y otros resultados interesantes.

Una característica deseable en los algoritmos de correspondencia es que sea robuzto a una buena cantidad de correspondencias erróneas. En el siguiente capítulo se presentan los experimentos y resultados de la robustez a correspondencias erróneas del algoritmo de *Transformación de Grafos*.

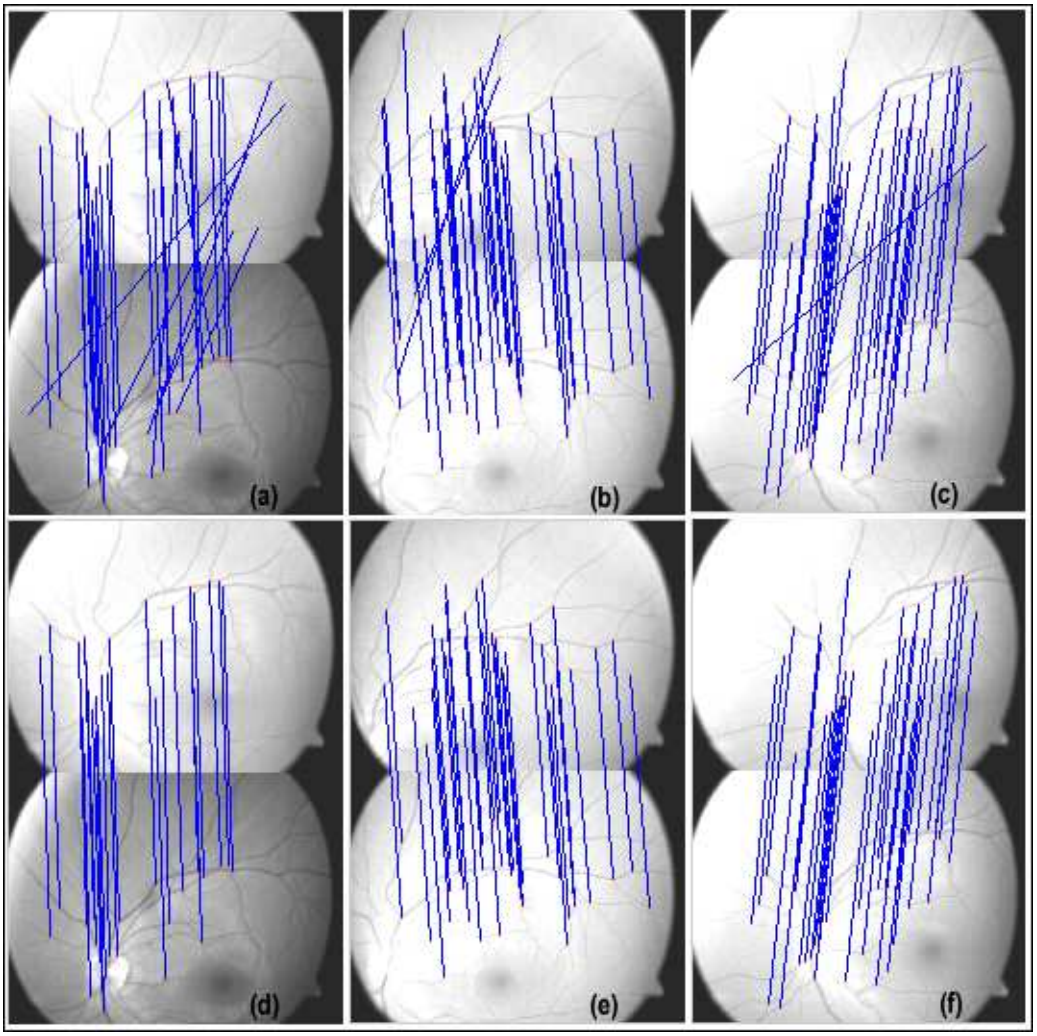


Figura 4.27: Correspondencias iniciales y finales de imágenes de fondo de ojo. (a)-(c) entradas obtenidas con correlación y (d)-(f) salidas obtenidas con el algoritmo de *Transformación de Grafos*.

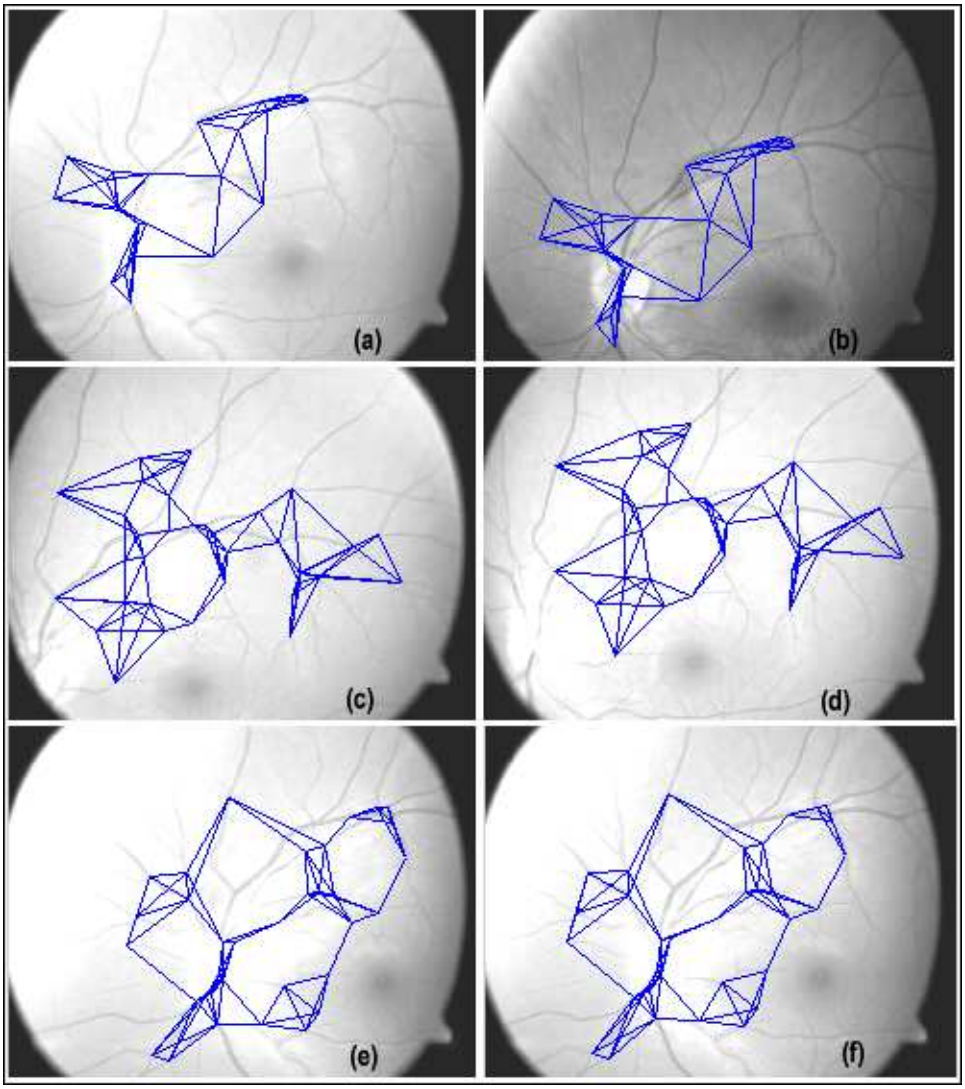


Figura 4.28: Grafos resultantes de aplicar el algoritmo de *Transformación de Grafos* a imágenes de fondo de ojo.

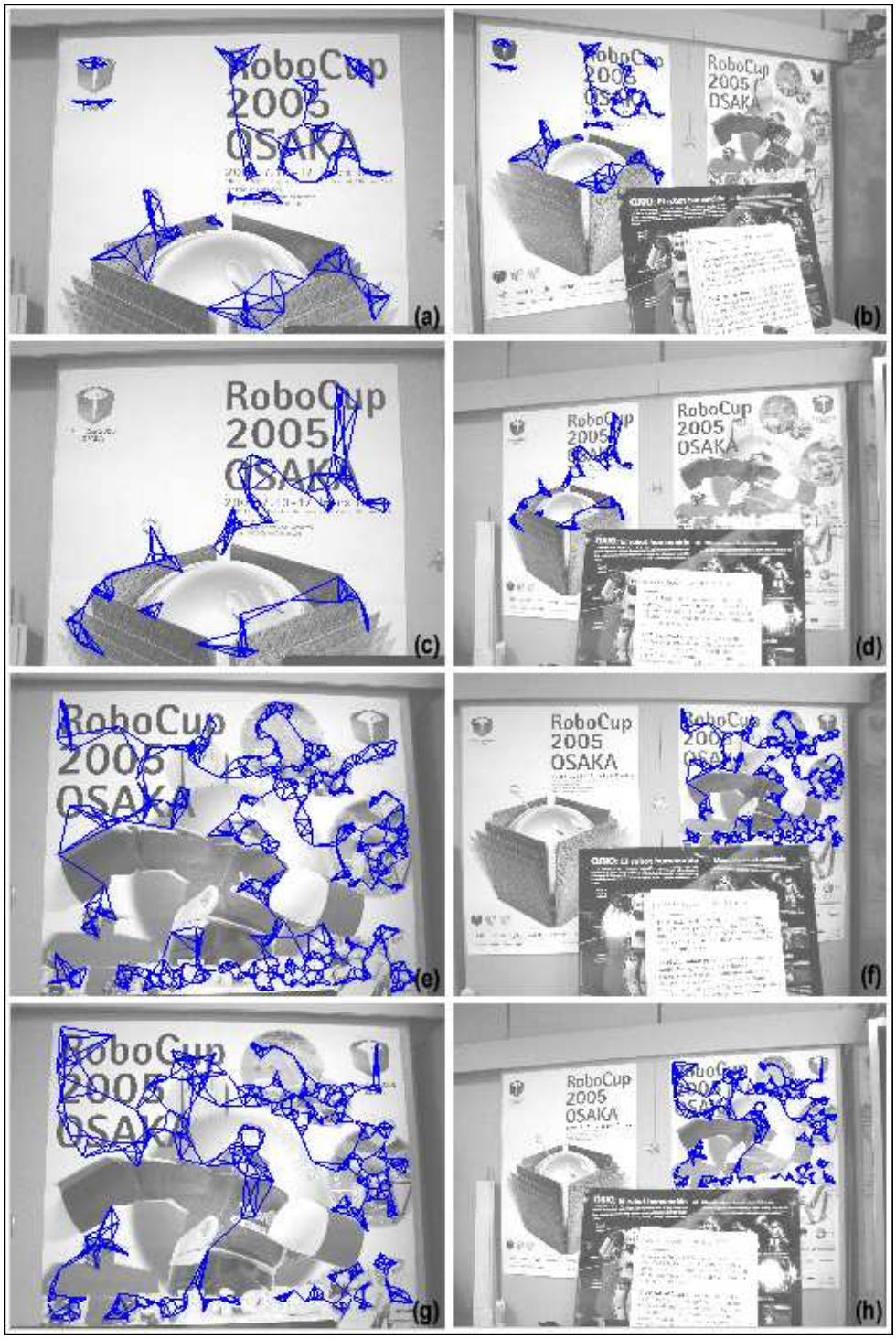


Figura 4.29: Grafos resultantes de aplicar el algoritmo de *Transformación de Grafos* a la escena de posters.

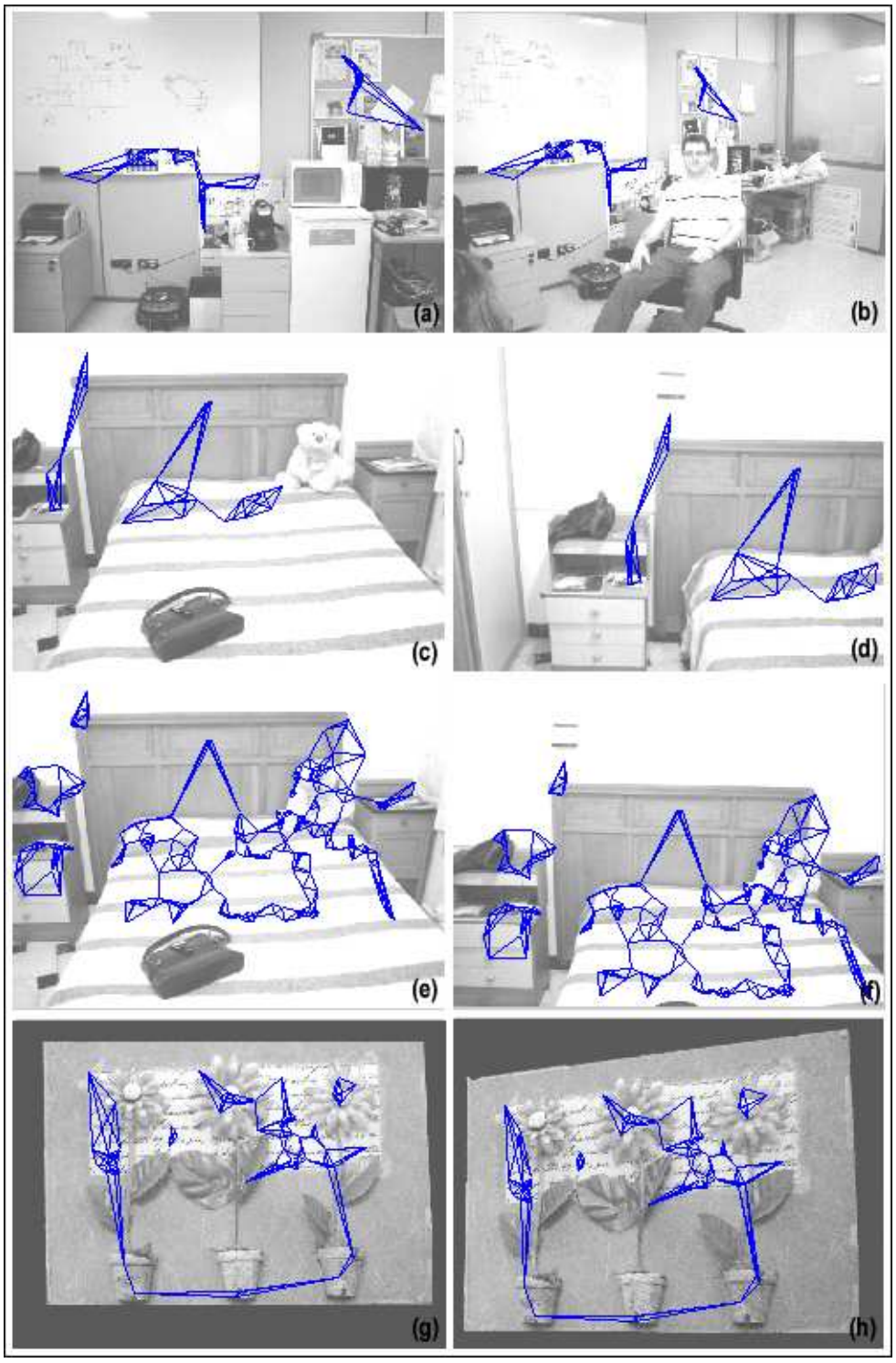


Figura 4.30: Grafos resultantes de aplicar el algoritmo de *Transformación de Grafos* a otras escenas y objetos.

Capítulo 5

Resultados de robustez a correspondencias erróneas

Los experimentos anteriores mostraron una gran eficacia, sin embargo, si observamos la entrada podemos darnos cuenta de que las correspondencias iniciales son en su gran mayoría correctas. En otras palabras, contienen muy pocas correspondencias erróneas. La pregunta ahora es, ¿cómo se comportará cuando la entrada sea ruidosa? Para contestar esta pregunta se diseñaron dos experimentos. El primero consiste en probar su robustez a correspondencias erróneas aleatorias. El segundo consiste en ponerlo a prueba pero con entradas que por su naturaleza contienen grandes porcentajes de correspondencias iniciales erróneas. En las siguientes secciones se decriben con detalle ambos experimentos.

