A Grasp Detection Method for Industrial Robots Using a Convolutional Neural Network

E. Ogas, L. Avila, G. Larregay, and D. Morán

***Abstract*—In the near future, most of the industrial robots will serve as assistants involved in targeted complex manufacturing tasks which are difficult to be automated. To achieve this, it is crucial to enhance the ability of manipulators to pick and place objects from the assembly line. Reorienting and picking up pieces for assembly are difficult tasks to be done by manipulators since, for different pieces, shapes and physical properties vary. In this work, we use Convolutional Neural Networks for recognizing a selected production piece on a cluster. Once the selected piece has been recognized, a grasping algorithm estimates the best gripper configuration so that the robot is able to pick the piece up. We tested our algorithm on grasping experiments with an ABB robot and using a common webcam as image input. We found that our implementations perform well and the robot was able to pick up a variety of objects.**

***Index Terms*—Industrial robot, Deep learning, Object grasping, Hough transform, Friction cones.**

I. Introducción

En un futuro cercano, la mayoría de los robots industriales servirán como asistentes humanos involucrados en tareas de fabricación complejas y específicas que son difíciles de automatizar por completo. En dicho contexto colaborativo, los manipuladores necesitarán la capacidad de seleccionar y recoger objetos de forma autónoma desde la línea de producción y colocarlos en una ubicación destino. Sin embargo, recoger objetos para ensamblar es una tarea difícil de automatizar, ya que la forma y las propiedades físicas varían de un objeto a otro. La complejidad radica en el hecho de que describir una escena de fabricación requiere numerosas y complejas tareas visuales, desde la detección y el reconocimiento de diferentes objetos en el plano, hasta la decisión de cómo deben manipularse. Dado que la posición de las piezas de producción es a priori desconocida para el robot, no es a menudo posible realizar una planificación de la tarea de agarre [1].

Recoger un objeto implica dos tareas principales: detectar y reconocer el objeto entre otros y decidir cuales son los mejores puntos para su agarre. En el campo de la visión artificial, la percepción implica construir un modelo de representación del entorno a partir de una matriz de valores de píxeles. Sin embargo, para la visión del robot también se deben incorporar aspectos de la robótica en sus algoritmos, como la cinemática y la calibración del marco de referencia [2]. Por ejemplo, para recoger un objeto dado debemos enviar información sobre la posición y orientación del objeto al controlador del robot antes de que cierre la pinza. En la literatura, estos procesos se traducen desde la identificación de la menor cantidad de puntos de agarre dadas imágenes 2D de objetos no antes vistos [3], [4], hasta la construcción de modelos 3D a partir de datos RGB-D y la solución de problemas de optimización para predecir la mejor posición de agarre para objetos conocidos [5]–[7]. Tenga en cuenta que, para la mayoría de los problemas, los puntos de contacto de la pinza están ubicados en un plano 2D y, por lo tanto, los métodos de agarre desarrollados para polígonos 2D pueden fácilmente extenderse a formas 3D.

En este trabajo, dividimos la tarea de recoger un objeto dado del plano de trabajo en dos partes: el reconocimiento de objetos y la búsqueda de puntos de agarre apropiados. En tareas visuales como el reconocimiento de objetos [8], [9] y la detección de objetos [6], [11], los métodos basados en el aprendizaje profundo muestran una mejora en el rendimiento del estado de la técnica. Como agarrar un objeto es inherentemente un problema de reconocimiento, utilizamos una red neuronal convolucional (ConvNet)[6] para clasificar los objetos. Por otro lado, el problema de agarre se resuelve identificando algunos puntos candidatos de donde agarrar la pieza haciendo uso de la transformada de Hough [12]. La transformada de Hough es una técnica para detectar figuras en imágenes digitales, agrupando los puntos que pertenecen a los bordes de las figuras. En la Fig. 1 se muestra un diagrama del sistema implementado. Observe cómo la cámara, ubicada en una posición fija con respecto al plano de trabajo del manipulador, cierra el bucle entre la detección de objetos en el plano de trabajo, la computadora integrada y el robot industrial. La computadora realiza el procesamiento de la imagen y envía las señales de control al controlador del robot, que a su vez controla el manipulador y su efector. Los resultados obtenidos son comparables a los del estado del arte ([4] por ejemplo) e incluso los agarres propuestos muestran similitud a implementaciones que utilizan información de profundidad (3D) [6].

