Identificación automática de diagramas cinemáticos con visión artificial

Gabriel Fontenla Carrera[1], Ángel Manuel Fernández Vilán[2], Pablo Izquierdo Belmonte[3]

[1]Grupo CIMA, área de ingeniería mecánica, Universidade de Vigo, España. Email: gabriel.fontenla@uvigo.es
[2] Grupo CIMA, área de ingeniería mecánica, Universidade de Vigo, España. Email: avilan@uvigo.es
[3] Grupo CIMA, área de ingeniería mecánica, Universidade de Vigo, España. Email: pabloizquierdob@uvigo.es

Resumen

En este trabajo se presenta un algoritmo de visión artificial para la detección y reconocimiento de diagramas cinemáticos en 2D a partir de imágenes procedentes tanto de fotografías de esquemas en papel como de archivos digitales. El algoritmo, que utiliza herramientas de la biblioteca libre de visión artificial OpenCV, permite el reconocimiento de dibujos hechos a mano alzada. Los mecanismos pueden incluir juntas de rotación, de traslación y rototraslación, bastidores y uniones rígidas entre barras. Sus principales aplicaciones están enfocadas en el mundo de la enseñanza, la comunicación y del diseño de nuevos sistemas mecánicos, posibilitando la realización de bocetos rápidos y que, tras ser procesados, estos se puedan utilizar para cálculos cinemáticos y simulaciones. Además, se almacenarán los píxeles de cada elemento por separado, lo que permite la animación de los propios dibujos.

Palabras clave: visión artificial; OpenCV; diagramas cinemáticos; mecanismos; reconocimiento de objetos.

Abstract

In this work, a computer vision algorithm for the detection and recognition of 2D kinematic diagrams created from images both paper schemes and digital files. The algorithm, which uses tools from the free computer vision library OpenCV, allows the recognition of handmade drawings. The mechanisms can include revolute, fixed, prismatic, cylindrical and rigid joints connected by bars. Its main applications are focused on the teaching world, the communication and the design of new mechanical systems, enabling the realization of quick sketches and, after being processed, these can be used for kinematic computations and simulations. Besides, the pixels of each element will be stored separately, which allows the animation of the drawings themselves.

Keywords: computer vision; OpenCV; kinematic diagrams; mechanisms; object recognition

1. Introducción

El diseño de mecanismos es un problema de gran interés en los ámbitos de la mecánica y de la robótica. Sin embargo, cada vez estos mecanismos son más complejos. Es por ello que los diagramas o esquemas cinemáticos son fundamentales para representar elementos y diseños reales de manera simplificada. Pese a todo, analizar y visualizar su cinemática sobre los diagramas puede ser complicado y los *softwares* especializados para simulación requieren un tiempo para familiarizarse con ellos, plantear el problema y realizar los cálculos propios [1] [2]. Es por ello que los dibujos hechos a mano presentan ventajas en cuanto a eficiencia, visualización y comunicación de nuevas ideas en el diseño mecánico o en la enseñanza [3]. En este punto sería realmente interesante que las herramientas de diseño y simulación virtuales pudieran interactuar con estos esquemas, a fin de optimizar el proceso de obtención de los modelos finales [3].

Por otro lado, la visión artificial cada vez está más presente en la sociedad. Las técnicas de visión artificial permiten automatizar la detección de distintos tipos de

formas y patrones, teniendo múltiples aplicaciones tecnológicas e industriales en campos tan variados como el militar, la automoción, la agricultura, la medicina, el deporte, o en líneas de producción y calidad, entre muchas otras. [4] [5] [6] [7].

Con el fin de acoplar los dibujos realizados a mano alzada, tanto en papel como en una pantalla, con las herramientas de diseño y simulación virtuales, en este trabajo se presenta un algoritmo que, mediante herramientas de la librería libre OpenCV, reconoce diagramas cinemáticos a través de un archivo digital de imagen (tipo JPG o PNG) y obtiene los componentes necesarios para la simulación/animación a posteriori.

Actualmente, el estado del arte encontrado relacionado con estas aplicaciones es escaso. En [1] y [2] proponen y prueban el reconocimiento de diagramas cinemáticos formados por barras y juntas de rotación mediante métodos de visión artificial para reconocimiento de objetos junto con algoritmos de optimización multiobjetivo evolutivos tipo NSGA-II. Por otra parte, en [3] y [8], se aplican redes neuronales convolucionales (CNN) para el reconocimiento de símbolos y diagramas de ingeniería realizados a mano alzada. El algoritmo presentado en este trabajo, en cambio, destaca por su sencillez. Se fundamenta en la búsqueda de rectángulos, circunferencias y segmentos para, posteriormente, mediante un post-procesado que permite asociar estas figuras con los elementos que componen el diagrama cinemático, distinguir entre barras rígidas (líneas rectas), uniones rígidas (convergencia de, al menos, dos barras), bastidores (tres líneas cortas paralelas y próximas), juntas de rotación (circunferencias), juntas de traslación (rectángulos) y juntas de rototraslación (rectángulos con una circunferencia en su interior) (ver **Figura 1**). Para ello, se recurre a técnicas de procesado y visión artificial, operaciones de morfología matemática y geometría básica.

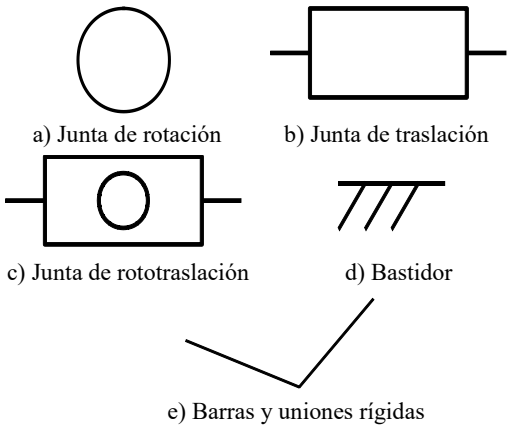


Figura 1. Simbología permisible en los mecanismos

Entre esta simbología no se incluyen elementos importantes como podrían ser juntas de rotación en medio de barras, figuras rígidas con diferentes formas a las barras o las juntas de rotación con bastidor. Para este último caso, como alternativa, se propone la formulación presentada en la **Figura 2**. Así se evita utilizar triángulos en la simbología, lo que facilita el reconocimiento y clasificación de los elementos. Además, permite la realización de inversiones cinemáticas de manera rápida (ver **Figura 3**). En futuros trabajos se contemplará la incorporación de nueva simbología para analizar mecanismos más complejos.