5.1. Pruebas de robustez a correspondencias erróneas aleatorias

Este experimento consistió en tomar una pareja de imágenes con correspondencias iniciales correctas (obtenidas con el algoritmo de *SIFT* y emparejadas con el algoritmo de *BBF*) y agregarles de manera controlada diferentes porcentajes de correspondencias erróneas de manera aleatoria con la idea de concluir de qué manera y hasta que punto el algoritmo se veía afectado por este tipo de ruido.

Se probó con 10 parejas de imágenes diferentes, las cuales mostraron resultados similares. Se seleccionó una pareja de imágenes representativa y se reportan sus resultados a continuación.

El cuadro 5.1 muestra los resultados obtenidos con el objeto *agenda*. En sus columnas presenta el porcentaje de correspondencias erróneas (del 15 % al 95 % del total), el número de correspondencias iniciales (CI), el número de correspondencias iniciales correctas (CIC), el número de correspondencias iniciales erróneas (CIE), el número de correspondencias finales (CF), el número de correspondencias finales correctas (CFC), el número de correspondencias finales erróneas(CFE), el número

de iteraciones y el tiempo en segundos. Es importante señalar que se utilizó la implementación en *Matlab* y que el tiempo reportando corresponde a esa versión.

Ruido	CI	CIC	CIE	CF	CFC	CFE	Iter.	Seg
15 %	71	60	11	59	59	0	12	.04
35 %	92	60	32	55	55	0	35	.12
55 %	133	60	73	54	55	0	80	.4
75 %	240	60	180	50	50	0	191	2.5
85 %	400	60	340	50	48	2	349	11.3
95 %	1196	60	1136	26	24	2	1168	313.17

Cuadro 5.1: Resultados de robustez a correspondencias erróneas aleatorias del objeto *Agenda*: Porcentaje de correspondencias erróneas, número de correspondencias iniciales (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg).

La Fig. 5.1 muestra en la parte superior las entradas al algoritmo con un porcentaje de correspondencias erróneas: (a) 15 %, (b) 35 % y (c) 55 %. En su parte inferior (d)-(f), muestra las salidas del algoritmo respectivamente. De manera similar, la Fig. 5.2 muestra las entradas y salidas pero ahora con porcentajes de correspondencias erróneas: (a) 75 %, (b) 85 % y (c) 95 %. Sus correspondientes grafos encontrados se muestran las Figuras 5.3 y 5.4.

5.1.1. Discusión

Los resultados mostraron que el algoritmo es robusto al ruido de correspondencias erróneas aleatorias, obteniendo buenos resultados incluso con un ruido del 95 %. En todos los casos el número de correspondencias finales erróneas no fue mayor a 5 (2 para el caso del objeto *agenda*), sin importar el porcentaje de ruido que se le agregaba. Un comportamiento generalizado fue que a mayor porcentaje de ruido el número de correspondencias finales disminuía. Para porcentajes de ruido menores al 85 % el algoritmo fué capaz de identificar por lo menos el 83 % de las correspondencias correctas iniciales sin ningún error. El peor de los casos se presentó con un porcentaje de ruido del 95 %, en el cual únicamente encontró 24 correspondencias finales correctas de 60. También se puede ver claramente en el Cuadro 5.1 que el número de iteraciones y el tiempo en segundos están íntimamente relacionados. Esto se debe a que en cada iteración se reconstruye el grafo, que toma tiempo de orden cuadrático.

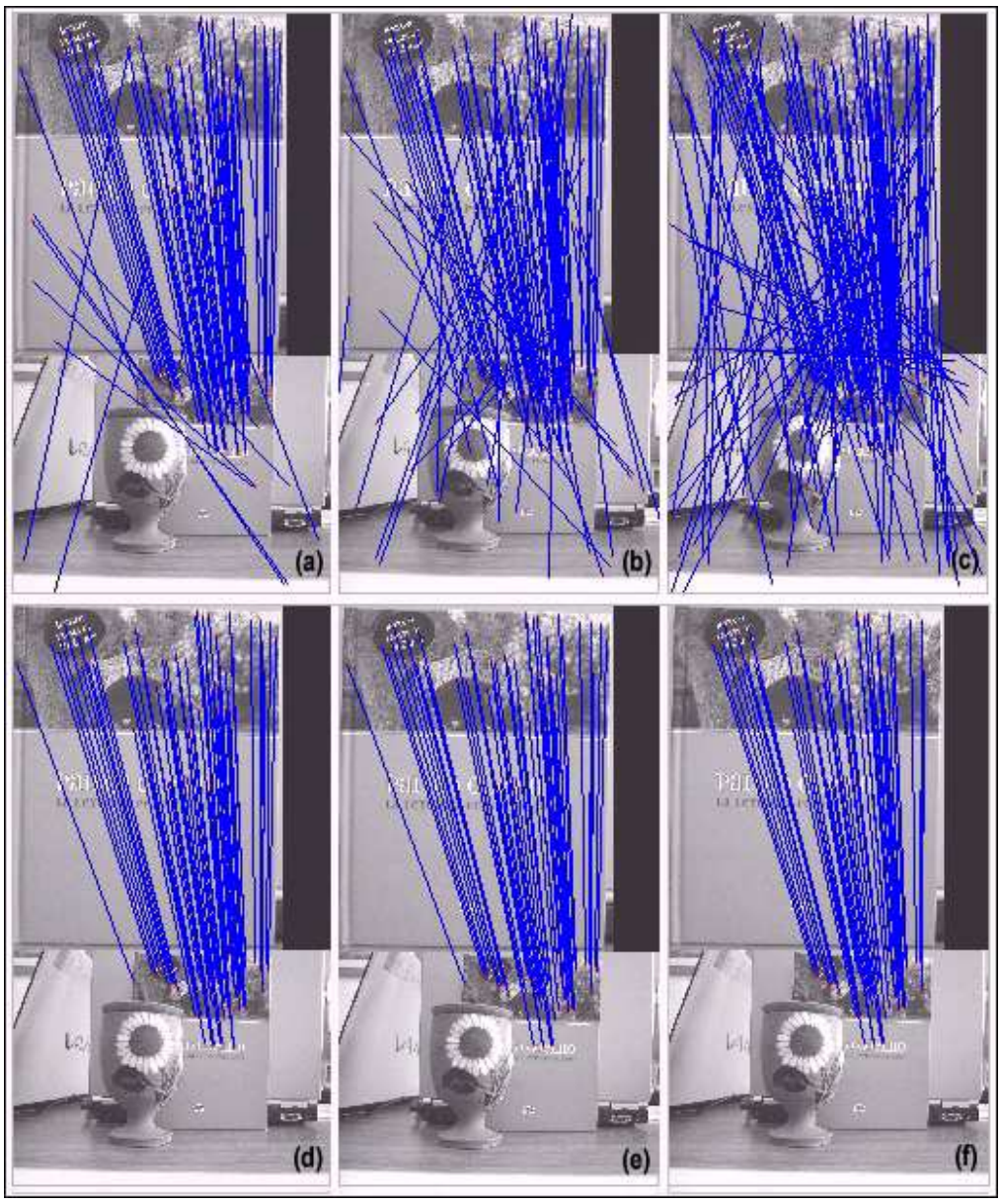


Figura 5.1: Resultados de robustez a correspondencias erróneas aleatorias del objeto *Agenda*. La parte superior muestra las entradas al algoritmo con un porcentaje de ruido: (a) 15 %, (b) 35 % y (c) 55 %. La parte inferior (d)-(f), muestra las salidas del algoritmo, respectivamente.

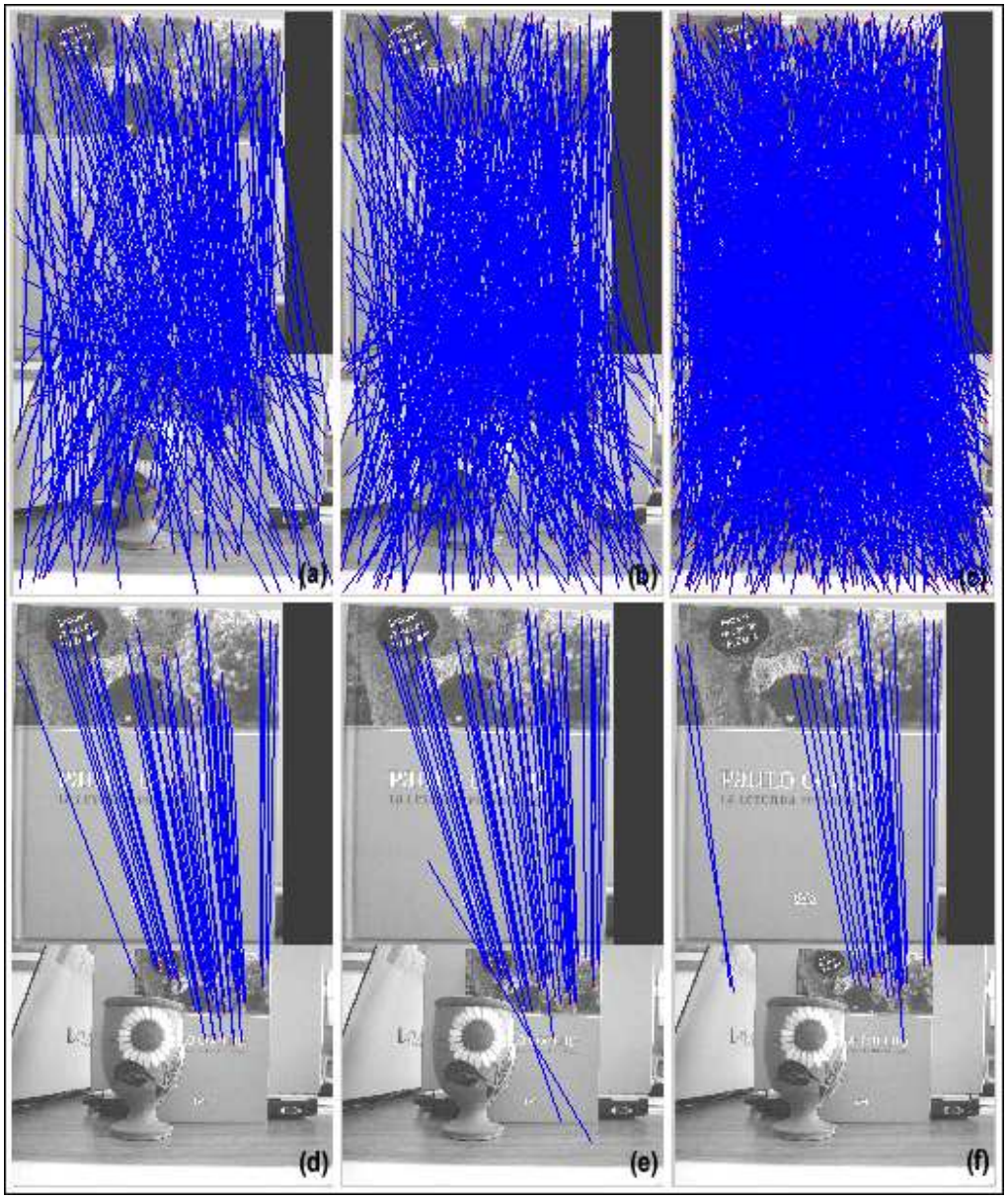


Figura 5.2: Resultados de robustez a correspondencias erróneas aleatorias del objeto *Agenda*. La parte superior muestra las entradas al algoritmo con un porcentaje de ruido: (a) 75 %, (b) 85 % y (c) 95 %. La parte inferior (d)-(f), muestra las salidas del algoritmo, respectivamente.

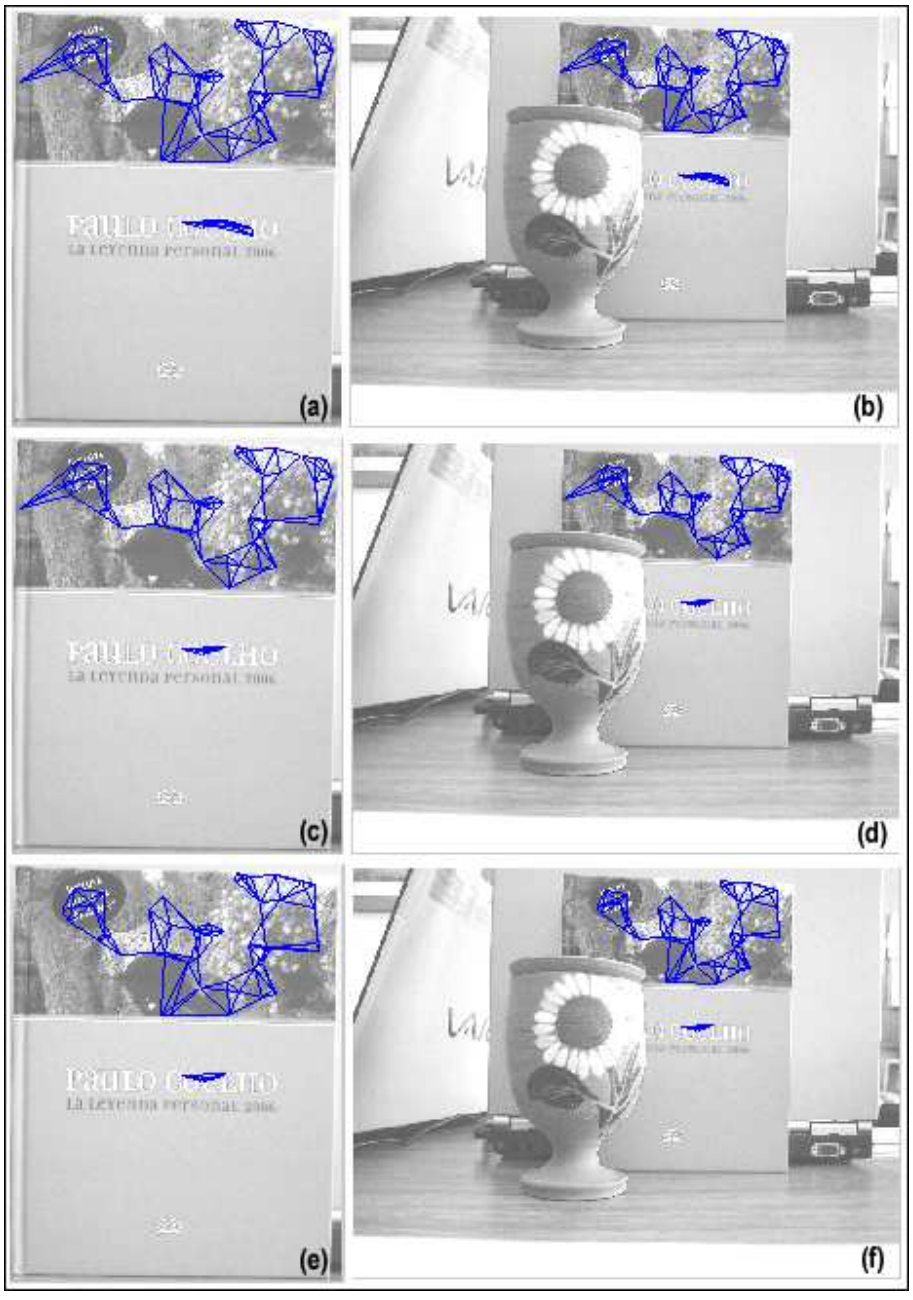


Figura 5.3: Grafos resultantes de robustez a correspondencias erróneas aleatorias del objeto *Agenda*. Del lado izquierdo se muestran los grafos sobre el modelo, con *outiers* del: (a) 15 %, (b) 35 % y (c) 55 %. Del lado derecho ((b),(d) y (f)), se muestran los grafos correspondientes sobre la imagen de la escena.