II. Reconocimiento de Objetos

Las ConvNets son redes neuronales en las cuales el flujo de información tiene lugar en una sola dirección, desde sus entradas hasta sus salidas. Las ConvNets tienen la capacidad de aprender representaciones de manera automática, lo que generalmente solo es posible cuando se dispone de muchos

Elio Ogas, LABME (FICA), Universidad Nacional de San Luis, San Luis, Argentina, ogaselio@gmail.com.

Luis Avila, LABME (FICA) and LIDIC (FCFMyN), Universidad Nacional de San Luis, San Luis, Argentina, loavila@unsl.edu.ar.

Guillermo Larregay, LABME (FICA), Universidad Nacional de San Luis, San Luis, Argentina, golarregay@unsl.edu.ar.

Daniel Moran, LABME (FICA), Universidad Nacional de San Luis, San Luis, Argentina, dmoran@unsl.edu.ar.

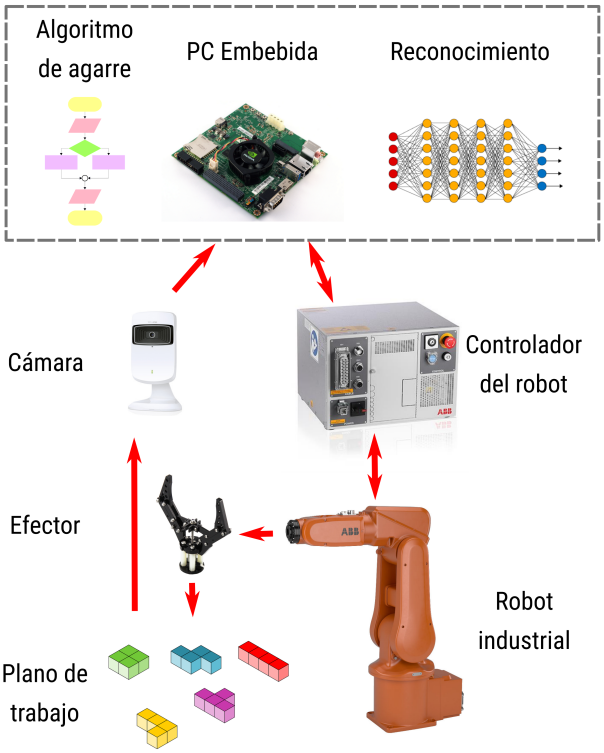


Fig. 1. Diagrama en bloques del sistema implementado.

datos de entrenamiento, especialmente para problemas en los que las muestras de entrada son de alta dimensionalidad como las imágenes. Sin embargo, las redes neuronales convolucionales son, por diseño, uno de los mejores modelos incluso cuando se tienen pocos datos para aprender. Esto se debe a que las ConvNets son capaces de capturar estructuras jerárquicas a partir de imágenes, donde las capas inferiores responden a características primitivas como bordes, esquinas y patrones comunes compartidos, y las capas superiores extraen información semántica como partes de objetos o caras.

La arquitectura general para una ConvNet se muestra en la Fig. 2, a partir del modelo ampliamente adoptado AlexNet [8]. Las capas de regularización (ReLU, *Rectified Linear Unit*) ayudan a minimizar el ajuste excesivo a los datos, al evitar que una capa vea varias veces el mismo patrón. Dado que el conjunto de entrenamiento utilizado es pequeño, los datos fueron aumentados (data augmentation) mediante varias transformaciones aleatorias para que la red vea diferentes configuraciones de la misma imagen. Esto también ayuda a evitar el sobreajuste y ayuda a que el modelo generalice mejor.