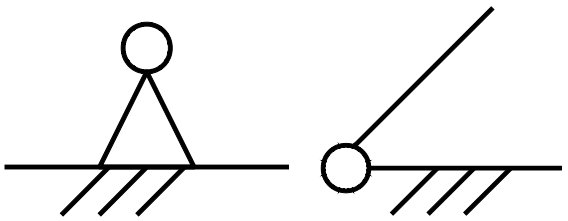


Figura 2. Junta de rotación con bastidor (a) y alternativa propuesta (b)

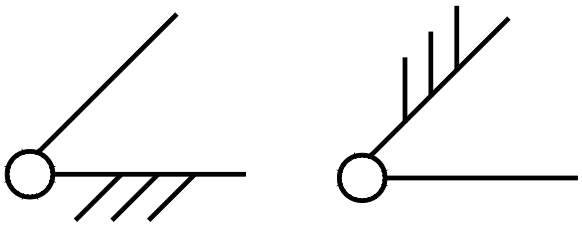


Figura 3. Ejemplo de inversión cinemática

Por último, los píxeles de cada elemento del diagrama son almacenados por separado, por lo que además de poder generar un diagrama virtual que represente el original para el cálculo de la cinemática, también se podría resolver la cinemática sobre los elementos del propio dibujo, pudiendo crear animaciones.

2. Metodología

En este apartado se explicará el método propuesto para resolver el problema planteado. En la **Figura 4** se muestra un diagrama de bloques en el que se simplifica de manera esquemática el funcionamiento del algoritmo creado.

2.1. Preprocesado de la imagen y breve explicación de la metodología aplicada.

El primer paso es convertir la imagen a color RGB original a escala de grises y consecutivamente, empleando el método de Otsu para establecer automáticamente el valor de umbral adecuado [9] [10], se obtiene la imagen en blanco y negro. Además, en caso de que tras la conversión sea el fondo blanco y el

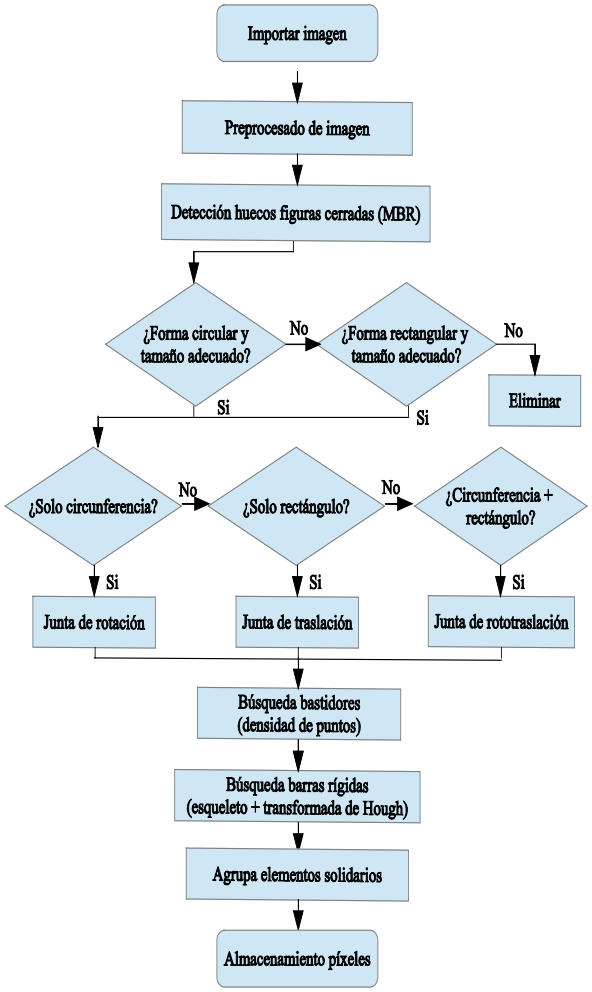


Figura 4. Diagrama de flujo del algoritmo propuesto

trazo negro, se debe invertir la imagen binaria para mantener la consistencia con la literatura [11], donde se consideran los píxeles activos como el trazo del dibujo y los ceros como fondo. Si la imagen lo requiere, se pueden aplicar filtros como el de la mediana, el suavizado Gaussiano u operadores morfológicos de erosión para limpiar el ruido y de dilatación para mejorar ciertas cualidades de la imagen, con el fin de conseguir una mayor probabilidad de detección de los elementos [10] [12]. Como se explicará en detalle en el apartado **2.2**, el reconocimiento de las figuras geométricas se realizará localizando los MBR (*Minimum Bounding Rectangle*), rectángulos de mínima área capaces de inscribir una figura en su interior. Es por ello que los contornos deben ser completamente cerrados, por lo que lo más habitual será introducir, al menos, una dilatación inicial que incremente el grosor del trazo del dibujo y cierre posibles aberturas, evitando así problemas de detección.

Con el fin de realizar un primer filtrado de las figuras del dibujo de manera sencilla se dedujo, en base a la experiencia tras varias pruebas, una relación entre la resolución de la imagen, el grosor del trazo de dibujo y el tamaño de las figuras. Como regla general, se establece que para imágenes con bajo número de píxeles (resoluciones bajas), los trazos no pueden ser demasiado gruesos y las figuras geométricas serán pequeñas. Por el contrario, a medida que se aumenta la resolución de la imagen, los símbolos presentan mayor libertad de tamaño, al poder dibujarse desde figuras geométricas más grandes, tanto con trazos gruesos como finos, hasta otras pequeñas con de trazo fino.