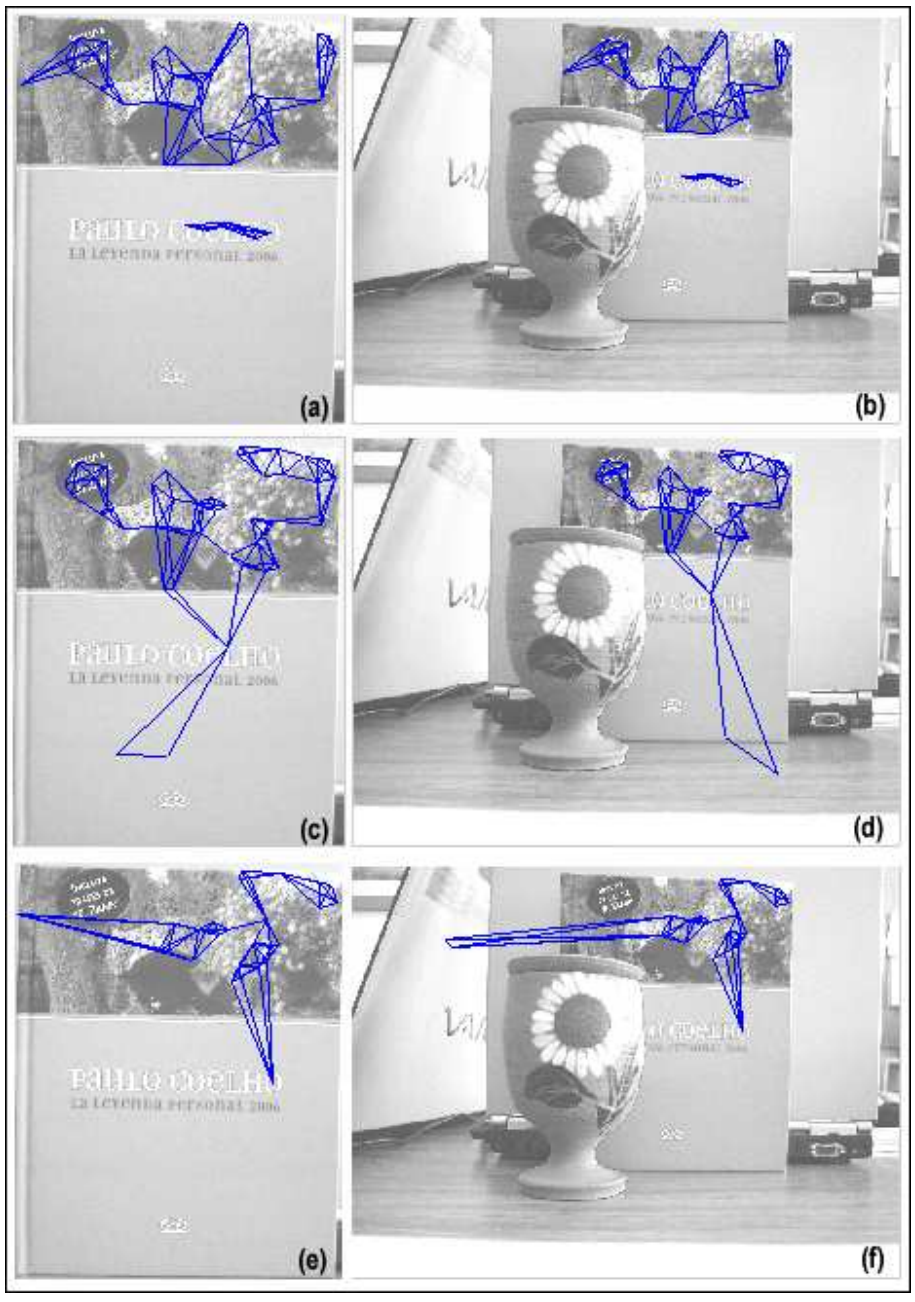


Figura 5.4: Grafos resultantes de robustez a correspondencias erróneas aleatorias del objeto *Agenda*. Del lado izquierdo se muestran los grafos sobre el modelo, con correspondencias erróneas del: (a) 75 %, (b) 85 % y (c) 95 %. Del lado derecho ((b),(d) y (f)), se muestran los grafos correspondientes sobre la imagen de la escena.

5.2. Pruebas de robustez a correspondencias erróneas reales

Las pruebas anteriores mostraron que el algoritmo se comporta bien ante un ruido aleatorio, sin embargo cabe la pregunta de cómo se comportará cuando las entradas contienen ruido de manera natural y que no es aleatorio. Este tipo de ruido ocurre cuando el detector de características junto con el método usado para generar las correspondencias iniciales no son muy buenos y generan una entrada con muchas correspondencias erróneas.

Para esta prueba se tomaron 10 parejas de imágenes a ser usadas en una aplicación de estimación de movimiento y se les aplicó una extracción de características muy simple basada en el *gradiente* (puntos que son máximos locales de grandiente de intensidad en una ventana de radio 10) y una mucho más compleja llamada *SIFT*. A los puntos de interés resultantes del método basado en *gradiente* se les hizo corresponder usando una matriz de correlación, lo cual produce un emparejamiento inicial muy ruidoso. Por otro lado, a las características *SIFT* se les hizo corresponder usando el algoritmo de *BBF*, lo cual produce entradas muy poco ruidosas. Esto nos da dos entradas extremas y además se diseñaron 2 filtros que nos permiten tener niveles de ruido intermedios.

Sea (f_i, f_i') la iésima correspondencia inicial con coordenadas de pixel $P_i = (x_i, y_i)$ y $P_i' = (x_i', y_i')$ respectivamente, en donde f_i es la iésima característica *SIFT* de la escena y f_i' del modelo. Sea también $C(P_i, P_i')$ la medida de similitud entre las apariencias locales en las vecindades de P_i y P_i', la cual está basada en la *correlación de Pearson*.

Los dos filtros diseñados se definen como:

1. *Filtro uno a uno*. Consiste en eliminar aquellas correspondencias que vayan o provengan del mismo pixel. En otras palabras, la correspondencia (P_i, P_i') se elimina si $\ni (P_j, P_j')$ tal que
$$p_i = p_j \text{ o } p_i' = p_j'$$
2. *Filtro calidad correlación*. Se eliminan aquellas correspondencias que cumplan con:
   a) baja similitud: $C(P_i, P_i') \leq C_{min}$, en donde C_{min} es un umbral que indica la correlación mínima necesaria para poder establecer una correspondencia,
   b) baja distintividad: es decir, existe también P_i'' que satisface la condición de que $C(P_i, P_i'')/C(P_i, P_i') \approx 1$ y
   c) unidireccionalidad: es decir, que el punto que tiene mayor correlación con P_i es P_i', pero el punto que tiene mayor correlación con P_i' NO es P_i.

El algoritmo de *Transformación de Grafos* dió resultados similares para las 10 parejas de imágenes. A continuación se presentan los resultados obtenidos al aplicarse a 3 de éstas parejas.

5.2.1. Pareja *UNO*

La primer pareja de imágenes representa un simple desplazamiento horizontal de la cámara. Este movimiento se puede observar en los resultados presentados en la Fig. 5.5, en donde (a)-(c) corresponden al resultado de *gradiente* y (d)-(e) a los de *SIFT*. La Fig. 5.6 muestra en la parte superior las correspondencias iniciales de entrada al algoritmo de *Transformación de Grafos*. Estas entradas son resultado de la extración de características por *gradiente* en donde (a) es la entrada sin ningún filtro, (b) con el *filtro uno a uno* y (c) con el *filtro calidad correlación*. En la parte inferior se muestran las correspondencias encontradas, (d), (e) y (f), respectivamente.

Para el caso de la extracción de características usando *SIFT*, se muestran en la Fig. 5.7. En la parte superior se presentan las entradas al algoritmo, (a) sin filtros y (b) con el *filtro uno a uno*. En la parte inferior muestra los resultados respectivos, (c) y (d). Los grafos resultantes para el caso de la extracción de características basada en *gradiente* se muestran en la Fig. 5.8. Las imágenes (a)-(b) para el caso de entrada sin filtros, (c)-(d) para el *filtro uno a uno* y (e)-(f) para el *filtro calidad correlación*. Así mismo, la Fig. 5.9 muestra los grafos pero ahora con la extracción de características *SIFT*, (a)-(b) sin filtros y (c)-(d) con el *filtro uno a uno*.

El Cuadro 5.2 presenta los datos numéricos de los resultados obtenidos: detector usado, número de correspondencias iniciales obtenidas con *BBF* (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), porcentaje de ruido, número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg).

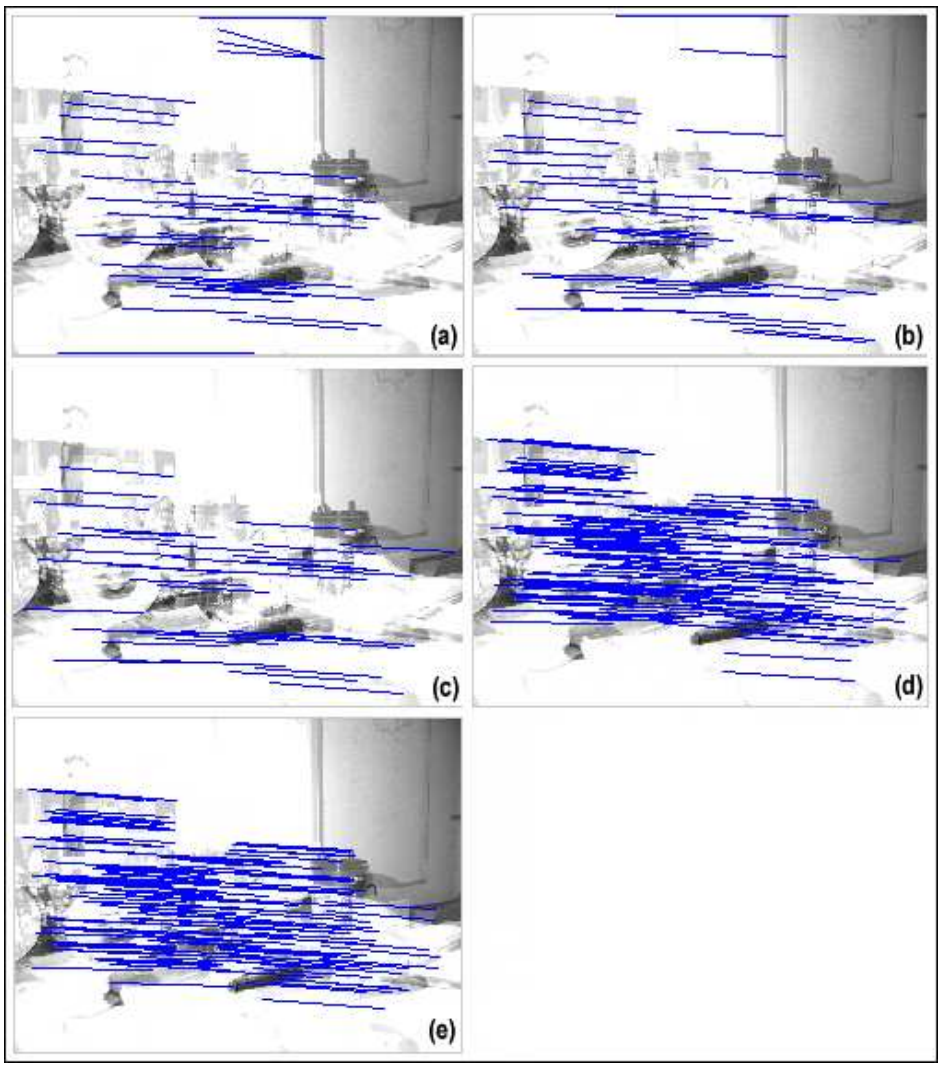


Figura 5.5: Pareja *UNO*, imágenes del movimiento encontrado. (a) gradiente sin filtro, (b) gradiente con *filtro uno a uno*, (c) gradiente con *filtro calidad correlación*, (d) SIFT sin filtros y (e) SIFT con *filtro uno a uno*.

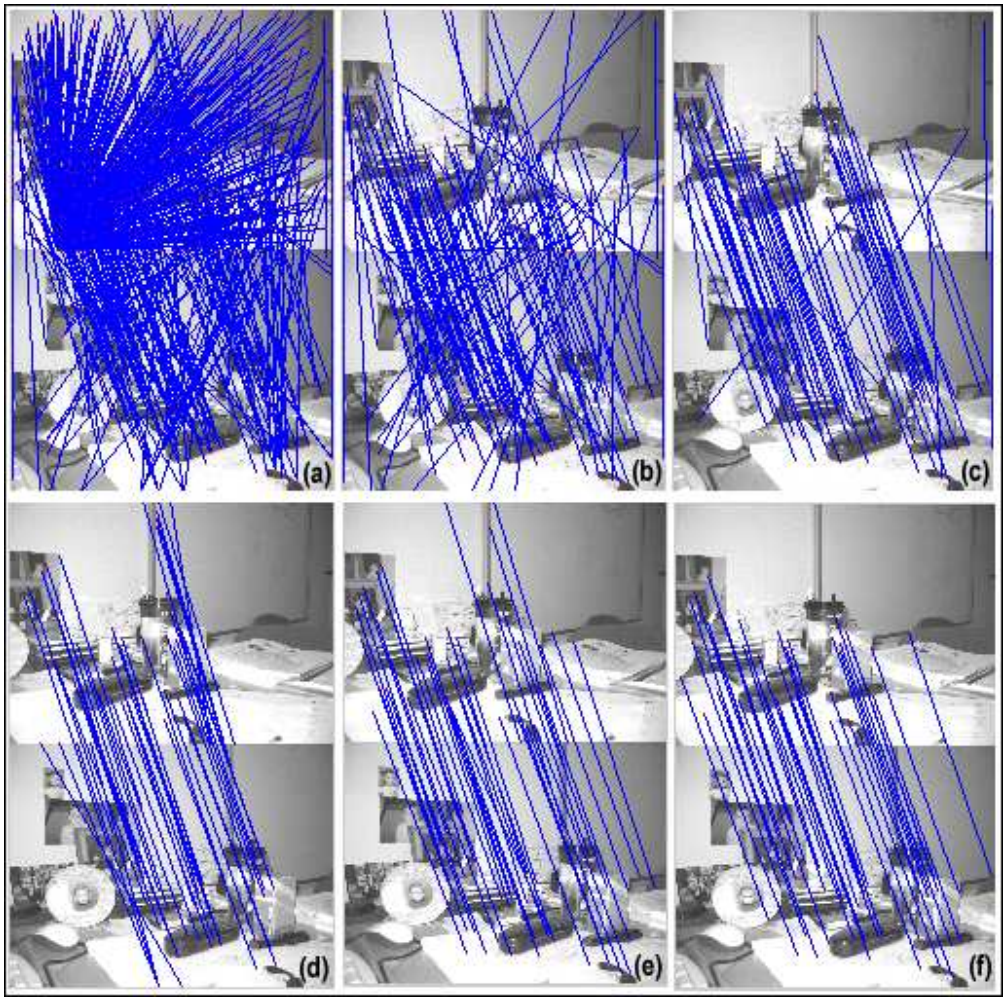


Figura 5.6: Pareja *UNO* con *gradiente*. La parte superior muestra las entradas al algoritmo sin filtros (a), con *filtro uno a uno* (b) y con *filtro calidad correlación* (c). La parte inferior muestra los resultados respectivos, (d),(e) y (f).