La arquitectura de la ConvNet implementada consiste en una pila de tres capas convolucionales alternadas con capas ReLU y capas de reducción (pooling) en varias etapas. Las capas convolucionales sirven para extraer patrones de las imágenes de entrada. Las funciones de activación no lineal permiten la extracción de características no lineales. ReLU es una función lineal por partes que retiene solo la parte positiva de la activación, reduciendo la parte negativa a cero. El propósito de las capas de reducción es minimizar la resolución espacial de los mapas de características. Específicamente, las capas de

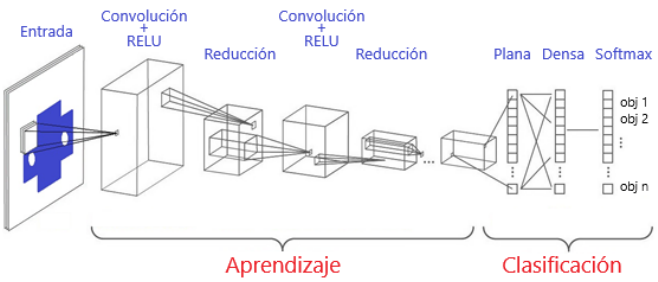


Fig. 2. Arquitectura de una red convolucional.

reducción máxima (maxpooling) propagan el valor máximo dentro de un campo receptivo a la siguiente capa [13]. Las capas planas y densas (totalmente conectadas, *fully connected*) que siguen a estas capas interpretan las representaciones de entidades y realizan la función de razonamiento de alto nivel. Finalmente, una capa de clasificación (función softmax) normaliza exponencialmente su entrada y clasifica la imagen del objeto.

III. Método de Agarre

Los datos de entrada al sistema de visión artificial son imágenes provenientes de cámara web inalámbrica, y se cuenta con la opción de que el usuario puede seleccionar un objeto mediante una interfaz gráfica de usuario que se ejecuta en una PC. Debido a que puede haber cualquier número de objetos diferentes dispersos al azar en el plano de trabajo, el robot tiene que discernir en primer lugar el objeto requerido de los demás. Cada imagen de la cámara se usa como entrada directa a la ConvNet, y una vez que se ha reconocido la pieza requerida, el controlador del robot genera la trayectoria desde la posición inicial (home del robot) a la ubicación de la pieza para poder realizar el agarre.

A. Calibración de la Cámara

Los datos de la posición de un objeto en el plano de trabajo son una proyección bidimensional de la imagen del objeto en el sensor de la cámara, por lo que es necesario conocer su representación tridimensional para que el robot pueda interpretarla adecuadamente. Para lograr esto, es crucial calibrar los parámetros de la cámara de acuerdo a la localización del efector final del robot. Estos parámetros se utilizan para corregir la distorsión de la lente, medir el tamaño de un objeto en coordenadas del mundo o determinar la ubicación de la cámara. Podemos dividir los parámetros de la cámara en dos grupos: parámetros que dependen del hardware de la cámara (intrínsecos) y aquellos que dependen del entorno (extrínsecos). El primer grupo consiste en una matriz de rotación R y una matriz de traslación t, mientras que el segundo grupo incluye la distancia focal, el centro óptico, también conocido como el punto principal, y el coeficiente de sesgo. Para determinar el conjunto de parámetros, se tomaron varias imágenes de un tablero cuyas especificaciones son conocidas y se utilizaron funciones de la biblioteca OpenCV [14]. Los parámetros de cámara obtenidos se representan en una matriz de 4 por 3 conocida como matriz de cámara.

La matriz representa un mapeo entre las coordenadas de referencia 3D y las coordenadas de imagen 2D [15]. Más precisamente, el algoritmo de calibración calcula la matriz de la cámara utilizando los parámetros extrínsecos obtenidos y los parámetros intrínsecos.