En la gráfica mostrada en la **Figura 5** se expresan las variaciones de los límites de tamaño que se utilizarán a lo largo del trabajo para filtrar los elementos del esquema en función del grosor medio del trazo (t) partiendo de distintos valores de r, siendo r la relación entre la resolución de la imagen dividido entre una constante igual a la resolución de una imagen de alta definición (1280x720px) (1). A estos límites se les deberá aplicar un *offset* determinado experimentalmente para ajustar estos valores según el tipo de figura geométrica que se busque (ver ecuaciones (2) a (6)).

$$r = \frac{N_{píxeles\ imagen}}{N_{píxeles\ HD}} = \frac{N_{píxeles\ imagen}}{921\,600} \qquad (1)$$

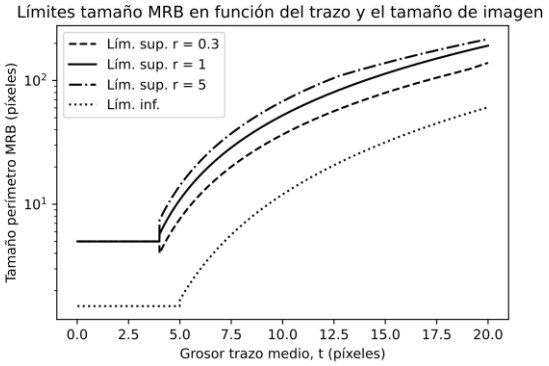


Figura 5. Límites de tamaño en función del trazo (t) y tamaño de la imagen (r)

Es importante tener en cuenta que para el caso del límite de tamaño inferior, este se considera que solo varía en función del grosor de trazo, ya que como se ha explicado con anterioridad, aumentando la resolución se seguirán pudiendo dibujar símbolos pequeños si el trazo es lo suficientemente fino.

Una vez localizados los MBR, se procederá al análisis de los contornos a los que pertenecen, diferenciando según su tamaño y forma entre juntas de rotación, traslación y rototraslación. Posteriormente, se detectarán los bastidores por densidad de puntos para después, mediante la transformada de Hough, obtener las rectas y sus intersecciones, que representan las barras y uniones rígidas, respectivamente. Por último, el algoritmo deduce qué elementos son solidarios al

mismo conjunto de barras y extrae los píxeles de cada símbolo por separado.

2.1.1. Dilataciones: tamaño de *kernel*

Para mejorar la capacidad de detección del algoritmo, se aplican cuatro dilataciones independientes a lo largo de su ejecución: una dilatación previa, que permite al algoritmo localizar correctamente los MBR en caso de existir figuras con contornos mal cerrados; una dilatación para mejorar la calidad de las circunferencias (al permitir trazos dibujados a mano, existen imperfecciones en los contornos); otra análoga para los rectángulos; y una última que permita incrementar la densidad de puntos de los bastidores para su correcta detección.

Para conseguir las dilataciones de los trazos del dibujo, se aplicaron *kernels* (también conocidos como matrices de convolución o núcleos) con elementos estructurales rectangulares, es decir, con todos los elementos de la matriz que lo conforman iguales a la unidad (activos) [13]. El *kernel* se posiciona de manera centrada sobre cada uno de los píxeles de la imagen, formando un vecindario con los píxeles que quedan definidos dentro de los valores activos del propio *kernel*. En caso de que un píxel perteneciente al vecindario sea igual a 1, automáticamente el píxel central pasará a valor 1, aumentando de esta manera el grosor del trazo [14]. Los *kernels* tendrán dimensiones nxn, siendo n un número entero, positivo e impar con el fin de poder mantener siempre un elemento central. Cuanto más grande sea el valor de n, mayor será la dilatación formada al incluir un vecindario más numeroso, por lo que será útil para cerrar aberturas más grandes, detectar figuras peor dibujadas o aumentar la densidad de puntos pero, como contrapartida, aumentará la posibilidad de obtener agrupamientos densos de píxeles que el algoritmo no pueda descifrar.

Los distintos valores que puedan tomar las dimensiones de cada *kernel* serán los únicos datos, junto con la imagen, que deba introducir el usuario manualmente.

2.2. Detección de figuras cerradas: reconocimiento de circunferencias y rectángulos

La detección de objetos es una técnica de gran madurez e importancia en el ámbito de la visión artificial [1] [2]. En este caso, para la detección de las figuras cerradas se utiliza el MBR. Esta es una estrategia común para almacenar aproximaciones de objetos que puedan cumplir determinadas características que se analizarán en pasos futuros [15] [16].

En este trabajo se optó por extraer los MBR de los contornos interiores de los símbolos, en lugar de englobar los propios elementos. Esto se debe a que por la parte externa se encuentran todas las barras rígidas unidas a las figuras y, por consiguiente, los MBR englobarán más de un único símbolo, perdiendo su utilidad. Una vez localizados estos contornos, para evitar problemas de símbolos partidos como los presentados en la **Figura 6**, provocados por las barras rígidas que actúan como guías y atraviesan las juntas de traslación y rototraslación dividiéndolos en varias secciones, se comprueba la existencia de contornos contiguos. En caso de producirse esta situación, se fusionan en un único rectángulo que englobe al conjunto de MBR cercanos.

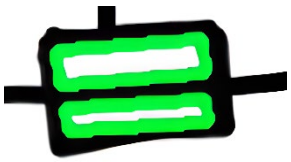


Figura 6. Problema MBR próximos

A continuación, se procede a la clasificación de las circunferencias. Partiendo de los MBR, se aplica el filtro mostrado en la **Figura 5** con un *offset* determinado experimentalmente mediante ensayos, obteniendo el valor de lsc(r,t) y lic(r,t) (ver (2) y (3)), siendo estas variables equivalentes al perímetro máximo y mínimo en píxeles, respectivamente, admisibles de la circunferencia.

$$lsc(r,t) = \lim\_\sup(r,t) + 120 \quad (2)$$

$$lic(r,t) = \lim\_\inf(t) + 5 \quad (3)$$

Una vez obtenidos los posibles candidatos a circunferencias, se intentan recorrer los píxeles de la figura que engloba a cada MBR siguiendo la ecuación de la circunferencia y, en caso de lograrlo, se considerará como una circunferencia real. Para facilitar este procedimiento, debido a las imperfecciones del dibujo, se barren circunferencias con distintos radios a la vez que se aplica una dilatación previa para ensanchar las líneas de la figura y que el algoritmo tenga la capacidad de encontrar la solución con mayor facilidad.