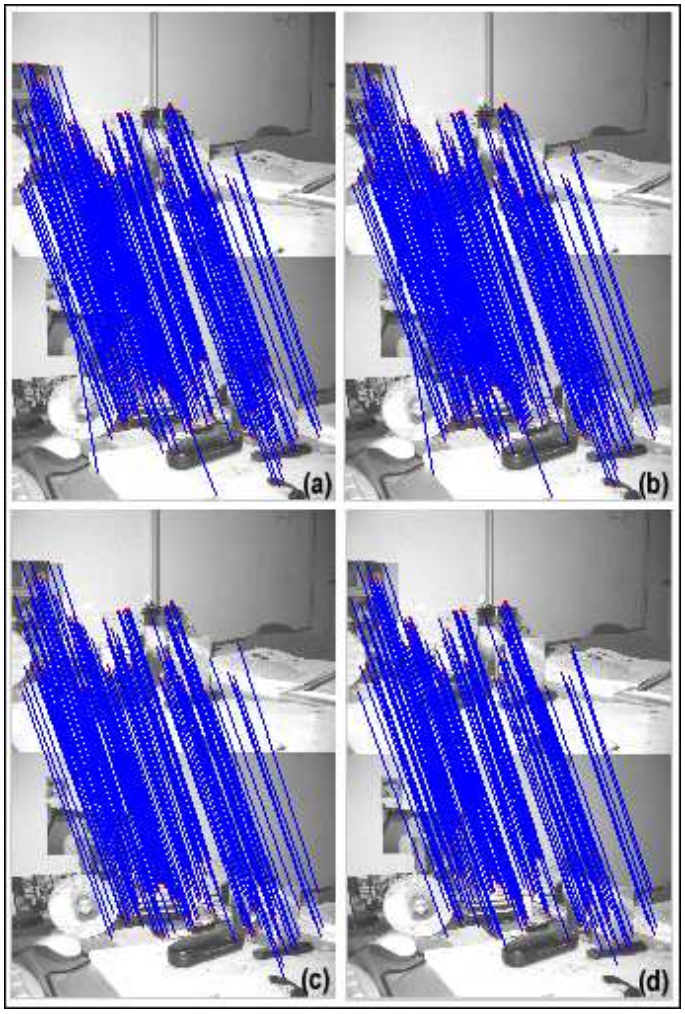


Figura 5.7: Pareja *UNO* con SIFT. La parte superior muestra las entradas al algoritmo sin filtros (a) y con *filtro uno a uno* (b). La parte inferior muestra los resultados respectivos, (c) y (d).

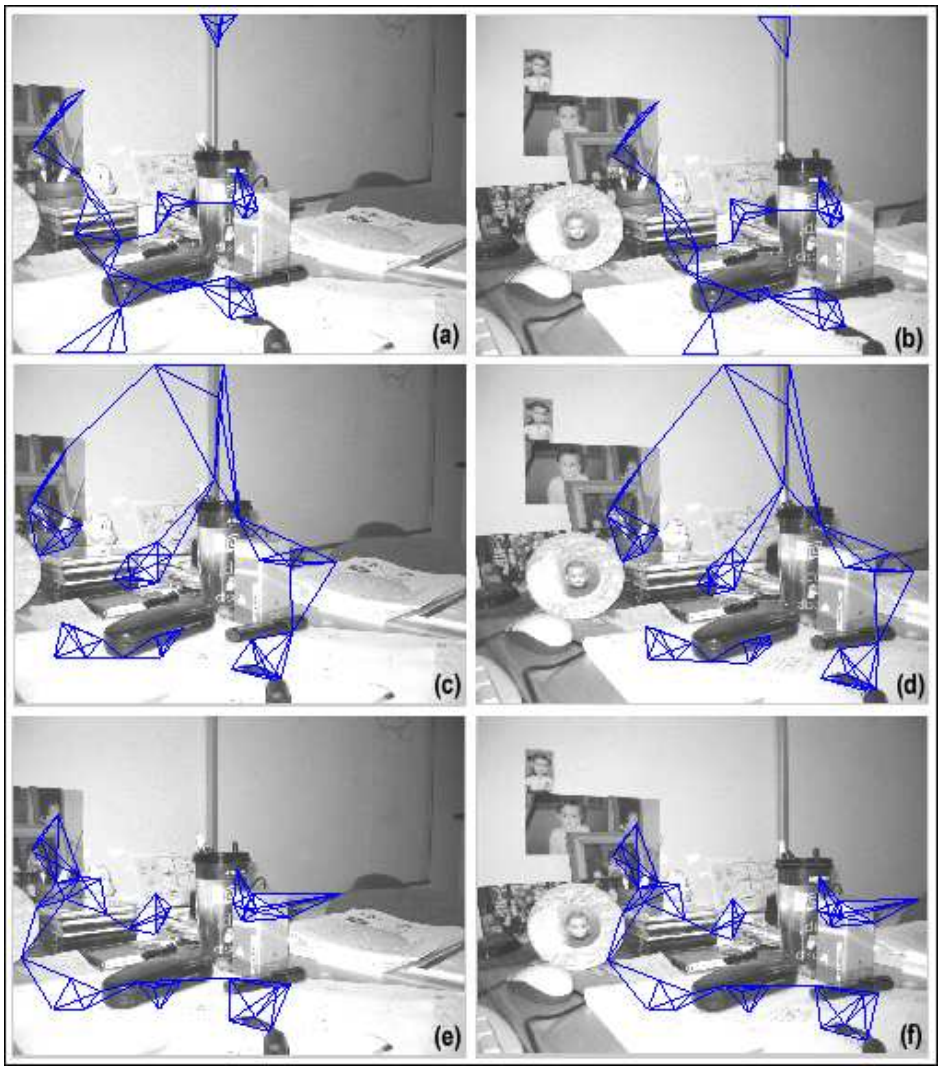


Figura 5.8: Grafos de la pareja *UNO*, con *gradiente*. Del lado izquierdo ((a), (c) y (e)) los grafos encontrados sobre la escena en el tiempo *t*. Del lado derecho ((b),(d),(f)) los grafos encontrados sobre la escena en el tiempo *t+1*, respectivamente. (a)-(b) sin filtros, (c)-(d) con *filtro uno a uno* y (e)-(f) con *filtro calidad correlación*.

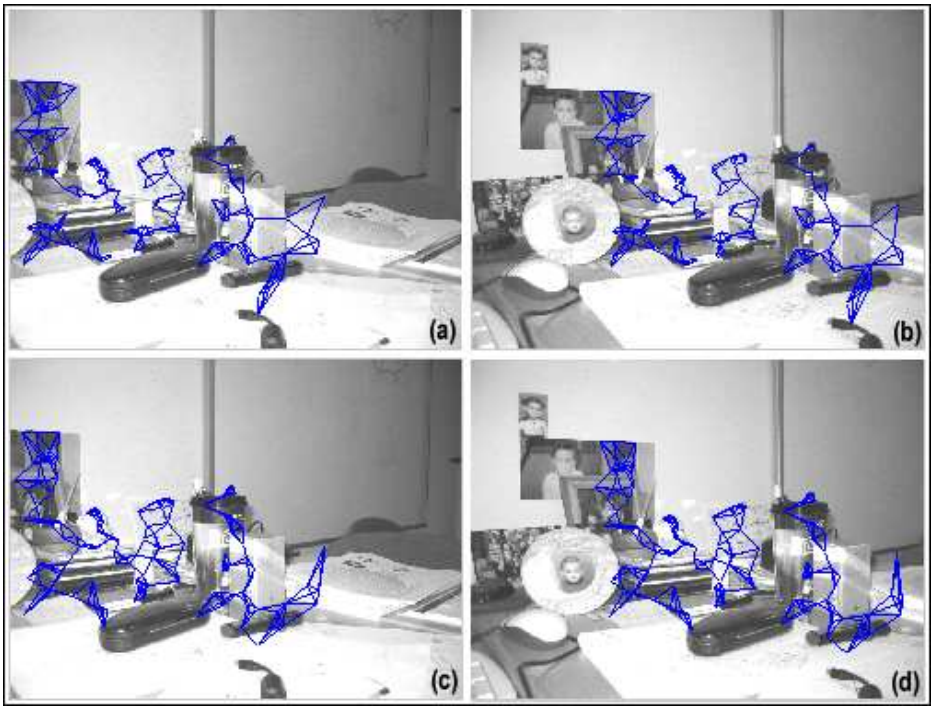


Figura 5.9: Grafos de la pareja *UNO*, con *SIFT*. Del lado izquierdo ((a) y (c)) los grafos encontrados sobre la escena en el tiempo *t*. Del lado derecho ((b) y (d)) los grafos encontrados sobre la escena en el tiempo *t+1*, respectivamente. (a)-(b) sin filtros y (c)-(d) con *filtro uno a uno*.

Detector	**CI**	**CIC**	**CIE**	**Ruido**	**CF**	**CFC**	**CFE**	**Iter.**	**Seg**
Gradiente sin filtros	478	67	411	86 %	46	36	10	433	23.09
Gradiente con filtro uno a uno	120	55	65	54 %	42	40	2	79	.42
Gradiente con filtro calidad correlación	62	54	8	13 %	38	38	0	23	0.09
SIFT sin filtros	225	205	20	8.9 %	192	192	0	34	1.03
SIFT con filtro uno a uno	194	192	2	1 %	151	151	0	44	.89

Cuadro 5.2: Resultados robustez al ruido de la pareja *UNO*: detector, número de correspondencias iniciales (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), Porcentaje de correspondencias erróneas, Número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg).

5.2.2. Pareja *SEIS*

La pareja *SEIS* refleja un movimiento un poco más complejo que la pareja *UNO*. En la Fig. 5.10 se puede ver la dirección del movimiento, en donde (a)-(c) corresponden al resultado de *gradiente* y (d)-(e) a los de *SIFT*. La Fig. 5.11 muestra en la parte superior las correspondencias iniciales de entrada al algoritmo de *Transformación de Grafos*. Estas entradas son resultado de la extración de características por *gradiente* en donde (a) es la entrada sin ningún filtro, (b) con el *filtro uno a uno* y (c) con el *filtro calidad correlación*. En la parte inferior se muestran las correspondencias encontradas, (d), (e) y (f), respectivamente.

Para el caso de la extracción de características usando *SIFT*, se muestran en la Fig. 5.12. En la parte superior se presentan las entradas al algoritmo, (a) sin filtros y (b) con el *filtro uno a uno*. En la parte inferior muestra los resultados respectivos, (c) y (d). Los grafos resultantes para el caso de la extracción de características basada en *gradiente* se muestran en la Fig. 5.13. Las imágenes (a)-(b) para el caso de entrada sin filtros, (c)-(d) para el *filtro uno a uno* y (e)-(f) para el *filtro calidad correlación*. Así mismo, la Fig. 5.14 muestra los grafos pero ahora con la extracción de características *SIFT*, (a)-(b) sin filtros y (c)-(d) con el *filtro uno a uno*.

El Cuadro 5.3 presenta los datos numéricos de los resultados obtenidos: detector usado, número de correspondencias iniciales (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), porcentajede ruido, número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg).

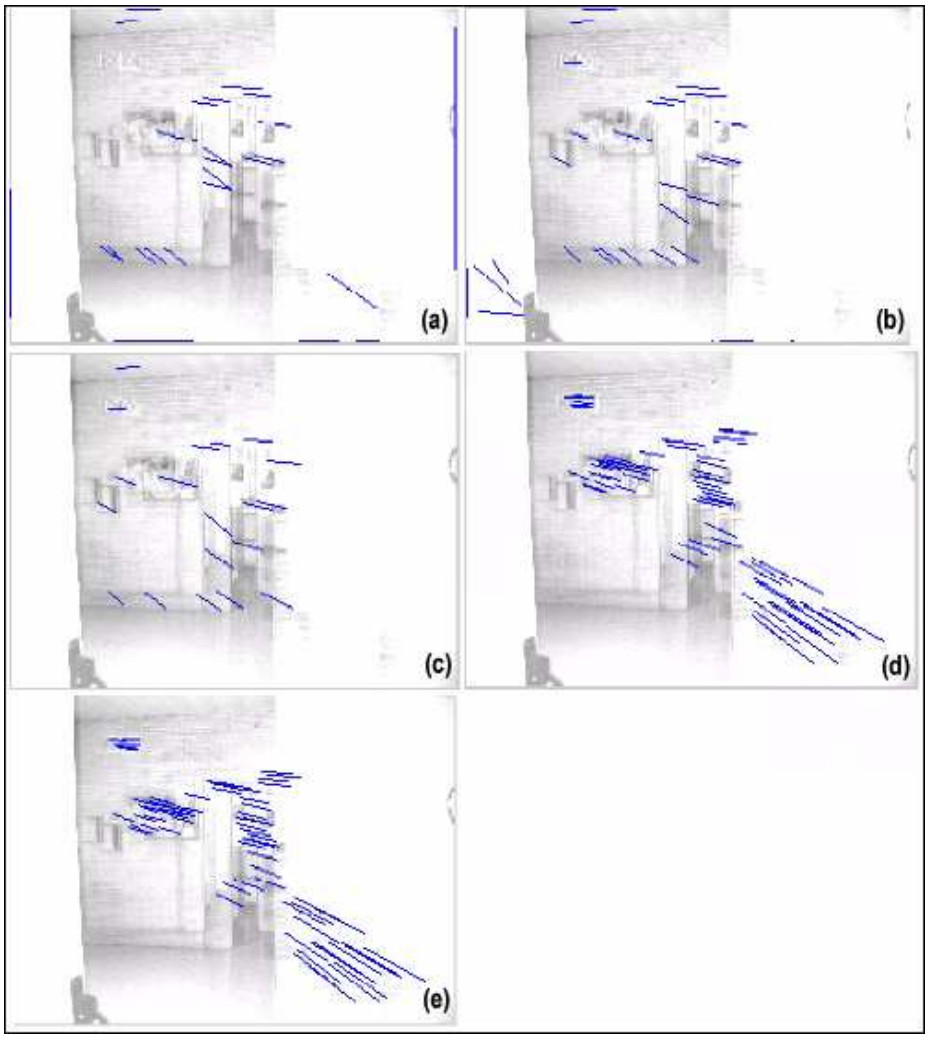


Figura 5.10: Pareja *SEIS*, imágenes del movimiento encontrado. (a) gradiente sin filtro, (b) gradiente con *filtro uno a uno*, (c) gradiente con *filtro calidad correlación*, (d) SIFT sin filtros y (e) SIFT con *filtro uno a uno*.

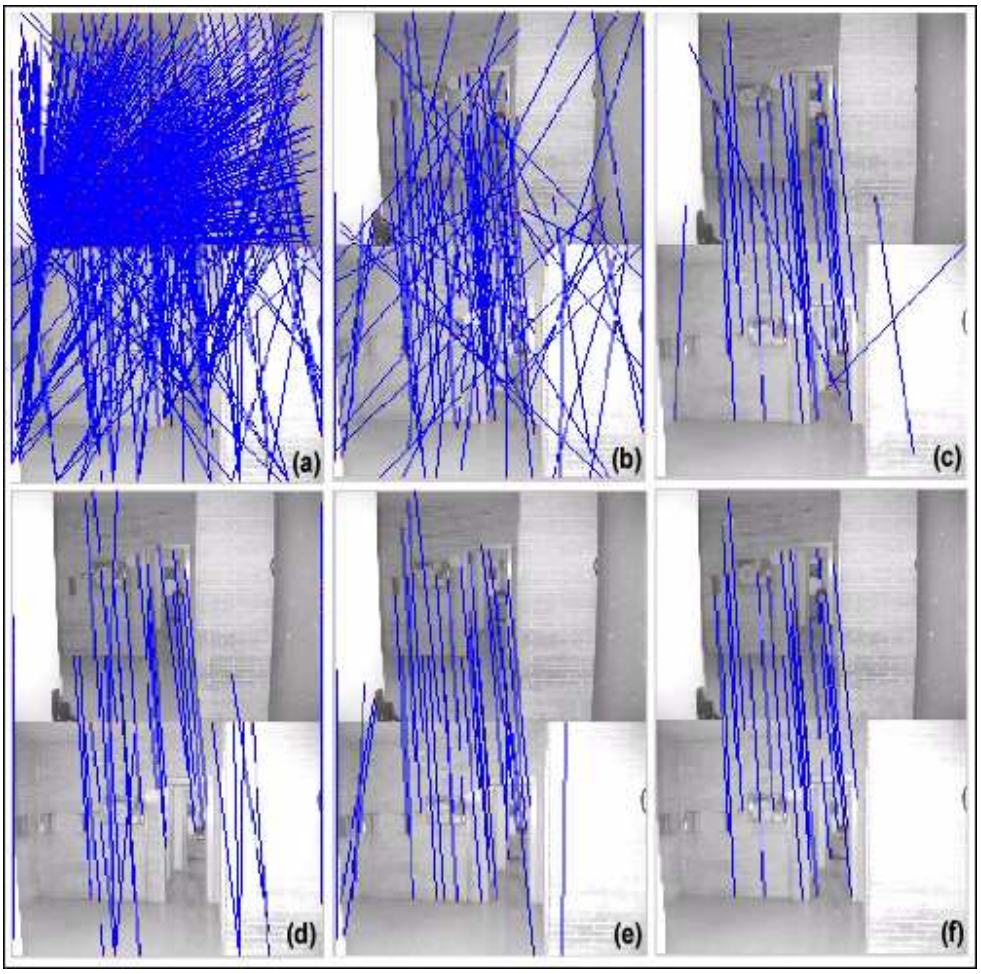


Figura 5.11: Pareja *SEIS* con *gradiente*. La parte superior muestra las entradas al algoritmo sin filtros (a), con *filtro uno a uno* (b) y con *filtro calidad correlación* (c). La parte inferior muestra los resultados respectivos, (d),(e) y (f).