B. Coordenadas del Robot

El siguiente paso es transformar los datos de posición y orientación de las coordenadas de la cámara a las coordenadas del robot. Esto se hace mediante el uso de operaciones de traducción y orientación entre estos sistemas de coordenadas. Primero, la matriz de la cámara se usa para transformar la posición de un objeto de píxeles a milímetros, y luego agregar la información de profundidad de las dimensiones conocidas del objeto detectado, es decir, pasar de un sistema bidimensional a un sistema tridimensional. Para lograr esto, el modelo de la cámara sin lente (pin-hole) nos permite describir matemáticamente la relación entre estos parámetros [16]. Dado que este modelo no tiene en cuenta la distorsión de la lente, podemos escribir

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \left(R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \right) \tag{1}$$

donde $\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ son las coordenadas tridimensionales del objeto con respecto al sistema de coordenadas de la cámara y $\begin{bmatrix} u & v & 1 \end{bmatrix}^T$ las coordenadas de proyección del objeto en píxeles. La matriz A contiene los parámetros intrínsecos de la cámara, mientras que R y t son las matrices de rotación y traslación de la cámara respectivamente. El parámetro s corresponde a un factor de escala. El parámetro del eje Z (altura del objeto) se establece previamente de acuerdo con las dimensiones del objeto. Los puntos $\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ definidos en las coordenadas tridimensionales de la cámara deben transformarse en las coordenadas del robot. Para hacer esto necesitamos calcular las operaciones de rotación y traslación en el sistema original [17]:

$$\begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix} = R_{eff} \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} - tcp_{off} \right) + \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} \tag{2}$$

donde $\begin{bmatrix} X_m & Y_m & Z_m \end{bmatrix}^T$ son las coordenadas con respecto al sistema de coordenadas del mundo (base del robot); R_{eff} es la matriz de cambio de orientación de la cámara con respecto al robot; tcp_{off} es una matriz que contiene una desviación en la distancia del TCP del robot de las coordenadas del objeto; $\begin{bmatrix} X_0 & Y_0 & Z_0 \end{bmatrix}^T$ es el vector de traslación de la cámara con respecto al robot. Dado que la herramienta del robot debe colocarse correctamente para capturar el objeto, la orientación debe definirse en el sistema de coordenadas del robot

$$R_{mundo} = R_{home} R_{cam} R_{obj} \tag{3}$$

donde R_{mundo} es la matriz de rotación con respecto a las coordenadas del mundo, R_{obj} es la matriz de rotación del objeto, R_{cam} es la matriz de rotación del objeto con respecto a la cámara y R_{home} es la matriz de rotación de la cámara con respecto al robot.

C. Procesamiento de la Imagen

Dado que las condiciones de un entorno industrial pueden no ser óptimas en términos de iluminación y ruido, las imágenes capturadas necesitan cierto procesamiento antes de ser enviadas al algoritmo de agarre. Primero, se aplica un filtro gaussiano para reducir el nivel de ruido. En la Fig. 3a podemos observar la imagen original de dos piezas en escala de grises y en la Fig. 3b después de aplicar el pre-procesamiento al tipo de datos administrado por las funciones OpenCV.

La segmentación basada en el umbral [18] se utiliza para obtener regiones bien definidas entre diferentes objetos. Este proceso consiste en asignar un rango de valor al grupo de píxeles que contienen el nivel de color que corresponde al objeto de interés, mientras que al fondo se le asigna un rango de valores diferente. Para realizar la segmentación por umbral, elegimos trabajar en el espacio de color HSV [19], esto permite definir el color de acuerdo con parámetros familiares como el tono, la saturación y el brillo. Los picos y valles del histograma de imágenes pueden ayudar a elegir el valor adecuado para los umbrales de segmentación. Para lograr esto, el método intenta encontrar el nivel de umbral óptimo que divide el histograma en dos clases: el fondo y el primer plano [20]. Este método pesa el histograma y elimina repetidamente el componente del lado más pesado hasta que los bordes de la balanza se igualan. Como los bordes de una imagen están representados por la discontinuidad en los valores de los píxeles vecinos, en la Fig. 3c se muestra una representación binaria de la imagen. Aquí, las operaciones de erosión y dilatación se utilizaron para eliminar pequeños puntos que no corresponden al objeto y rellenar los orificios que aparecen en ellos.

Finalmente, la Fig. 3d muestra los contornos detectados en la imagen, que presentan un buen seguimiento de la forma de la pieza y no muestran grandes discontinuidades. También calculamos los momentos de la imagen para el objeto detectado. Los momentos en imágenes digitales corresponden a descriptores de forma que se estiman integrándose a través de todos los píxeles de una representación binaria. Los momentos de imagen resultan muy útiles para describir objetos después de realizar la segmentación, ya que proporcionan numerosos parámetros como área (o intensidad total), centroide, orientación y más. Como ejemplo, la Fig. 3d muestra la ubicación del centroide en el objeto y el valor en píxeles de ese punto. Aquí, la orientación estimada se mide con respecto al eje de coordenadas horizontal.