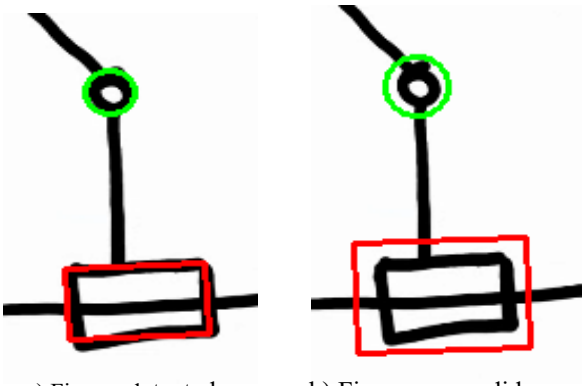
El siguiente paso es el reconocimiento de los rectángulos, el cual es análogo al anterior. Primero, se aplica un filtro en el tamaño de los MBR. En este caso lsr se corresponde con la longitud máxima en píxeles que puede tener el lado largo del rectángulo y lir es el límite inferior del lado corto del rectángulo (ver (4) y (5)). Consecutivamente se aplica una dilatación, que se puede ajustar manualmente, y finalmente se recorren los cuatro lados del MBR intentando seguir únicamente píxeles correspondientes a la figura y, si se logra, esta se clasificará como un rectángulo. En este caso, debido a que con un trazo a mano alzada es complicado realizar líneas lo suficientemente rectas, se estableció un umbral del 60%. Si siguiendo las cuatro rectas que

forman el rectángulo, al menos un 60% de los píxeles son parte de una figura, entonces se aceptará como rectangular. Este valor se determinó empíricamente durante la realización de las pruebas, siendo el que mejor resultados proporcionaba, aunque se entiende que existirán particulares donde este umbral deba ser cambiado.

$$lsr(r,t) = \text{lim\_sup}(r,t) + 250 \quad (4)$$

$$lic(t) = \text{lim\_inf}(t) + 30 \quad (5)$$

Además, tanto en las circunferencias como en los rectángulos detectados, se les añade un *offset* de un 15% en las dimensiones para asegurarse en futuros pasos de que todos los píxeles de los símbolos quedan englobados en área detectada (ver **Figura 7**).



a) Figuras detectadas b) Figuras expandidas

Figura 7. En a) se muestran las figuras cerradas detectadas: no todos los píxeles quedan dentro. En b) se muestran las figuras detectadas con un *offset* predefinido

2.3. Primera clasificación: juntas de rotación, de traslación y rototraslación

Una vez se han encontrado las circunferencias y los rectángulos, se procede a una primera clasificación de objetos muy sencilla.

- Si se localiza una circunferencia, se considera una junta de rotación.
- Si se localiza un rectángulo, se considera una junta de traslación.
- Si una circunferencia se encuentra en el interior de un rectángulo, el algoritmo la determinará como una junta de rototraslación.

A su vez, también se determinan los puntos de corte de estos elementos con las barras rígidas que llegan a ellos, lo cual será útil para conocer posteriormente qué elementos están solidarios por el mismo conjunto de barras.

2.4. Detección y reconocimiento de bastidores

Antes de iniciar el reconocimiento de los bastidores, con el fin de mejorar las probabilidades de detectarlos y ayudar al algoritmo a ser más eficiente, se suprimen los símbolos ya localizados del dibujo.
El método de la ventana deslizante es ampliamente utilizado en técnicas de reconocimiento de objetos como en [1] [17]. Para la búsqueda de bastidores el algoritmo crea una ventana deslizante con las dimensiones obtenidas aplicando la ecuación (6). Esta ventana deslizante recorre la imagen de arriba abajo y de izquierda a derecha y calcula el porcentaje de píxeles dibujados con respecto a los totales dentro de cada ventana. Como los bastidores son representados por tres líneas rectas muy próximas, la densidad de píxeles marcados en estas zonas será mucho mayor que la de las líneas rectas que forman las barras rígidas. Por lo tanto, todas aquellas subregiones que pasen un umbral mínimo fijado en un tercio del total de píxeles contenidos en el interior de la ventana, serán consideradas bastidores.

$$dim(t) = \text{lim\_inf}(t) + 50 \quad (6)$$

2.5. Barras, guías y uniones rígidas

En este punto ya estarían identificados todos los símbolos, quedando solamente las líneas rectas que representan las barras y guías del mecanismo sin reconocer.

Desde los años 60, la forma principal que existe para la detección de líneas rectas es la transformada de Hough (HT) y sus posteriores variantes. Esta técnica se basa en la dualidad punto-línea. Por un mismo punto pueden pasar infinitas rectas con diferente pendiente cada una. Si se tiene un conjunto de puntos colineales y, en cada uno de ellos, se generan conjuntos de rectas con distintas inclinaciones, existirá una única recta que pase a través de todo el conjunto de puntos [18] [19]. Para evitar el problema de números infinitos o muy elevados cuando las rectas son verticales o próximas a esta situación, en lugar de utilizar la típica formulación pendiente – intersección, se suele trabajar con la forma "normal" (ρ,θ) [19]:

$$\rho = x \sin\theta + y \cos\theta \quad (7)$$

siendo ρ es la distancia del segmento entre el origen de coordenadas hasta el punto más próximo de la línea recta y θ es el ángulo que forma ese segmento con el eje de abscisas.

De esta manera, cada conjunto de líneas que pasa por cada punto formarán una función sinusoidal en el espacio de parámetros (θ,ρ) y, si existiese colinealidad entre diferentes puntos, estas funciones intersecarán todas en el mismo lugar. Sin embargo, el proceso de

"votación" a la hora de acumular los valores de las funciones sinusoidales para posteriormente buscar los máximos locales (correspondientes a los puntos donde estas se cruzan), es un proceso que consume mucho tiempo y memoria [19] [20].