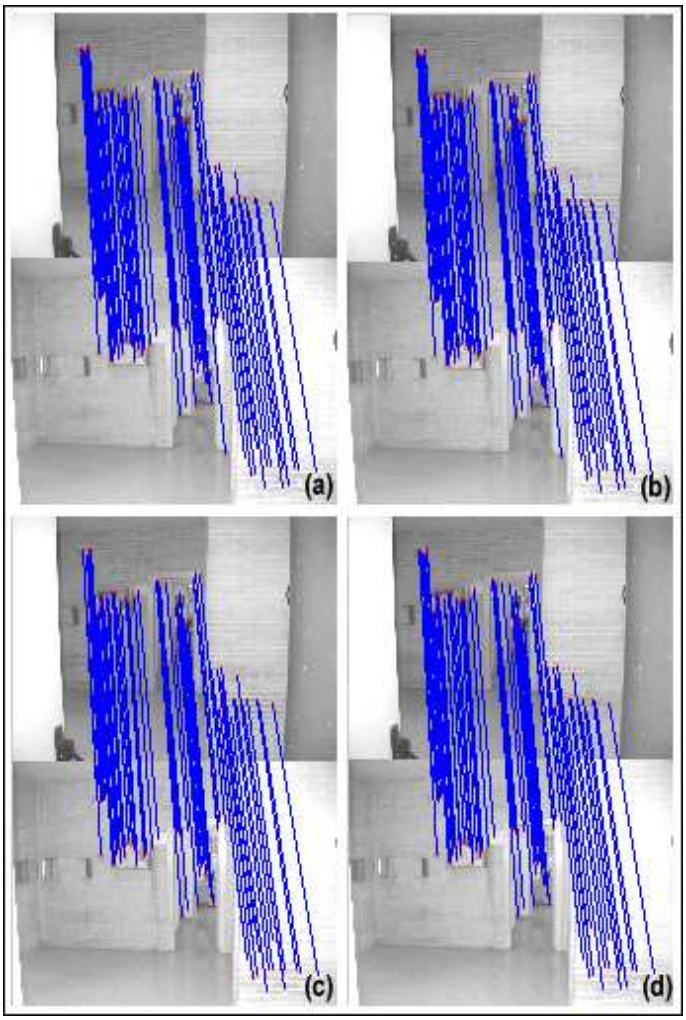


Figura 5.12: Pareja *SEIS* con SIFT. La parte superior muestra las entradas al algoritmo sin filtros (a) y con *filtro uno a uno* (b). La parte inferior muestra los resultados respectivos, (c) y (d).

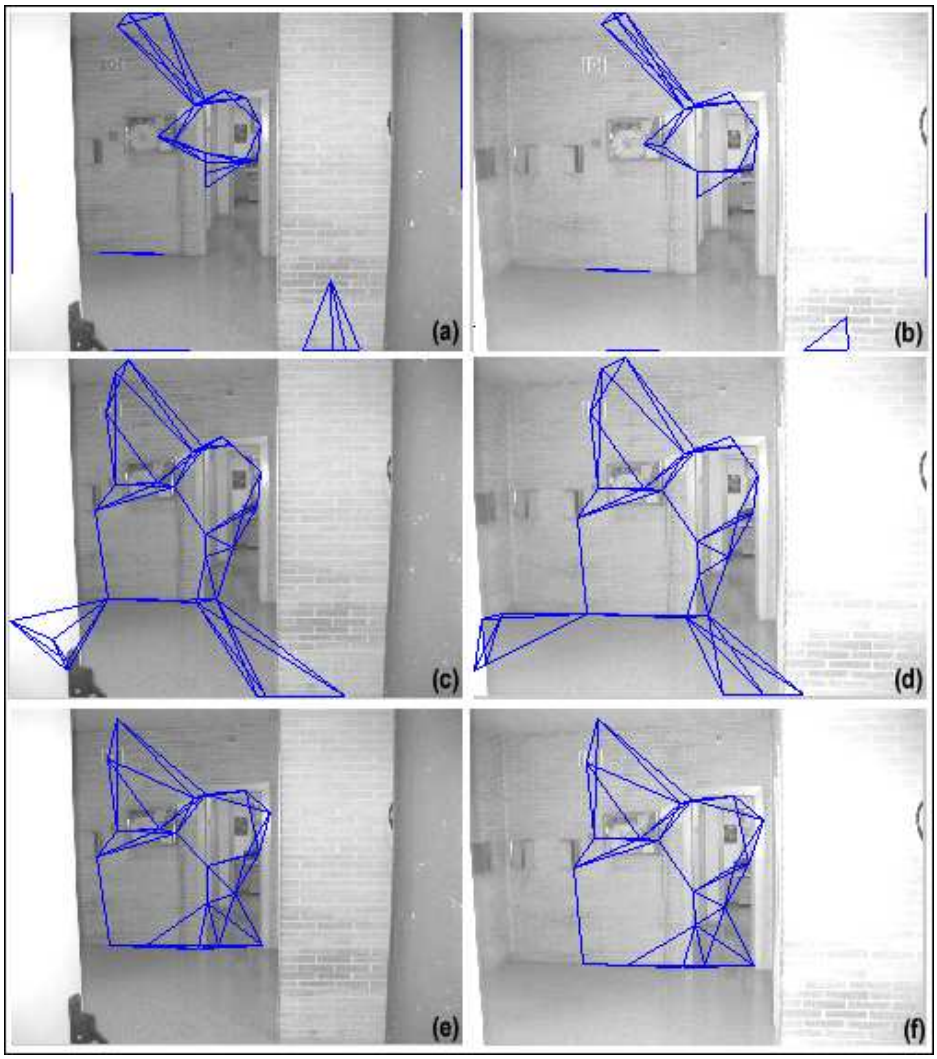


Figura 5.13: Grafos de la pareja *SEIS*, con *gradiente*. Del lado izquierdo ((a), (c) y (e)) los grafos encontrados sobre la escena en el tiempo *t*. Del lado derecho ((b),(d),(f)) los grafos encontrados sobre la escena en el tiempo *t+1*, respectivamente. (a)-(b) sin filtros, (c)-(d) con *filtro uno a uno* y (e)-(f) con *filtro calidad correlación*.

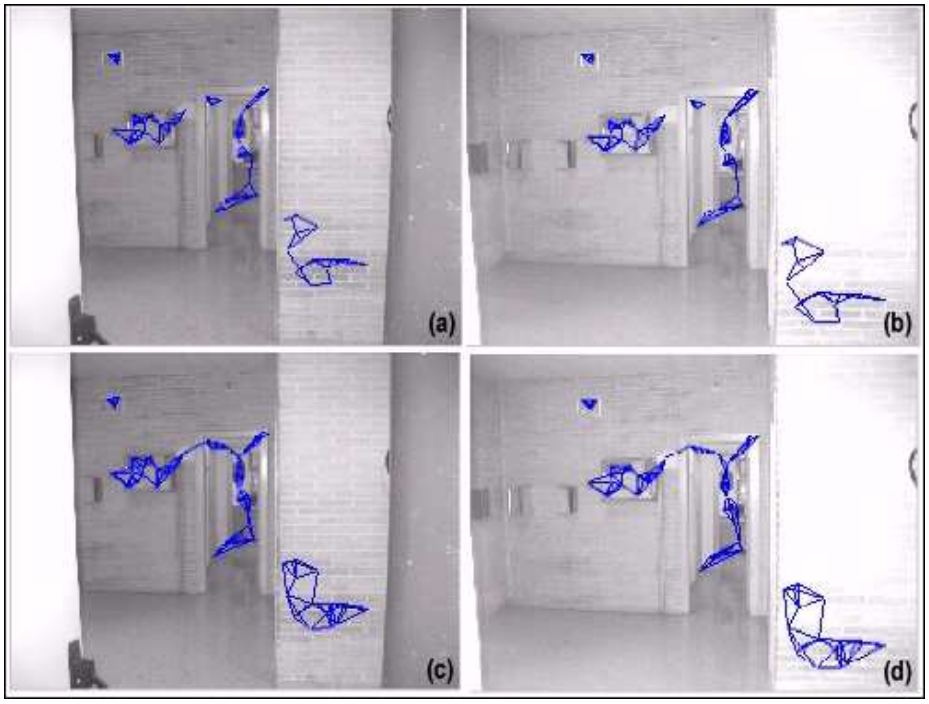


Figura 5.14: Grafos de la pareja *SEIS*, con *SIFT*. Del lado izquierdo ((a) y (c)) los grafos encontrados sobre la escena en el tiempo *t*. Del lado derecho ((b) y (d)) los grafos encontrados sobre la escena en el tiempo *t+1*, respectivamente. (a)-(b) sin filtros y (c)-(d) con *filtro uno a uno*.

Detector	CI	CIC	CIE	%	CF	CFC	CFE	Iter.	Seg
Gradiente sin filtros	559	52	507	90.7 %	42	20	22	518	36.6
Gradiente con filtro uno a uno	87	28	58	67.8 %	19	13	6	68	.18
Gradiente con filtro calidad correlación	24	21	3	12.5 %	20	20	0	3	.01
SIFT sin filtros	96	90	6	6.25 %	86	86	0	11	.09
SIFT con filtro uno a uno	77	76	1	1.3 %	71	71	0	7	.04

Cuadro 5.3: Resultados robustez al ruido de la pareja *SEIS*: detector, número de correspondencias iniciales obtenidas con *BBF* (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), porcentaje de correspondencias erróneas, número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg).

5.2.3. Pareja *OCHO*

Finalmente, la pareja *OCHO* refleja otro tipo de movimiento. En la Fig. 5.15 se puede ver la dirección del movimiento, en donde (a)-(c) corresponden al resultado de *gradiente* y (d)-(f) a los de *SIFT*. La Fig. 5.16 muestra en la parte superior las correspondencias iniciales de entrada al algoritmo de *Transformación de Grafos*. Estas entradas son resultado de la extración de características por *gradiente* en donde (a) es la entrada sin ningún filtro, (b) con el *filtro uno a uno* y (c) con el *filtro calidad correlación*. En la parte inferior se muestran las correspondencias encontradas, (d), (e) y (f), respectivamente.

Para el caso de la extracción de características usando *SIFT*, se muestran en la Fig. 5.17. En la parte superior se presentan las entradas al algoritmo, (a) sin filtros y (b) con el *filtro uno a uno*. En la parte inferior muestra los resultados respectivos, (c) y (d). Los grafos resultantes para el caso de la extracción de características basada en *gradiente* se muestran en la Fig. 5.18. Las imágenes (a)-(b) para el caso de entrada sin filtros, (c)-(d) para el *filtro uno a uno* y (e)-(f) para el *filtro calidad correlación*. Así mismo, la Fig. 5.19 muestra los grafos pero ahora con la extracción de características *SIFT*, (a)-(b) sin filtros y (c)-(d) con el *filtro uno a uno*.

El Cuadro 5.4 presenta los datos numéricos de los resultados obtenidos: detector usado, número de correspondencias iniciales (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), porcentajede ruido, número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg).

5.2.4. Discusión

Los resultados mostraron que el algoritmo es más sensible a este tipo de ruido que al ruido aleatorio, en particular para el caso de entradas sin filtros. Esto se puede deber a que las correspondencias iniciales sin filtros tienen la relación particular de uno a muchos. Esto provoca que un punto valide la estructura de sí mismo tantas veces como correspondencias haya a él. Sin embargo, apesar de esta característica peculiar el algoritmo eliminó una gran parte de las correspondencias erróneas demostrando ser también robusto al ruido real. El peor de los casos fue cuando existía un 90.7 % de ruido y en el cual se encontraron únicamente 20 correspondencias correctas de 52 e indico que 22 eran correctas cuando en realidad eran erróneas. Con el 78.5 % y el 86 % se equivocó en 10 correspondencias, pero encontró poco mas del 50 % de las correspondencias correctas totales. Para el caso de las entradas con el *filtro uno a uno* se obtuvieron mucho mejores resultados. El mayor porcentaje de ruido fué del 67.8 % con el cual solo 6 correspondencias resultaron erróneas de 19 totales que encontró. Con las otras parejas de imágenes se equivocó únicamente en 2 correspondencias y encontró más del 50 % de las correspondencias correctas totales.

Finalmente, las entradas con el *filtro calidad correlación* y aquellas con características *SIFT* tuvieron porcentajes de ruido menores al 13 % que el algoritmo re-

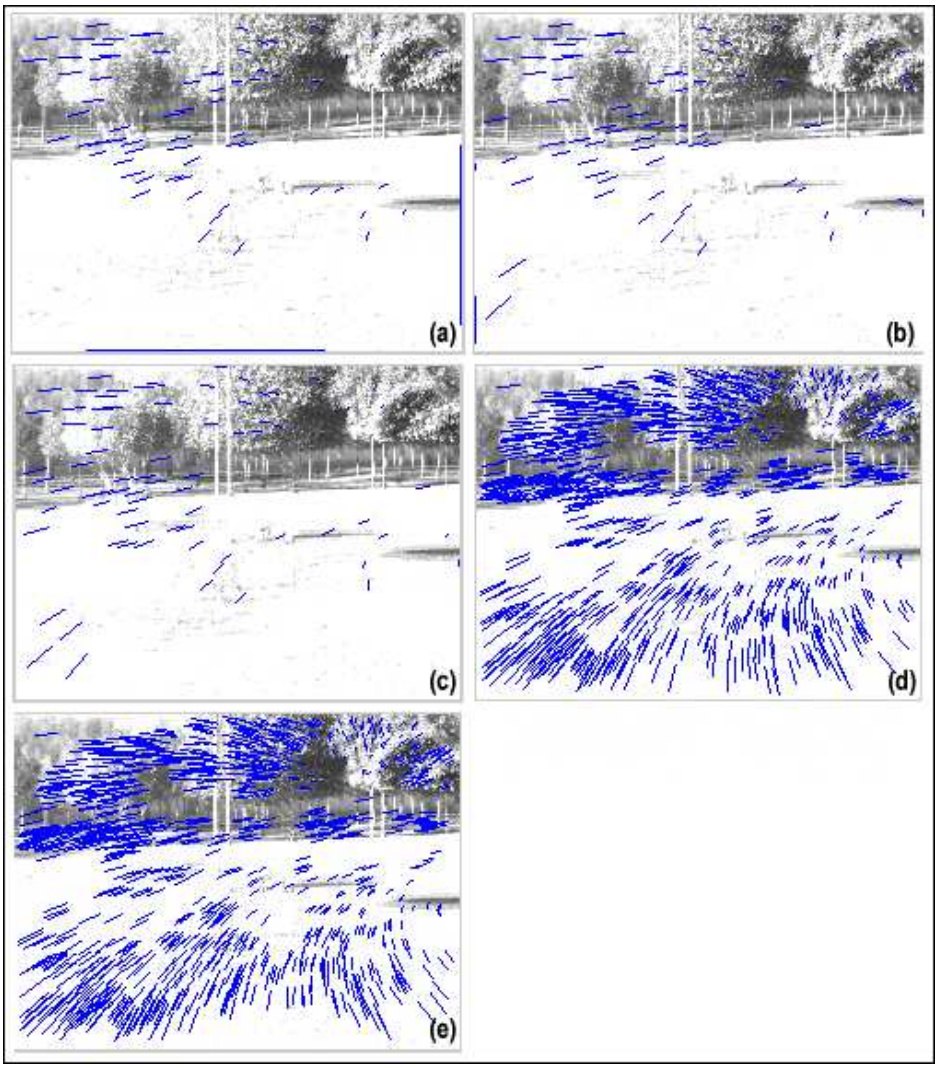


Figura 5.15: Pareja *OCHO*, imágenes del movimiento encontrado. (a) gradiente sin filtro, (b) gradiente con *filtro uno a uno*, (c) gradiente con *filtro calidad correlación*, (d) SIFT sin filtros y (e) SIFT con *filtro uno a uno*.