D. Búsqueda de los Lados para el Agarre

La extracción del conjunto completo de parámetros de características lineales en imágenes como dirección, centroide o punto medio, longitud y ancho, es un paso fundamental para el análisis automatizado de imágenes digitales [21]. En este sentido, utilizamos la detección de bordes para obtener puntos de imagen que se encuentran en una curva hipotética en el espacio de la imagen. Sin embargo, puede haber algunas

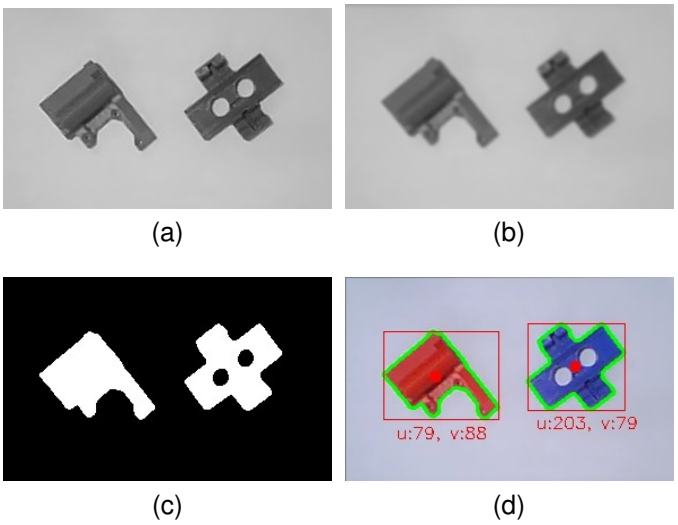


Fig. 3. Procesamiento de imágenes: a) imagen original en escala de grises, b) imagen filtrada, c) representación binaria, d) contorno y centroide del objeto.

imperfecciones en los datos de la imagen o en el detector de bordes debido a la falta de puntos o píxeles en las curvas deseadas. El propósito de la transformación de Hough [22] es resolver este problema agrupando las características de borde extraídas en candidatos objeto mediante un procedimiento de votación explícito [18]. El procedimiento de votación se lleva a cabo en un espacio paramétrico, a partir del cual los elementos candidatos objeto se obtienen como máximos locales. A pesar de que la transformación clásica de Hough se refería a la identificación de líneas en la imagen, posteriormente se amplió el método para permitir la identificación de formas arbitrarias [23].

El caso más simple de la transformada de Hough es detectar líneas rectas. En general, una línea recta $y = mx + b$ puede representarse como un punto (b, m) en el espacio de parámetros. Sin embargo, las líneas verticales plantean un problema, ya que pueden dar lugar a valores ilimitados del parámetro de pendiente m. Así, por razones computacionales, Duda y Hart propusieron el uso de la forma normal de Hesse

$$r = x * cos(\theta) + y * sen(\theta) \tag{4}$$

donde r es la distancia desde el origen hasta el punto más cercano en la línea recta, y θ es el ángulo entre el eje x y la línea que conecta el origen con el punto más cercano. Es posible asociar con cada línea de la imagen un par (r, θ). El plano (r, θ) a veces se denomina espacio de Hough para el conjunto de líneas rectas en dos dimensiones. Dado un único punto en el plano, entonces el conjunto de todas las líneas rectas que pasan a través de ese punto corresponde a una curva sinusoidal en el plano (r, θ), que es único para ese punto.