Es por ello que en este algoritmo se utilizó una versión más moderna de la transformada de Hough, en concreto la transformada de Hough probabilística progresiva (PPHT), la cual está implementada en el módulo OpenCV [21]. En la transformada de Hough probabilística original (PHT), se considera que con una cantidad pequeña de puntos seleccionados al azar (del orden incluso del 2% de los totales para algunos casos), son suficientes para determinar si el elemento del dibujo es una línea recta o no, ahorrando un tiempo de ejecución considerable a cambio de una leve pérdida en la calidad [9] [18].

A diferencia de la PHT, como se explica en [22], la PPHT no ejecuta la resolución de la HT estándar a partir de un pequeño subconjunto de datos preseleccionados, sino que reduce los cálculos necesarios explotando la diferencia en la fracción de votos necesarios para detectar líneas fiablemente. Es decir, la fracción de puntos necesaria para realizar el proceso de votación no está predefinida, sino que se adapta a cada caso particular según la complejidad de los datos. De esta manera, en [22] se afirma que "el número de votos (velocidad) es casi independiente de la longitud de la línea (puntos de entrada)".

Debido a que los trazos de las líneas no son de un único píxel, si se aplicara directamente la transformada de Hough, el resultado sería el de múltiples líneas superpuestas y partidas [6] [23] ya que existe información redundante y los trazos no son completamente rectos. Con el fin de eliminar el primer problema del exceso de información, se aplica una esqueletonización previa a la detección de las líneas. El esqueleto es definido en [24] como "un conjunto de líneas mediales entre las extremidades de las figuras. … La idea básica del esqueleto es eliminar información redundante mientras se retiene solo la información topológica relativa a la forma y estructura del objeto que pueda ayudar con el reconocimiento."

Para resolver el segundo inconveniente, después de aplicar la PPHT, se agruparon las rectas obtenidas según su pendiente y proximidad de puntos finales e iniciales, método utilizado también en [6] y [23], extrayendo así las barras rígidas y sus puntos de unión, siendo estos considerados cuando se aprecia un cambio brusco de pendiente entre dos barras contiguas.

En este trabajo, se considera la diferenciación entre barras rígidas y guías (aquellas barras por las que se pueden desplazar las juntas de traslación y rototraslación). Siguiendo esta definición, la separación es sencilla: si una barra contiene una junta de traslación o rototraslación con una inclinación similar, el algoritmo la considerará como guía.

Por último, una vez que se tienen todos los elementos definidos y sus uniones, el algoritmo es capaz de determinar qué elementos del mecanismo están solidarios a un mismo conjunto de barras rígidas teniendo en cuenta las posiciones en las que se encuentran y los puntos de corte calculados con anterioridad, lo cual resulta útil para facilitar posibles simulaciones y animaciones.

2.6. Extracción de píxeles

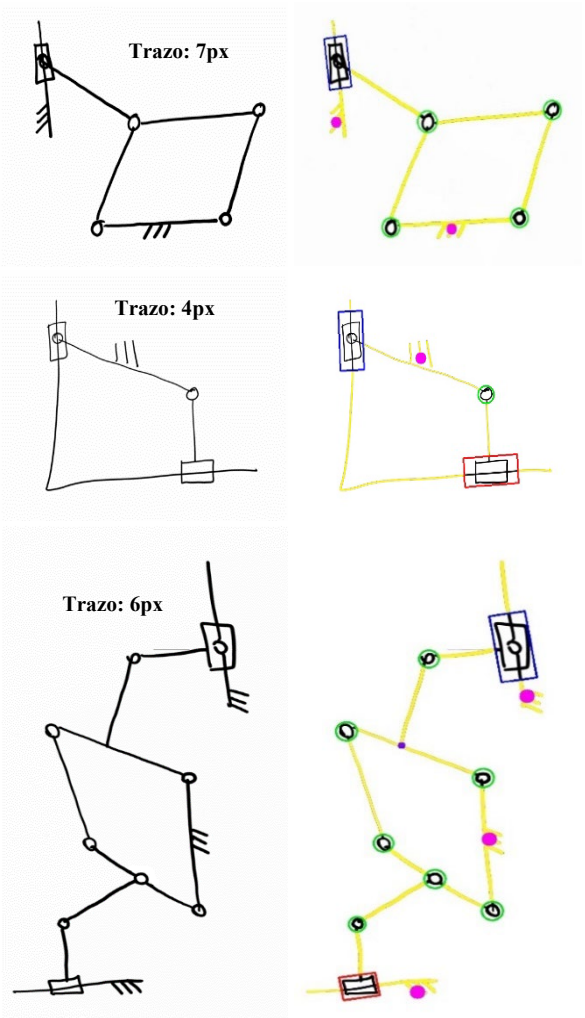
Después de detectar todos los objetos, la extracción de píxeles es sencilla: elemento a elemento, estos se aíslan en un ROI (región de interés) en la cual todos los píxeles activos pertenecerán a dicha figura, por lo que la única operación necesaria será el almacenamiento de las posiciones de dichos píxeles con respecto a la imagen global. Sin embargo, se detectó una excepción, producida en la parte interior de los símbolos de las juntas de traslación y rototraslación. En este caso, si se resolvieran las ecuaciones cinemáticas y se quisiera simular el movimiento, la junta de traslación y rototraslación al desplazarse y trasladar todos los píxeles por los que está formado, dejaría un espacio en blanco en la guía. Para evitar este problema se propusieron dos soluciones: la primera, si en el esquema aparecen los puntos de la guía en el interior de las figuras, se separan mediante un algoritmo de seguimiento de píxeles, típicamente utilizados para la extracción de contornos [25]. Como segunda solución, en caso de que no se hubieran dibujado los píxeles de la guía en el tramo interior a estos símbolos, se crearía una línea virtual que uniese los puntos de corte de la figura con la propia guía con el grosor del trazo medio del dibujo.

3. Resultados

Los resultados se adquieren de una serie de pruebas realizadas sobre un conjunto de dibujos hechos tanto en una pizarra digital en una *tablet* como en papel a través de fotografías. Estos dibujos fueron elaborados por distintos investigadores y se utilizaron diferentes grosores y colores en el trazo. A continuación, se muestran una serie de diagramas cinemáticos dibujados a mano alzada (izquierda) con los resultados obtenidos (derecha) (ver **Figura 8** a **Figura 11**) junto con las tablas (ver **Tabla 1** a **Tabla 4**) donde se recopilan los valores de los tamaños de *kernel* utilizados en cada caso (ver **2.1.1**).