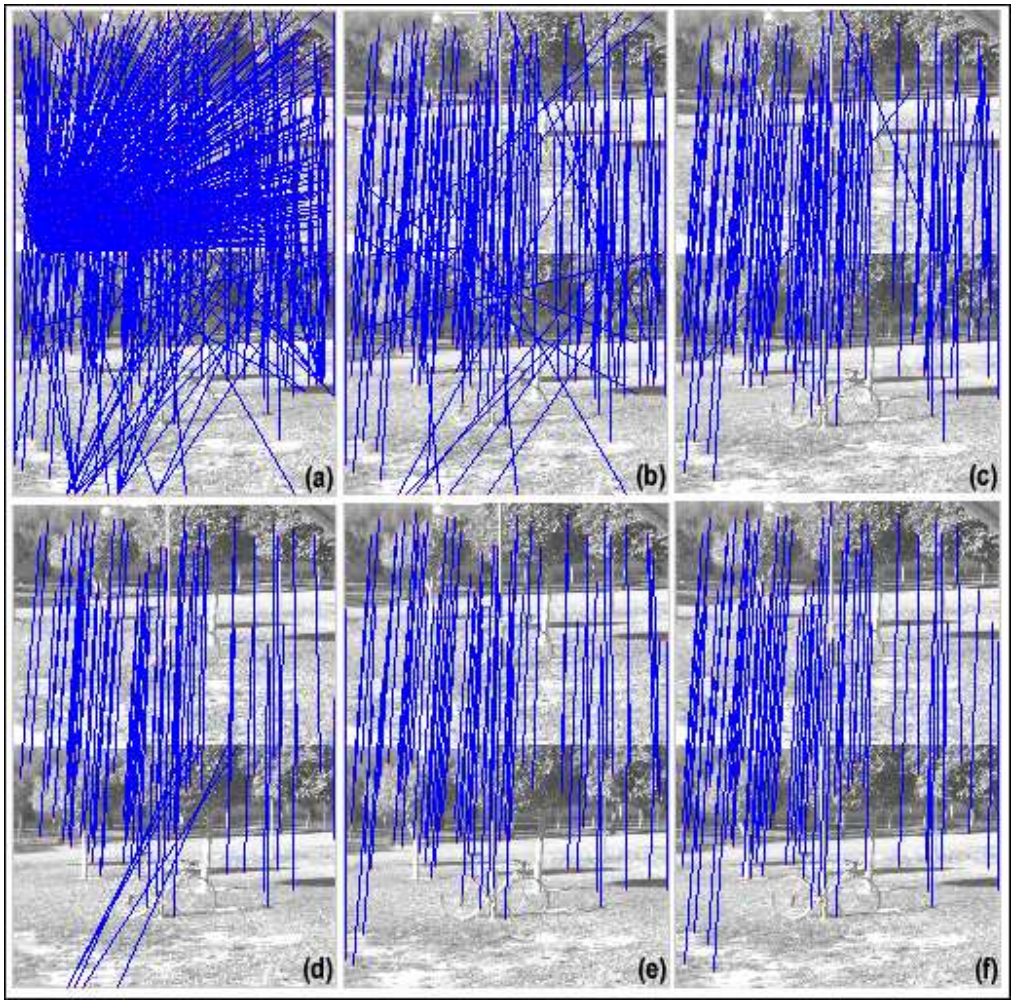


Figura 5.16: Pareja *OCHO* con correlación. La parte superior muestra las entradas al algoritmo sin filtros (a), con filtro uno a uno (b) y con filtro fco (c). La parte inferior muestra los resultados respectivos, (d),(e) y (f).

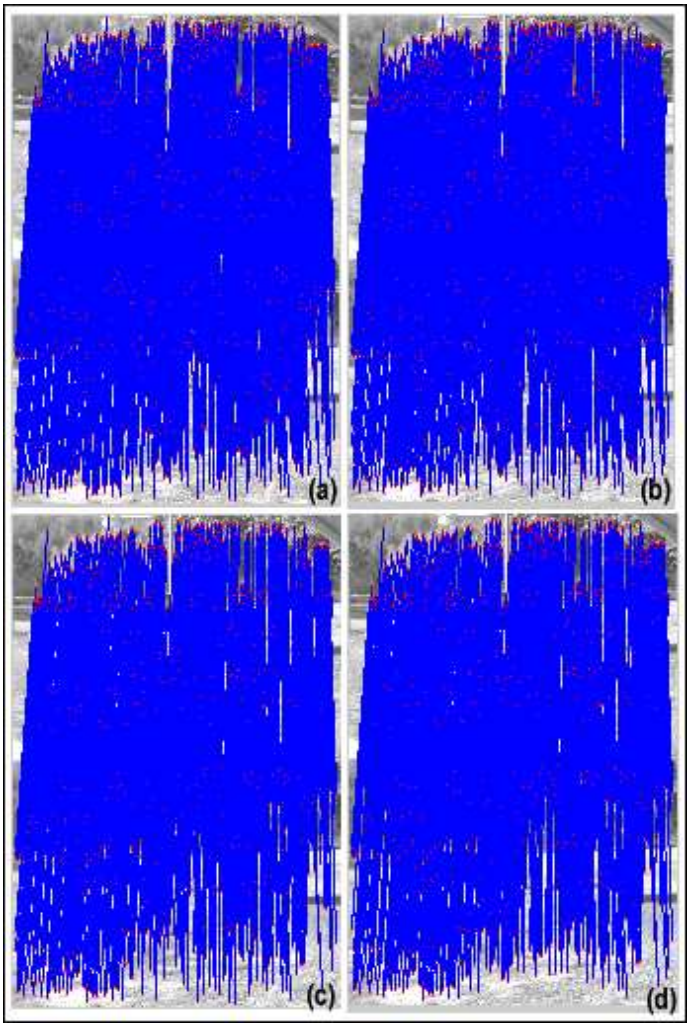


Figura 5.17: Pareja *OCHO* con SIFT. La parte superior muestra las entradas al algoritmo sin filtros (a) y con filtro uno a uno (b). La parte inferior muestra los resultados respectivos, (c) y (d).

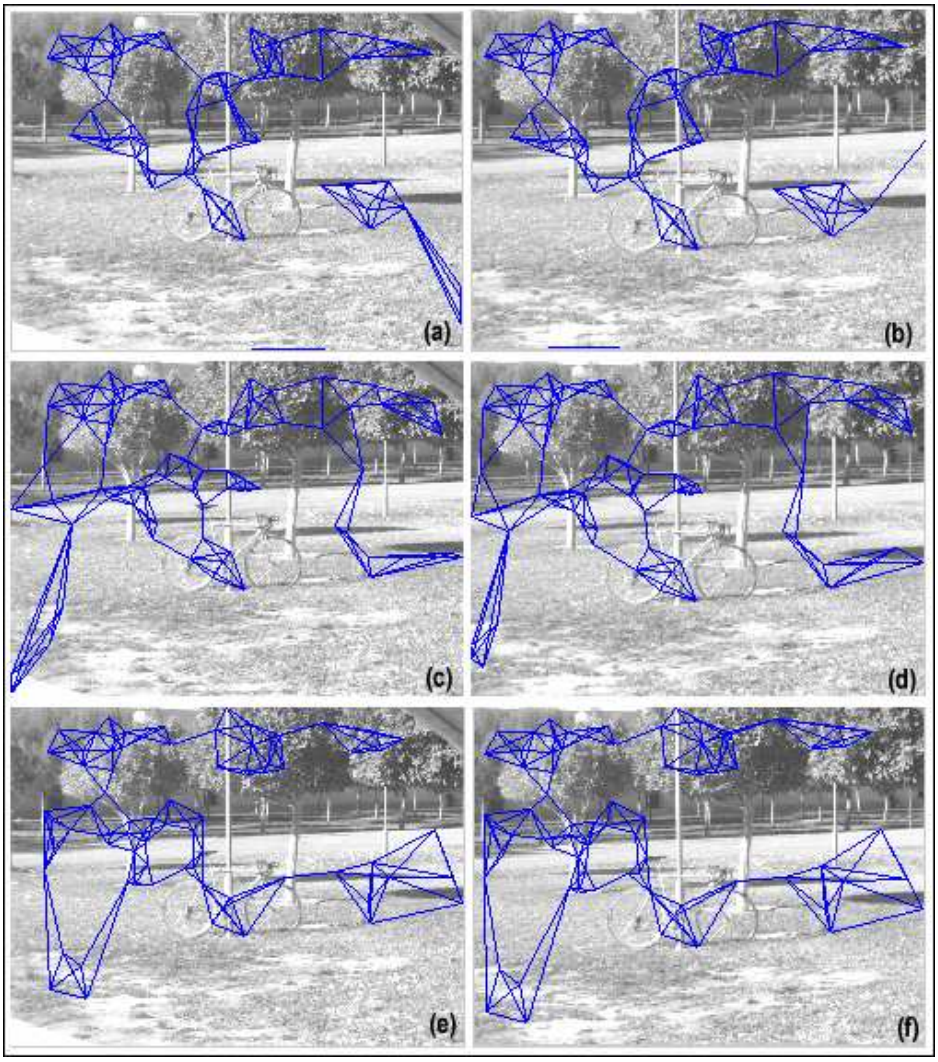


Figura 5.18: Grafos de la pareja *OCHO*, con *gradiente*. Del lado izquierdo ((a), (c) y (e)) los grafos encontrados sobre la escena en el tiempo *t*. Del lado derecho ((b),(d),(f)) los grafos encontrados sobre la escena en el tiempo *t+1*, respectivamente. (a)-(b) sin filtros, (c)-(d) con *filtro uno a uno* y (e)-(f) con *filtro calidad correlación*.

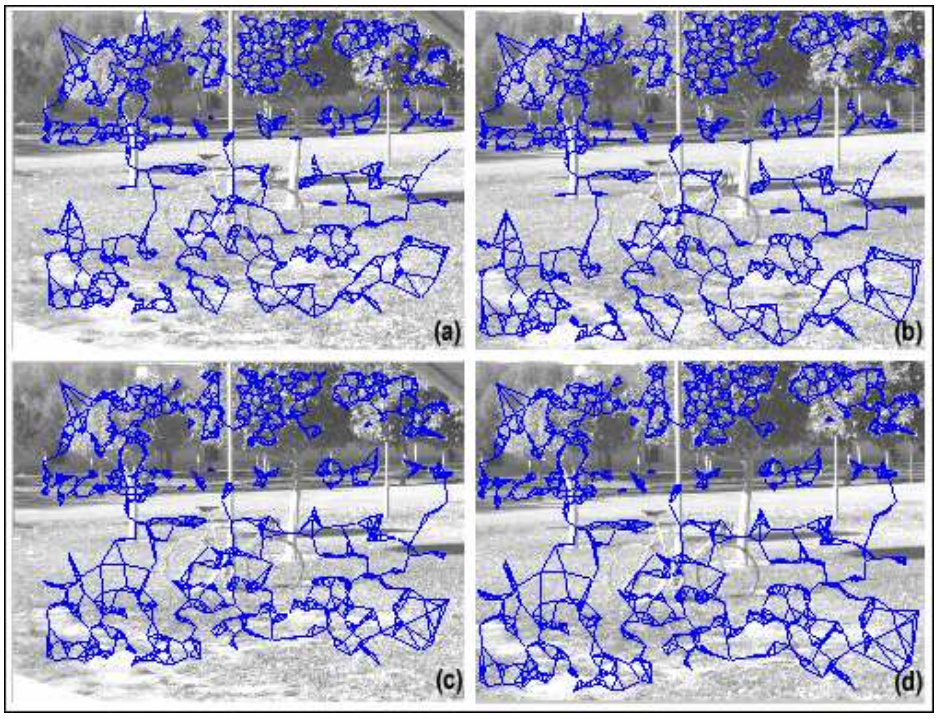


Figura 5.19: Grafos de la pareja *OCHO*, con *SIFT*. Del lado izquierdo ((a) y (c)) los grafos encontrados sobre la escena en el tiempo *t*. Del lado derecho ((b) y (d)) los grafos encontrados sobre la escena en el tiempo *t+1*, respectivamente. (a)-(b) sin filtros y (c)-(d) con *filtro uno a uno*.

Detector	**CI**	**CIC**	**CIE**	**%**	**CF**	**CFC**	**CFE**	**Iter.**	**Seg**
Gradiente sin filtros	574	123	451	78.5 %	78	68	10	497	39.07
Gradiente con filtro uno a uno	143	103	40	28 %	76	74	2	68	.58
Gradiente con filtro calidad correlación	107	104	3	2.8 %	75	75	0	33	.25
SIFT sin filtros	1545	1352	193	12.5 %	1334	1334	0	212	278.1
SIFT con filtro uno a uno	1361	1306	55	4 %	1165	1165	0	197	195.6

Cuadro 5.4: Resultados robustez al ruido de la pareja *OCHO*: detector, número de correspondencias iniciales (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), porcentaje de correspondencias erróneas, número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y Tiempo en segundos.

solvió correctamente sin ningún error.

Algo importante a notar es que las correspondencias erróneas generadas por el algoritmo se debieron en su mayoría a pequeñas variaciones en las posiciones de los pixels, por lo cual el flujo del movimiento entre las escenas se mantuvo. Esto se puede observar en las imágenes de movimiento de las Figuras 5.5, 5.10 y 5.15.

Como punto de comparación, se le dieron como entrada al algoritmo de *RANSACing Thin-Plate Splines* las entradas sin filtros y con el *filtro uno a uno* para la pareja *NUEVE* presentada en la sección 4.2.3. La Fig. 5.20 muestra las imágenes de movimiento: (a) la entrada sin filtros, (b) la salida del algoritmo de RANSACing Thin-Plate Splines y (c) la salida del algoritmo de *Transformación de Grafos*. La Fig. 5.21 muestra los grafos encontrados. Los resultados de ambos algoritmos se presentan en el Cuadro 5.5. En sus columnas muestra: nombre del algoritmo, número de correspondencias iniciales obtenidas con correlación(CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg). De la misma manera, la Fig. 5.22 muestra los resultados para la entrada con el *filtro uno a uno* y la Fig. 5.23 sus respectivos grafos encontrados. Así mismo, el Cuadro 5.6 presenta los resultados obtenidos.

Estos resultados hacen contrastar la gran sensibilidad al ruido de *RANSACing Thin-Plate Spline* contra la poca sensibilidad del algoritmo de *Transformación de Grafos*.

Algoritmo	CI	CIC	CIE	CF	CFC	CFE	Iter.	Seg
RANSAcing Thin-plate Splines	520	35	485	156	0	156	100	162.9
Transformación de grafos	520	35	485	25	13	12	494	29.1

Cuadro 5.5: Comparativo de robustez a correspondencias erróneas reales (*sin filtros*) con la pareja *NUEVE*: algoritmo, número de correspondencias iniciales obtenidas con correlación (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg).

Algoritmo	CI	CIC	CIE	CF	CFC	CFE	Iter.	Seg
RANSAcing Thin-plate Splines	93	36	57	28	10	18	100	5.26
Transformación de grafos	93	36	57	20	16	4	74	0.29

Cuadro 5.6: Comparativo de algoritmo estructural vs de alineación (*filtro uno a uno*) con la pareja *NUEVE*: algoritmo, número de correspondencias iniciales obtenidas con correlación (CI), número de correspondencias iniciales correctas (CIC), número de correspondencias iniciales erróneas (CIE), número de correspondencias finales (CF), número de correspondencias finales correctas (CFC), número de correspondencias finales erróneas (CFE), número de iteraciones (Iter.) y tiempo en segundos (Seg).