En este trabajo, utilizamos la versión probabilística del algoritmo de transformación de Hough [24] que no tiene en cuenta todos los puntos de borde, sino solo un subconjunto aleatorio de ellos. Para este algoritmo, necesitamos introducir dos nuevos parámetros. La longitud mínima de línea permite rechazar todos los segmentos de línea que sean más cortos que un valor predefinido. A su vez, el espacio máximo permitido entre los segmentos de línea se utiliza para considerar un conjunto dado de puntos cercanos como una sola línea. Al establecer estos parámetros obtenemos dos puntos finales de un segmento de borde, y un punto de agarre candidato que reside en el punto medio del segmento. Observe que al elegir un parámetro de línea de longitud mínima lo suficientemente pequeño, podemos considerar segmentos curvos para encontrar puntos de agarre.

E. Búsqueda de los Puntos de Agarre

En adelante, abordamos el problema de sintetizar agarres planos que tienen un cierre de fuerza. Un agarre sobre un objeto es un cierre de fuerza si y solo si podemos ejercer, a través del conjunto de contactos, una fuerza arbitraria sobre el objeto [25]. Un agarre se describe mediante dos puntos en el borde de la pieza, mientras que la línea que conecta esos puntos se conoce como el eje de agarre, ver la Fig. 4. Para seleccionar los mejores puntos de agarre, debemos considerar cada par de segmentos de borde adecuados. Asumimos que el contacto de la pinza con el objeto ejerce una fuerza en forma de conos de fricción (en ambos lados de la pieza) y proporcionamos un par alrededor del eje de la empuñadura cuando se levanta la pieza. Para explicar el método, consideremos el par de segmentos s_1 y s_4 en la Fig. 4. Sea O el punto en el que estos segmentos se intersecan. Se dibuja una bisectriz ponderada del ángulo $\beta_{1,4}$ entre los segmentos, de manera que el ángulo entre el segmento s_1 o s_4 y $\beta_{1,4}$ se define como [26].

$$\gamma_1 = \left(\frac{\alpha_1}{\alpha_1 + \alpha_4}\right)\beta_{1,4} \tag{5}$$

$$\gamma_4 = \left(\frac{\alpha_4}{\alpha_4 + \alpha_1}\right)\beta_{1,4} \tag{6}$$

Un cono de fricción se define por medio ángulo $\alpha = arctan(\mu)$, donde μ es el coeficiente de fricción entre el objeto y la pinza. Recorremos cada par de segmentos posibles en el objeto, rechazando los pares que no contienen un agarre factible. Para hacer esto, comparamos si el ángulo $\beta_{j,k}$ entre los segmentos e_j y e_k es mayor que la suma de los ángulos de cono de fricción para esos segmentos, es decir, $\beta_{j,k} > (\alpha_j + \alpha_k)$. Si ningún agarre en este par de segmentos puede cumplir con los criterios de fricción, entonces el par es rechazado. Al hacer esto, evitamos que el objeto se deslice del efector durante el agarre.

Una vez que se prueban todos los pares, se obtiene un conjunto de puntos de agarre candidatos. Para cada uno de esos candidatos, calculamos la distancia desde el eje de agarre hasta el centroide de la pieza para minimizar la probabilidad de falla al levantar el objeto. Finalmente, evaluamos los n puntos de agarre más cerca del centroide y seleccionamos el más corto como la mejor posición de agarre.

IV. EXPERIMENTALES

Para evaluar el rendimiento del reconocimiento de objetos y la performance de agarre del efector del robot, realizamos una secuencia de experimentos en un robot ABB IRB120

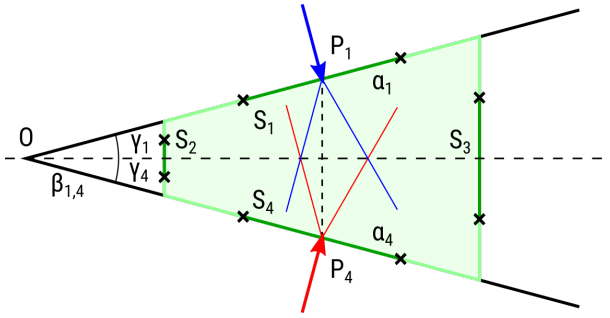


Fig. 4. Bisector para dos segmentos evaluados y sus conos de fricción.