Además, aunque existan infinitos valores y combinaciones para ajustarlos, en ningún caso resultó necesario filtrar las imágenes con un tamaño de *kernel* superior a 9. Asimismo, pese a que los valores de las

tablas son muy variables, conociendo el orden de ejecución del algoritmo, el proceso de calibración de estos es muy intuitivo. Si en la imagen original los símbolos formados por los rectángulos y circunferencias presentan contornos completamente cerrados, el valor del *kernel* para la dilatación previa puede dejarse en un valor bajo. Seguidamente, se observará si las circunferencias son localizadas correctamente. En caso de fallar su reconocimiento, el problema se solucionará aumentando el valor del *kernel* perteneciente a la dilatación circunferencia. Sucesivamente, se comprueban los rectángulos y los bastidores, siempre en ese orden, siguiendo un procedimiento análogo al de las circunferencias.

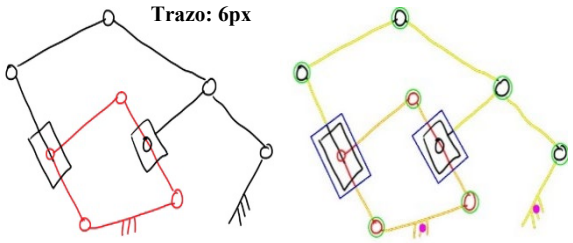


Dibujo diagrama Reconocimiento diagrama

Figura 8. Pruebas pizarra digital (r ≈ 0.3)

Tabla 1. Valores tamaño *kernel* Figura 8 (*nxn*)

DILATACIÓN	TAMAÑO *KERNEL*
Previa	3x3
Circunferencias	5x5
Rectángulos	3x3
Bastidores	3x3

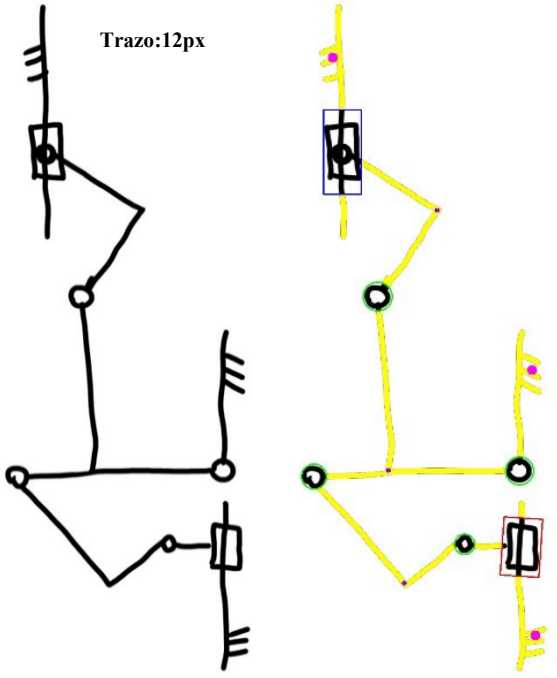


Dibujo diagrama Reconocimiento diagrama

Figura 9. Pruebas pizarra digital (r ≈ 0.7)

Tabla 2. Valores tamaño *kernel* Figura 9 (*nxn*)

DILATACIÓN	TAMAÑO *KERNEL*
Previa	3x3
Circunferencias	7x7
Rectángulos	5x5
Bastidores	5x5

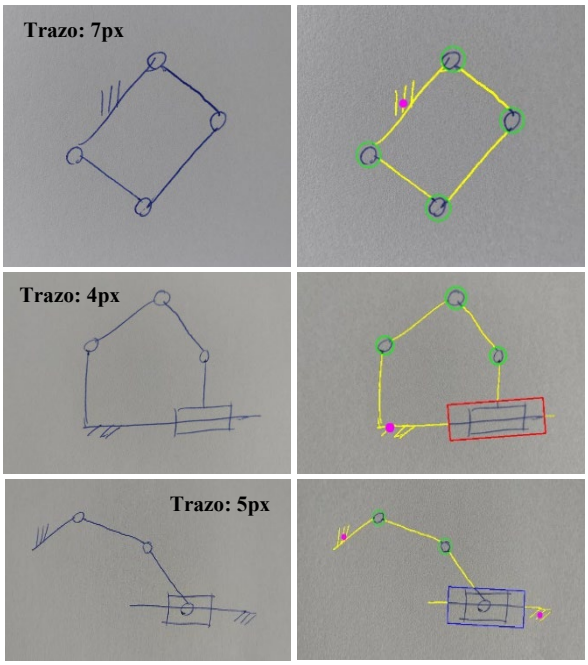


Dibujo diagrama Reconocimiento diagrama

Figura 10. Pruebas pizarra digital (r ≈ 1)

Tabla 3. Valores tamaño *kernel* Figura 10 (*nxn*)

DILATACIÓN	TAMAÑO *KERNEL*
Previa	1x1
Circunferencias	3x3
Rectángulos	3x3
Bastidores	1x1



Dibujo diagrama Reconocimiento diagrama

Figura 11. Pruebas papel (r < 1)

Tabla 4. Valores tamaño *kernel* Figura 11 (*nxn*)

DILATACIÓN	TAMAÑO *KERNEL*
Previa	7x7
Circunferencias	7x7
Rectángulos	9x9
Bastidores	5x5

3.1. Limitaciones

Debido a la sencillez y novedad del algoritmo, existen algunas limitaciones que futuras investigaciones permitirán resolver.

- No se permite el uso de simbología o elementos mecánicos no contemplados en este artículo. Como se muestra en la **Figura 12** marcado con una flecha roja, el algoritmo puede confundir con facilidad símbolos no implementados. En este caso, una junta de rotación en medio de las barras que indicaría que B0.0 y B0.1 son en realidad una única barra en la que se articula una segunda barra B1, es interpretada en manera errónea y el algoritmo considerará que son, en realidad, tres barras independientes.