5.3. Síntesis

En este capítulo se presentaron dos experimentos para probar la robustez al ruido del algoritmo de *Transformación de Grafos*. En el primero se probó su robustez a correspondencias erróneas aleatorias, dando buenos resultados incluso con un 95 % de correspondencias erróneas. En el segundo experimento se probó su robustez a correspondencias erróneas reales, es decir, que los datos los contenían de manera natural. Para este tipo de ruido, el algoritmo mostró ser más sensible, sin embargo también soportó porcentajes altos de correspondencias erróneas, de hasta 54 % con resultados de menos de 2 errores.

Finalmente, el siguiente capítulo presenta las conclusiones.

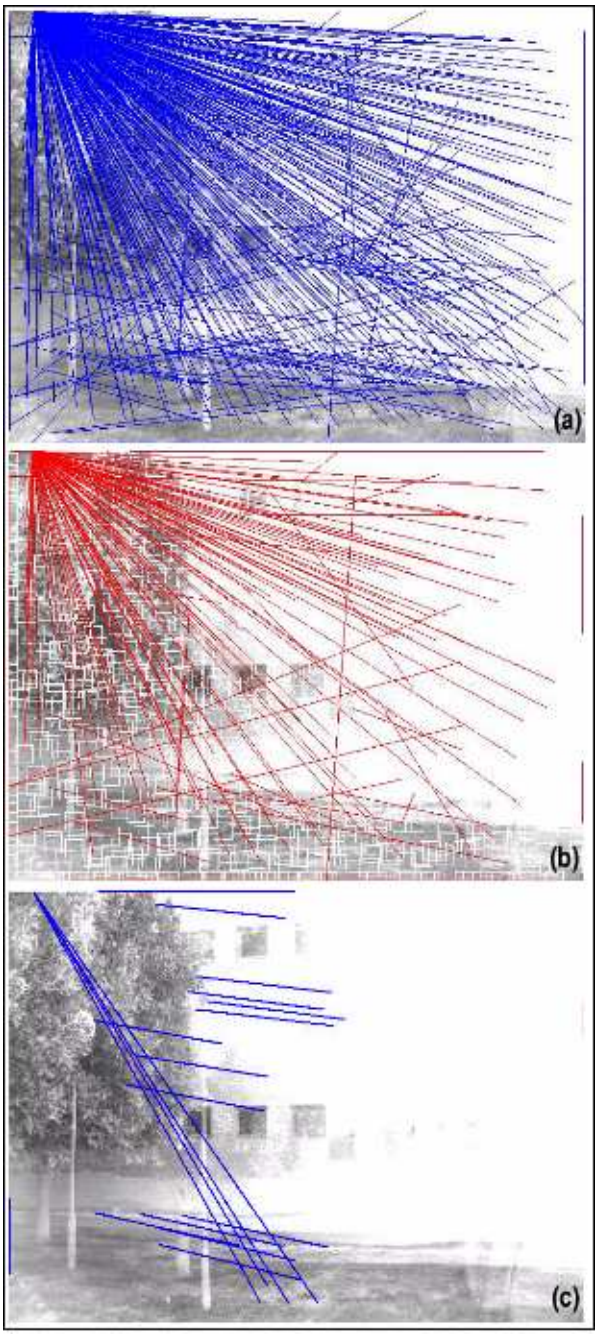


Figura 5.20: Pareja *NUEVE*, imágenes de movimiento de robustez a correspondencias erróneas reales: (a) la entrada sin filtros, (b) la salida del algoritmo de RANSACing Thin-Plate Splines y (c) la salida del algoritmo de *Transformación de Grafos*

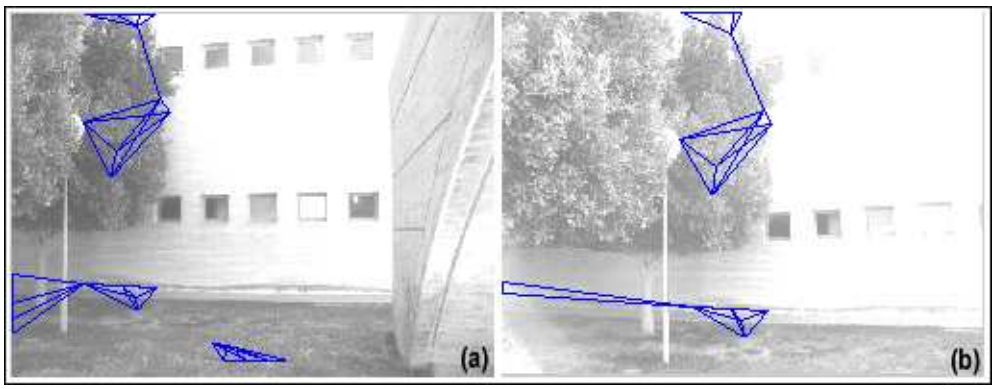


Figura 5.21: Pareja *NUEVE*, grafos resultantes sin filtros: (a) El grafo encontrado en la escena del tiempo $t-1$ y (b) el grafo encontrado en el tiempo t.

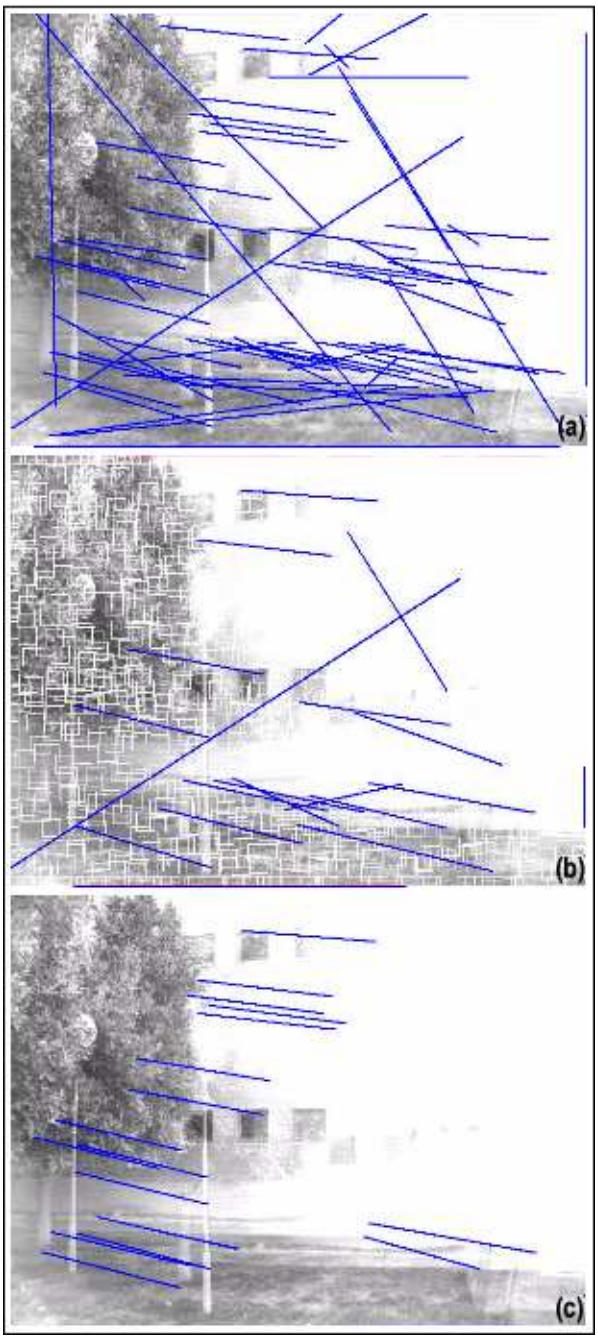


Figura 5.22: Pareja *NUEVE*, imágenes de movimiento de robustez a correspondencias erróneas reales: (a) la entrada con *filtro uno a uno*, (b) la salida del algoritmo de RANSACing Thin-Plate Splines y (c) la salida del algoritmo de *Transformación de Grafos*

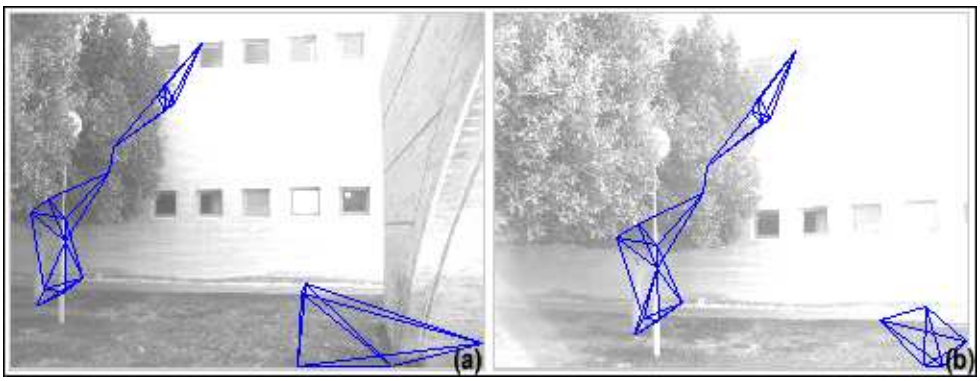


Figura 5.23: Pareja *NUEVE*, grafos resultantes con *filtro uno a uno*: (a) El grafo encontrado en la escena del tiempo $t-1$ y (b) el grafo encontrado en el tiempo t.

Capítulo 6

Conclusiones

Se propuso un algoritmo estructural para resolver el problema del reconocimiento de objetos basado en la correspondencia de características locales. Este algoritmo consiste en la transformación simultánea de dos grafos, uno para los puntos característicos de la imagen del modelo y otro para los de la escena. El algoritmo es sencillo y la implementación del algoritmo de fuerza bruta tiene una complejidad en tiempo de $O(n^3 Log(n))$, en donde cada iteración es de $O(n^2 Log(n))$ causada principalmente por la reconstrucción de los grafos.

Se comparó el algoritmo de *Transformación de Grafos* contra otro estructural llamado *Softassign* en sus versiones con *kernels* y *kernels y costos*. Los resultados mostraron que el algoritmo propuesto es superior en eficiencia y eficacia que el de *Softassign*, resolviendo todos los casos con error 0, en contraste con *Softassign* (en su versión de *kernel y costos*) que en el mejor de los casos daba como resultado practicamente la misma entrada.

También se comparó contra un algoritmo de aproximación de alineación llamado *RANSAcing Thin-plate splines*. Los resultados mostraron que el algoritmo propuesto resolvió (en la mayoría de los casos) mejor el problema que el algoritmo de aproximación de alineación. Demostró ser superior en cuanto al número de correspondencias correctas que lograba encontrar, al número de iteraciones requeridas, al tiempo y al error reportado.

Se realizaron 3 implementaciones del algoritmo de *Transformación de Grafos*, las dos primeras implementan el algoritmo de fuerza bruta, uno escrito en *Matlab* y otro escrito en *C*. La tercera implementación esta escrita en *C* e implementa el algoritmo optimizado, el cual optimiza la parte del algoritmo que corresponde a la reconstrucción de los grafos en cada iteración. Las pruebas de desempeño de los algoritmos mostraron que la implementación en *C* del algoritmo de fuerza bruta fue más eficiente que la implementación en *Matlab* hasta por 37.25 segundos para el caso de 1000 correspondencias iniciales y 40 iteraciones. Con respecto a las implementaciones en *C* del algoritmo de fuerza bruta contra el algoritmo optimizado, los experimentos reportaron que la versión optimizada fue superior hasta por 21.62 segundos para el caso de 559 correspondencias iniciales y 500 iteraciones. Mejor aún, la versión opti-

mizada generó gráficas muy parecidas a las de una constante indicando que éste es muy poco sensible en tiempo al número de iteraciones requeridas para converger y con ésto al número de correspondencias iniciales erróneas. Se llegó a la conclusión de que para considerar aspirar a realizar el reconocimiento de objetos en tiempos razonables se tome como límite 200 características iniciales, las cuales se observó son suficientes para realizar el reconocimientos de objetos y el algoritmo requeriría de aproximadamente 0.18 segundos para procesarlas.

Se hicieron pruebas con oclusiones parciales, con las cuales se comprobó que el algoritmo de *Transformación de Grafos* funciona correctamente. Esto gracias en parte a que el algoritmo permite que el grafo sea disconexo y con esto el que las características se encuentren acumuladas en grupos espacialmente lejanos, aunque lo que más influyó en que el reconocimiento de objetos con oclusiones parciales fuera exitoso es que el detector de características usado (*SIFT*) fue capaz de detectar las mismas características en ambas imágenes apesar de que había cambios de escala, de puntos de vista e incluso un poco de variación en la iluminación.

Se detectaron dos casos particulares en los cuales el algoritmo indica que algunas correspondencias son correctas cuando no lo son. El primer caso ocurre cuando se tienen (desde un inicio o en alguna iteración) únicamente $k + 1$ correspondencias, donde k es la conectividad del grafo. Se observó experimentalmente que la conectividad óptima es $k = 4$ y con esto que el algoritmo requiere de cúmulos de por lo menos 6 vértices. El segundo caso se puede presentar cuando existe más de una instancia de alguna parte del objeto y el algoritmo hace corresponder una parte de las características con un objeto y la otra parte con un segundo objeto.

El algoritmo demostró ser robusto al ruido de correspondencias erróneas aleatorias y a correspondencias erróneas reales, es decir, que existían de manera natural en las correspondencias iniciales. Para el caso de correspondencias erróneas aleatorias, logro encontrar el 40 % de las correspondencias correctas, con solo 2 errores y con un ruido del 95 %. Para porcentajes de ruido menores lograba encontrar el 83 % de las correspondencias correctas sin ningún error. En el caso de correspondencias erróneas reales, el algoritmo resultó ser más sensible a este tipo de ruido que al aleatorio. Este tipo de ruido tenía una característica particular, y ésta es que las correspondencias tienen una relación uno a muchos. Apesar de esta peculiaridad, el algoritmo eliminó una gran parte de las correspondencias erróneas demostrando ser también robusto al ruido real, hasta para un 54 % de correspondencias iniciales erróneas con respuestas menores a 2 errores.

Se probó el algoritmo con imágenes a ser utilizadas en diferentes aplicaciones como en el reconocimiento de objetos, estimación de movimiento, creación de mosaicos y reconocimiento de escenas. Las imágenes se tomaron en interiores y exteriores, usando objetos simples y objetos con texturas y se incluyeron pruebas con imágenes médicas de fondo de ojo.

Como trabajo a futuro se desea aplicar el algoritmo al problema de autolocalización basado en el reconocimiento de marcas visuales naturales, esto para robots o en dispositivos vestibles para uso de personas con discapacidad visual.

Bibliografía

[1] Abu-Mostafa, Y.S and Pslatis, D. Optical neural computing. *Scientific American*, 256, 66-73, 1987.

[2] Amit Yali. 2D Object Detection and Recognition. The MIT Press, Cambridge Massachusetts, 2002.

[3] Ando S. and Hontani H. Automatic visual searching and reading of barcodes in 3-d scene. In *IEEE International Vehicle Electronics Conference, IVEC 2001*, pp. 49-57, Sept. 2001.

[4] Beis J.S and Lowe D.G. Indexing without invariants in 3d object recognition. *PAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1000-1015, October 1999.

[5] Biederman I. Recognition by components: A theory of human image understanding. *Psychological Review*, 94, 115-147, 1987.

[6] Bischof H., Wildenauer H. and Leonardis A. Illumination insensitive eigenspaces. In *ICCV01*, pages I:233-238, 2001.

[7] Bischof H. and Leonardis A. Robust recognition of scaled eigenimages through a hierarchical approach. In *CVPR98*, pages 664-670, 1998.

[8] Boult T.T, Blum R.S, Nayar S.K, Allen P.K and Kender J.R. Advanced visual sensor systems. In *DARPA98*, pages 939-952, 1998.

[9] Brown M. and Lowe D. Invariant features from interest point groups. In *BMVC02*, 2002.

[10] Brown M. and Lowe D.G. Recognising panoramas. In *ICCV03*, pages 1218-1225, 2003.

[11] Burel G. and Hnocq H. Three-dimensional Invariants and Their Application to Object Recognition. *Signal Processing*, 45, pp. 1-22, 1995.