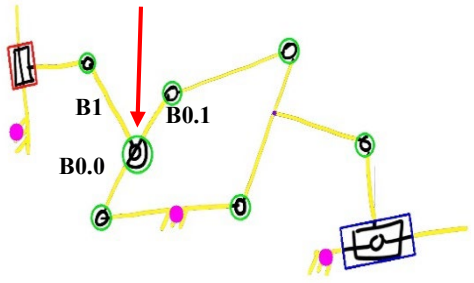


Figura 12. Fallo simbología. Reconocimiento erróneo de una junta de rotación en medio de barras (flecha roja) detectada como una junta de rotación entre tres barras

- Se deben evitar barras cruzadas. En este caso, el algoritmo detectará correctamente todos los elementos del mecanismo. Sin embargo, en el momento de relacionar las conexiones entre ellos, fallará. En el caso que se muestra en la **Figura 13**, la barra B0 debería conectar únicamente las juntas de rotación A1 y A3 y, por otro lado, la barra B1, las juntas de rotación A0 y A2. Sin embargo, en el resultado final propuesto por el algoritmo, las cuatro juntas de rotación aparecerán unidas por un conjunto de cuatro barras soldadas con una unión rígida en el centro.

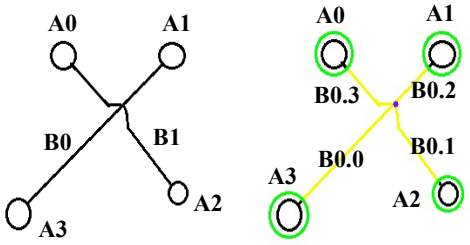


Diagrama en cruz Fallo reconocimiento diagrama

Figura 13. Reconocimiento erróneo de dos barras que se cruzan. El algoritmo detecta una unión rígida.

- En el caso de imágenes de muy alta resolución (r >= 2), pueden aparecer problemas para identificar elementos. Esto es provocado porque la función definida en el apartado **2.1** pierde precisión al aumentar r y, como en general no se han contemplado imágenes de resoluciones tan elevadas, no se consideró un problema importante en este trabajo inicial. Además, existe una solución muy sencilla que a su vez permite reducir el número de cálculos y aumentar la velocidad de obtención de los resultados, consistente en reducir la resolución de la imagen antes de ejecutar el algoritmo. Como ejemplo particular a fin de ilustrar este problema, en la **Figura 14** se muestra el mismo dibujo que en la **Figura 11**, con la única diferencia de que en este caso r es igual a 2 (valor de muy alta resolución). En esta nueva imagen, se aprecia cómo el programa confunde dos juntas de rotación con las de traslación. Esto se debe a que los límites

superiores e inferiores de los símbolos quedan mal definidos y los tamaños de los elementos están fuera de los rangos permisibles. En este caso concreto, las circunferencias que componen las juntas de rotación superan el valor del límite superior admisible para estas figuras y, al aplicar un tamaño de *kernel* de dilatación elevado para la detección de los rectángulos manteniendo un coeficiente de 0.6 de similitud con respecto a un rectángulo perfecto, confunde las circunferencias con los paralelogramos.

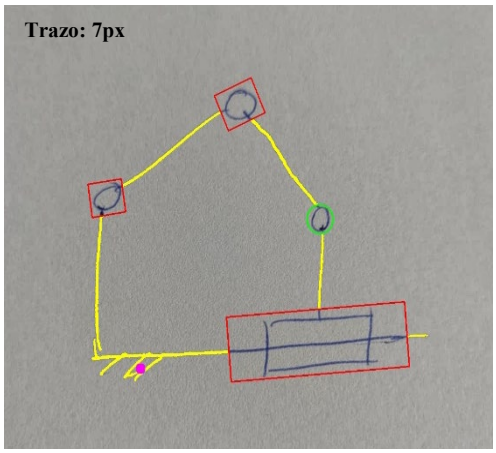


Figura 14. Fallo por elevada resolución. El tamaño de los símbolos queda fuera de rango. En este caso concreto, la junta de rotación superior y la izquierda son interpretadas como juntas de traslación.

4. Conclusiones

Este trabajo muestra cómo mediante técnicas de visión artificial es posible el reconocimiento en tiempo real de diagramas cinemáticos previamente dibujados a mano alzada, tanto a partir de dispositivos digitales como de fotografías de diseños en papel, presentando diversos campos de aplicación, como son la educación, la comunicación o el diseño de mecanismos.

El algoritmo es sencillo, rápido y podría ejecutarse en un ordenador o *tablet* sin consumir apenas recursos, convirtiéndose así en una herramienta prometedora. En futuras investigaciones se continuarán desarrollando sus capacidades, incluyendo nueva simbología con el fin de poder analizar mecanismos más complejos. Además, estos progresos deben de ir acompañados de una mayor robustez y fiabilidad ante casos de falsos positivos y negativos, mejorando la capacidad de reconocimiento, ya que al aumentar la simbología también aumentarán las probabilidades de que el algoritmo la confunda, tal y como se explica en el punto **3.1**. Por último, se espera que el algoritmo pueda analizar imágenes de muy alta resolución sin la necesidad de preprocesarlas, con el fin de aprovechar la mayor cantidad de información disponible. Para ello, se tratará de solucionar la imprecisión en el cálculo de los límites de tamaño superiores e inferiores relativos a los elementos que componen la simbología.

Aunque en esta investigación se trataron sólo casos donde los esquemas cinemáticos habían sido dibujados con anterioridad, en paralelo también se está trabajando en el desarrollo de un programa complementario que identifique los símbolos en tiempo real a medida que se crea el diagrama cinemático en una pizarra digital interactiva con el fin de integrar ambas posibilidades en una única aplicación. Si bien con esta segunda opción no será posible la entrada de dibujos en papel, permitirá el diseño de diagramas cinemáticos más complejos garantizando su correcta detección. Esto es posible porque se posee una mayor cantidad de información, al poder seguir en todo momento el trazo de los símbolos. Además, será más cómodo e intuitivo para el usuario, que podrá observar a medida que dibuja si los símbolos son detectados correctamente y no tendrá que esperar a finalizar el diseño para generar un archivo digital de imagen e introducirla en el programa.

5. Referencias

[1] M. Eicholtz, L. B. Kara y J. Lohn, «Recognizing Planar Kinematic Mechanisms from a Single Image Using Evolutionary Computation,» de *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014. DOI: 10.1145/2576768.2598354

[2] M. Eicholtz y L. Burak Kara, «Intermodal image-based recognition of planar kinematic mechanisms,» *Journal of Visual Languages & Computing,* vol. 27, pp. 38-48, 2015. DOI: https://doi.org/10.1016/j.jvlc.2014.10.024

[3] L. Fu y K. Levent, «Recognizing Network-Like Hand-Drawn Sketches: A Convolutional Neural Network Approach,» de *Proceedings of the ASME Design Engineering Technical Conference*, 2009. DOI: 10.1115/DETC2009-87402

[4] B. G. Batchelor, «1.3 Machine Vision Is Not Computer Vision,» de *Machine Vision for Industrial Applications*, Cardiff: Springer, 2012. ISBN: 978-1-84996-169-1

[5] S. Zehang, G. Bebis y R. Miller, «On-road vehicle detection: a review,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, nº 5, pp. 694-711, 2006. DOI: 10.1109/TPAMI.2006.104

[6] N. Samarasekara, Sports Analysis Using Video Tracking, 2015. DOI: 10.13140/RG.2.2.34195.78883

[7] H. Müller, N. Michoux, D. Bandon y A. Geissbuhler, «A review of content-based image retrieval systems in medical applications—clinical benefits and future directions,» *International Journal of Medical Informatics,* vol. 73, nº 1, pp. 1-23, 2004. DOI: https://doi.org/10.1016/j.ijmedinf.2003.11.024

[8] «From engineering diagrams to engineering models: Visual recognition and applications,» *Computer-Aided Design,* vol. 43, nº 3, pp. 278-292, 2011. DOI: https://doi.org/10.1016/j.cad.2010.12.011

[9] M. Zakaria, H. Choon y S. A. Suandi, «Object Shape Recognition in Image for Machine Vision Application,» *International Journal of Computer Theory and Engineering,* pp. 76-80, 2012. DOI: 10.7763/IJCTE.2012.V4.428

[10] E. R. Davies, «CHAPTER 4. Thresholding Techniques,» de *Computer and Machine Vision: Theory, Algorithms, Practicalities*, Fourth Edition ed., Elsevier, 2012. ISBN: 978-0-12-386908-1

[11] E. R. Davies, «CHAPTER 2. Images and Imaging Operations,» de *Computer and Machine Vision: Theory, Algorithms, Practicalities*, Fourth Edition ed., Elsevier, 2012. ISBN: 978-0-12-386908-1

[12] E. R. Davies, «CHAPTER 3. Basic Image Filtering Operations,» de *Computer and Machine Vision: Theory, Algorithms, Practicalities*, Fourth Edition ed., Elsevier, 2012. ISBN: 978-0-12-386908-1

[13] «Image Filtering,» OpenCV, [En línea]. Available: https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#gac342a1bb6eabf6f55c803b09268e36dc. [Último acceso: 28 Septiembre 2022].

[14] «Types of Morphological Operations,» MathWorks, [En línea]. Available: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html. [Último acceso: 28 Septiembre 2022].

[15] D. Papadias, Y. Theodoridis, T. Sellis y M. Egenhofer, «Topological Relations in the World of Minimum Bounding Rectangles: a Study with R-trees,» *ACM SIGMOD Record,* 1995. DOI: 10.1145/223784.223798

[16] A. Sleit y E. Al-Nsour, «Corner-Based Splitting: An Improved Node Splitting Algorithm for R-Tree,» *Journal of Information Science,* vol. 40, nº 2, p. 222–236, 2014. DOI: 10.1177/0165551513516709

[17] C. Wojek, S. Walk y B. Schiele, «Multi-Cue onboard pedestrian detection,» de *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009. DOI: 10.1109/CVPRW.2009.5206638

[18] N. Kiryati, Y. Eldar y A. M. Bruckstein, «A probabilistic Hough transform,» *Pattern Recognition,* vol. 24, nº 4, pp. 303-316, 1991. DOI: https://doi.org/10.1016/0031-3203(91)90073-E

[19] E. R. Davies, «CHAPTER 11 Line Detection,» de *Computer and Machine Vision: Theory, Algorithms, Practicalities*, Fourth Edition ed., Elsevier, 2012. ISBN: 978-0-12-386908-1

[20] H. Kälviäinen, P. Hirvonen, L. Xu y E. Oja, «Probabilistic and non-probabilistic Hough transforms: overview and comparisons,» *Image and Vision Computing,* vol. 13, nº 4, pp. 239-252, 1995. DOI: https://doi.org/10.1016/0262-8856(95)99713-B

[21] «Hough Line Transform,» OpenCV, [En línea]. Available: https://docs.opencv.org/4.x/d6/d10/tutorial_py_houghlines.html. [Último acceso: 2 Julio 2022].

[22] J. Matas, C. Galambos y J. Kittler, «Robust Detection of Lines Using the Progressive Probabilistic Hough Transform,» *Computer Vision and Image Understanding,* vol. 78, nº 1, pp. 119-137, 2000. DOI: https://doi.org/10.1006/cviu.1999.0831

[23] D. G. Lowe, «Three-Dimensional Object Recognition from Single Two-Dimensional Images,» *Artificial Intelligence,* vol. 31, nº 3, p. 355–395, 1987.

[24] E. R. Davies, «CHAPTER 9. Binary Shape Analysis,» de *Computer and Machine Vision: Theory, Algorithms, Practicalities*, Fourth Edition ed., Elsevier, 2012. ISBN: 978-0-12-386908-1

[25] J. Seo, S. Chae, J. Shim, D. Kim, C. Cheong y T.-D. Han, «Fast Contour-Tracing Algorithm Based on a Pixel-Following Method for Image Sensors,» *Sensors,* vol. 16, nº 3, 2016. DOI: 10.3390/s16030353