[12] Costa M.S, Shapiro L. Scene analysis using appearance-based models and relational indexing, *IEEE Computer Society International Symposium on Computer Vision*, pp. 103-108, 1995.

[13] Chang, N. S., Fu, K. S., Parallel Parsing of Tree Languages for Syntactic Pattern Recognition, *Pattern Recognition*, Vol 11, page 213, 1979.

[14] Christmas W., Kittler J. and Petrou M., Structural matching in computer vision using probabilistic relaxation, *IEEE Trans. Pattern Anal. Mach. Intell.* 17, 749-764, 1995.

[15] Dickinson S.J., Pentland A.P. and Rosenfeld A. 3-D Shape Recovery Using Distributed Aspect Matching. *Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, pp. 174-198, 1992.

[16] Dorko G. and Schmid C. Selection of scale-invariant parts for object class recognition. In *ICCV03*, pages 634-640, 2003.

[17] Evans A. C., Thacker J. W. E., Mayhew.The use of geometric histograms for model-based object recognition. *Proceedings of the 4th British Machine Vision Conference*, pp. 429-438, Sept. 1993.

[18] Ferrari V., Tytelaars T. and Van Gool L. Wide-baseline multiple-view correspondences. In *CVPR03*, pages I: 718-725, 2003.

[19] Fineman Mark. The Nature of Visual Illusion. Dover Publications, 1996.

[20] Fischler M.A. and Bolles R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Comm. of the ACM*, Vol 24, pp 381-395, 1981.

[21] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567-585, June 1989.

[22] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3D Object Modeling and Recognition Using Local Affine-Invariant Image Descriptors and Multi-View Spatial Constraints. *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231-259, March 2006.

[23] Funt B.V. and Finlayson G.D. Color constant color indexing. *PAMI:IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):522-529, May 1995.

[24] Gold S. and Rangarajan S., A Graduated Assignment Algorithm for Graph Matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(4), 377-388, 1996.

[25] Gould, J.L. How bees remember flower shapes. *Science*, 227, 1492-1494.

[26] Hamming Richard W. Error-detecting and error-correcting codes. *Bell System Technical Journal* 29(2):147-160, 1950.

[27] Harris C. and Stephens M. J. A combined corner and edge detector. In *Alvey88*, pages 147-152, 1988.

[28] Healey G. and Slater D.A. Using illumination invariant color histogram descriptors for recognition. In *CVPR94*, pages 355-360, 1994.

[29] Herault L., Horaud R., Veillon F. and Niez J.J. Symbolic image matching by simulated annealing. In *Proc. British Machine Vision Conference*, pp 319-324, Oxford, 1990.

[30] Hopfield J.J. Neural networks and physical systems with emergent colective computational abilities. *Proceedings of the National Academy of Science*, USA, 79, 2554-2558, 1982.

[31] Horaud R. and Skordas T. Stereo correspondence through feature grouping and maximal cliques. *IEEE Trans. Pattern Anal. Machine Intell.* vol. 11, pp. 1168-1180, 1989.

[32] Hu M. K. Visual Pattern Recognition by Moment Invariants. *IEEE Transactions in Information Theory*, 8, pp 179-187, 1962.

[33] Huet B., Hancock E. Line pattern retrieval using relational histograms, *IEEE Trans. Pattern Anal. Mach. Intell.* 21, 1363-1370, 1999.

[34] Iannizzotto G. et al. Badge3D for Visually Impaired. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[35] Jogan M. and Leonardis A. Robust localization using eigenspace of spinning-images. In *OMNIVIS00*, 2000.

[36] Klinger A. Pattern and search statistics. *Optimizing Methods in Statistics*, Academic Press, 1971.

[37] Koenderink, J. J. The structure of images. *Biological Cybernetics*, 50:363-396, 1984.

[38] Kohonen T. *Associative Memories: A System Theoretic Approach.* Berlin: Springer-Verlag, 1978.

[39] Krumm J. Eigenfeatures for planar pose measurement of partially occluded objects. In *CVPR96*, pages 55-60, 1996.

[40] Lamdan Y, Schwartz J.T and Wolfson H.J, Object recognition by affine invariant matching, in: *Proceedings of the conference on Computer Vision and Pattern Recognition*, pp. 335-344, 1998.

[41] Leonardis A. and Bischof H. Robust recognition using eigenimages. *Computer Vision and Image Understanding: CVIU*, 78(1):99-118, 2000.

[42] Leonardis A. and Bischof H. Dealing with occlusions in the eigen space approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 453-458, June 1996.

[43] Lepore E. and Pylyshyn. Rutgers University Lectures on Cognitive Science. Basel Blackwell publishers, pp. 172-207, 1999.

[44] Li S.Z. Shape Matching Based on Invariants, Chapter X in Omidvar (ed). *Shape Analysis* Progress in Neural Networks, Vol. 6, Ablex 1998.

[45] Lindeberg, T. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2):224-270, 1994.

[46] Lisin D. et al. Combining Local and Global Image Features for Object Class Recognition *IEEE Workshop on Learning in Computer Vision and Pattern Recognition*, San Diego, CA, June 2005.

[47] Lozano M.A and Escolano F. Structural Recognition with Kernelized Softassign. *IBERAMIA 2004*: pp 626-635, 2004.

[48] Lozano M.A and Escolano F. A Significant Improvement of Softassign with Diffusion Kernels. *SSPR& SPR 2004, Lecture Notes in Computer Science 3138*, pp. 76-84, 2004.

[49] Lowe D.G. Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 2004.

[50] Lowe D.G. Object recognition from local scale-invariant features. In *ICCV99*, pages 1150-1157,1999.

[51] Lowe D.G. Local feature view clustering for 3d object recognition. In *CVPR01*, pages I:682-688, 2001.

[52] Marr David. Vision: A computational investigation into the human representation and processing of visual information. New York: W. J. Freeman, 1982.

[53] Marr D. and Nishihara H. K. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society*, B 200:269-294, 1978.

[54] Martinez-Perez M. E., Hughes A. D., Stanton A. V., Thom S. A., Chapman N., Bharath A. A. and Parker K. H. Retinal Vascular Tree Morphology: A Semi-automatic Quantification. *IEEE Transactions on Biomedical Engineering*, Vol. 49, No. 8. pp 912-917. August 2002.

[55] Matas J. and Obdrzálek. Object recognition methods based on transformation covariant features. In *XII European Signal Processing Conference EUSIPCO-2004*, pp 1333-1336, Viena, Austria, Sept. 2004.

[56] Matas J., Chum O. Urban M. and Pajdla T. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, pages 384-393, 2002.

[57] Matas J., Obdrzálek S. and Chum O. Local affine frames for wide-baseline stereo. In *ICPR02*, August 2002.

[58] Messmer B. and Bunke H. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Trans. PAMI*, 20:493-505, 1998.

[59] Mikolajczyk K. and Schmid C. An affine invariant interest point detector. In *ECCV02*, page I:128 ff., 2002.

[60] Mikolajczyk K. and Schmid C. Indexing based on scale invariant interest points. In *ICCV01*, pages I:525-531, 2001.

[61] Mikolajczyk K. and Schmid C. A Performance Evaluation of Local Descriptors. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 10, october 2005.

[62] Miller, W. F., Shaw, A. C. Linguistic Methods in Picture Processing - a Survey. *AFIPS Fall Joint Computer Conference,* Vol 33, Part 1, p279, 1968.

[63] Mindru F., Moons T. and Van Gool L. Recognizing color patterns irrespective of viewpoint and illumination. In *Proceedings of the Computer Vision and Pattern Recognition*, pages 368-373, 1999.

[64] Moayer, B., Fu, K. S., A Tree System Approach for Fingerprint Pattern Recognition, *IEEE Transactions on Computers*, Vol C-25, 1976.

[65] Moghaddam B. and Pentland A. Probabilistic visual learning for object detection. In *International Conference on Computer Vision (ICCV'95*, pages 786-793, Cambridge, USA, June 1995.

[66] Mundy J.J. and Zisserman A. *Geometric Invariance in Computer Vision*. Book, 1992.

[67] Murase H. and Nayar S.K. Image spotting of 3d objects using parametric eigenspace representation. in *SCIA95*, pages 325-332, 1995.

[68] Nagao, M., Matsuyama, T., Mori, H., Structural Analysis of Complex Aerial Photographs, *Proceedings 6th IJCAI*, pp610-616, 1979.

[69] Nayar S.K, Nene S.A and Murase H. Real-time 100 object recognition system. In *ARPA96*, pages 1223-1228, 1996.

[70] Obdrzálek S. and Matas J. Object recognition using local affine frames on distinguished reguions. In *The British Machine Vision Conference (BMV02)*, September 2002.

[71] Obdrzálek S. and Matas J. Local affine frames for image retrieval. In *The Challenge of Image and Video Retrieval (CIVR2002)*, July 2002.

[72] Obdrzálek S. and Matas J. Image retrieval using local compact dct-based representation. In *DAGM 2003: Proceedings of the 25th DAGM Symposium*, pages 490-497, 9, 2003.

[73] Oppel J. J. Uber geometrisch-optische Tauschungen. *Jahresbericht phys. ver. Frankfurt*, pp 37-47, 1855.

[74] Pritchett P. and Zisserman A. Matching and reconstruction from widely separated views. *Lecture notes in Computer Science*, 1506:78-85, 1998.

[75] Pritchett P. and Zisserman A. Wide baseline stereo marching. In *ICCV*, pages 754-760, 1998.

[76] Robinson J. O. The Psychology of Visual Illusion. Dover Publications, 1998.

[77] Rothganger F., Lazebnik S., Schmid C., and Ponce J. 3d object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In *CVPR03*, pages II:272-277, 2003.

[78] Sanfeliu A and Fu K.S, A distance measure between attributed relational graph, *IEEE SMC* 13, 353-362, 1983.

[79] Shaffalitzky F. and Zisserman A. Geometric grouping of repeated elements within images. In *BMVC98*, 1998.

[80] Shaffalitzky F. and Zisserman A. Viewpoint invariant texture matching and wide baseline stereo. In *ICCV01*, pages II:636-643, 2001.

[81] Shaffalitzky F. and Zisserman A. Multi-view matching for unordered image sets, or 'how do i organize my holiday snaps?'. In *CIVR02*, page I:414 ff., 2002.

[82] Shaffalitzky F. and Zisserman A. Automated scene matching in movies. In *CIVR02*, pages 186-197, 2002.

[83] Shaffalitzky F. and Zisserman A. Automated location matching in movies. *CVIU*, 92(2-3):236-264, November 2003.

[84] Schiele B. and Crowley J.J. Object recognition using multidimensional receptive field histograms. In *ECCV96*, pages I:610-619, 1996.

[85] Schiele B. and Crowley J.J. Probabilistic object recognition using multidimensional receptive field histograms. In *ICPR96*, 1996.

[86] Schiele B. and Crowley J.J. Recognition without correspondence using multidimensional receptive field histograms. *International Journal on Computer Vision*, 36(1):31-50, January 2000.

[87] Schmid C. and Mohr R. Local grayvalue invariants for image retrieval. *PAMI*, 19(5):530-535, May 1997.

[88] Schmid C. and Mohr R. Combining grey value invariants with local constraints for object recognition. In *CVPR96*, pages 872-877, 1996.

[89] Schmid C. Constructing models for content-based image retrieval. In *CVPR01*, pages II:39-45, 2001.

[90] Schmid C. and Mohr R. Image retrieval using local characterization. In *ICIP96*, page 18A1, 1996.

[91] Shapiro L. G and Haralick R.M, Structural description and inexact matching, *IEEE Trans. Pattern Anal. Mach. Intell.* 3, 504-519, 1981.

[92] Shapiro L. G and Haralick R.M, A metric for comparing relational descriptions, *IEEE Trans. Pattern Anal. Mach. Intell.* 7 (1), 90-94, 1985.

[93] Sivic J. and Zisserman A. Video google: A text retrieval approach to object matching in videos. In *ICCV03*, pages 1470-1477, 2003.

[94] Sinkhorn R. A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *Ann. Math. Statictics*, vol. 35, p.p 876-879, 1964.

[95] Skocaj D. and Leonardis A. Weighted and robust incremental method for subspace learning. In *ICCV03*, pages 1494-1501, 2003.

[96] Skocaj D., Bischof H. and Leonardis A. A robust pca algorithm for building representations from panoramic images. In *ECCV02, page IV:761 ff., 2002.*

[97] Stein F. and Medioni G. Structural indexing: efficient 2D object recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 1192-1204, 1992.

[98] Swain M.J. and Ballard D.H. Indexing via color histograms. In *Ph. D.*, 1990.

[99] Swain M.J. and Ballard D.H. Color indexing. *International Journal on Computer Vision*, 7(1):11-32, November 1991.

[100] Swets D. and Weng J. Using discriminant eigenfeatures for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):831-836, 1996.

[101] Tarr Michael J. and Bülthoff Heinrich H. Is Human Object Recognition Better Described By Geon-Structural-Descriptions Or By Multiple-Views? *Journal of Experimental Psychology: Human Perception and Performance*, 21(6), 1494-1505, 1995.

[102] Turina A., Tuytelaars T., Moons T. and Van Gool L. Grouping via the matching of repeated patterns. In ÎCAPR01, pages 250-259, 2001.

[103] Turk M. and Pentland A. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71-86, 1991.

[104] Tuytelaars T., Van Gool L., D'haene L. and Koch R. Matching of affinely invariant regions for visual servoing. In *International Conference on Robotics and Automation*, pages 1601-1606, 1999.

[105] Tuytelaars T., Turina A. and Van Gool L. Noncombinatorial detection of regular repetitions under perspective skew. *PAMI*, 25(4):418-432, April 2003.

[106] Tuytelaars T. and Van Gool L. Wide baseline stereo matching based on local, affinely invariant regions. In *BMVC00*, 2000.

[107] Tuytelaars T. Local Invariant Features for Registration and Recognition. PhD thesis, University of Leuven, ESATPSI, 2000.

[108] Tuytelaars T. and Van Gool L. Content-based image retrieval based on local affinely invariant regions. In *Visual Information and Information Systems*, pages 493-500, 1999.

[109] Ullman Simon. High-Level Vision, Object Recognition and Visual Cognition. The MIT Press, Cambridge Massachusetts, 1996.

[110] Vincent E. and Laganière R., Matching Feature Points in Stereo Pairs: A Comparative Study of Some Matching Strategies, *in Machine Graphics & Vision*, vol. 10, no. 3, pp. 237-259, 2001.

[111] Wang Y. -K, Fan K. -C, Horng J. -T. Genetic-based search for error-correcting graph isomorphism. *IEEE Trans. on Systems, Man and Cybernetics*, 27:588-597, May 1997.

[112] Weiss I. Geometric invariants and object recognition. *International Journal on Computer Vision*, 10(3):207-231, June 1993.

[113] Willshaw D.J., Buneman O.P. and Longuet-Higgins H.C. Non-holographic associative memory. *Nature*, 222, 960-962, 1969.

[114] Wilson R.C and Handcock E.R.,Structural matching by discrete relaxation, *IEEE Trans. Pattern Anal. Mach. Intell.* 19, 634-648, 1997.

[115] Wilson R.C, Cross A.D.J and Handcock E.R., Structural matching with active triangulation, *Computer Vision Image Understanding*, 72, 21-38, 1998.

[116] Witkin, A. P. Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, pp. 1019-1022, 1983.

[117] Witkin, A. P. Scale Space Filtering: A New Approach to MultiScale Description. In *Image Understanding*, S. Ullman and W. Richards, eds., Norwood, N.J.: Ablex, 1984.

[118] Wong A.K.C and You M. Entropy and distance of random graphs with application to structural pattern recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* 7, 599-609, 1985.

[119] Wood Jeffrey. Invariant Pattern Recognition: A Review. *Pattern Recognition*, 29 (1), pp. 1-17, 1996.

[120] Xu L. and Oja E. Improved simulated annealing, Boltzmann machine, and attributed graph matching. In L. Almeida, editor, *Lecture Notes in Computer Science 412*, pp 151-161, Springer Velarg, 1990.