con 6 grados de libertad y un controlador IRC5 Compact. Como efector final, utilizamos una pinza de dos dedos con una abertura máxima de 76 mm. La entrada a los algoritmos de visión es proporcionada por una cámara web inalámbrica TP-Link NC200 con resolución de 720p a 25 cuadros por segundo, ubicada a 85 cm por encima del plano de trabajo. Tanto el algoritmo de visión artificial como el método de agarre robótico se ejecutan en una computadora embebida NVIDIA Jetson TK1. La computadora posee un SOC Tegra que integra una CPU NVIDIA 4-Plus-1 ARM Cortex de cuatro núcleos y una GPU NVIDIA Kepler con 192 núcleos CUDA. Utilizamos la plataforma ROS como sistema de desarrollo. Los datos se envían desde la computadora integrada al controlador del robot a través de una red TCP/IP. En el controlador del robot, se creó un socket de servicio utilizando Rapid, que recibe las conexiones de comandos remotos desde la placa Jetson y envía señales de comando a los actuadores del robot. La aplicación de visión funciona como un cliente que se conecta al servidor mediante un protocolo ad-hoc implementado para el sistema completo.

La tarea del sistema de manejo del robot comienza con el reconocimiento del objeto. El conjunto de datos utilizado para entrenar la ConvNet consta de 300 imágenes de diez objetos distintos. Luego aumentamos el número de imágenes de entrenamiento para lograr 20 nuevas imágenes, lo que da un total de 6000 imágenes para el conjunto de datos de entrenamiento. La entrada a la ConvNet es un parche de imagen extraído alrededor del objeto. Para obtener este parche, utilizamos el detector de bordes Canny [27], que permite estimar la ubicación exacta de los bordes de los objetos. Con esta información podemos determinar dónde se ubican los objetos en el plano de trabajo y luego extraer un parche de imagen de cada uno de ellos. La resolución de las imágenes capturadas por la cámara es de 640x480 píxeles y luego se redimensionan a 128x128 píxeles en el espacio RGB (3 componentes de color), para coincidir con las dimensiones de las entradas ConvNet.

La arquitectura de red neuronal consiste en una pila de tres capas convolucionales implementadas en Python usando la biblioteca de redes neuronales Keras que se ejecuta sobre Tensorflow [29], una biblioteca de código abierto desarrollada por Google para el aprendizaje automático. Empleando el optimizador Adam [30], un algoritmo que es una extensión del descenso estocástico por el gradiente (SGD, *Stochastic Gradient Descent*) con tasas de aprendizaje adaptivas, el entrenamiento de esta red tomó aproximadamente 30 minutos. Para cada conjunto de validación cruzada, entrenamos cada modelo durante 25 épocas. Durante el entrenamiento, el modelo se evaluó en un conjunto de datos de validación después de cada época. El proceso de entrenamiento se monitoreo comparando la performance del modelo en el dataset de entrenamiento y la performance en el dataset de validación. Cuando el rendimiento del modelo en el conjunto de datos de validación comienza a degradarse (la precisión comienza a disminuir), entonces el proceso de entrenamiento se detiene (early stopping). Se reportan precisiones de 0.97 en el set de entrenamiento y de 0.91 en el set de validación. Los resultados son buenos debido a que se utilizó regularización y se aumentó el set de entrenamiento (data augmentation).

El procesamiento de la imagen obtenida de la cámara consiste en convertir del espacio de color RGB al espacio HSV, del cual se toma el canal S, se filtra con un kernel gaussiano de desenfoque, y se aplica el proceso de umbralización para la detección de bordes. Con el fin de eliminar posibles ruidos en la imagen ya binarizada, se aplica la operación morfológica de apertura (*opening*) que consiste en una erosión y una dilatación. El kernel utilizado para la operación de erosión fue una circunferencia de dimensiones 5x5 píxeles, y para la de dilatación fue una circunferencia de dimensiones 1x1 píxeles.

Para detectar el grupo de puntos que forman un segmento de borde que se puede agarrar usando la transformada probabilística de Hough, establecemos el número mínimo de puntos que pueden formar una línea en *minLinLength*=25, y la brecha máxima entre dos puntos a considerar en la misma línea en *maxLineGap*=15. Es importante recordar que la cantidad de fricción que puede soportar el contacto de la pinza y un objeto afecta en gran medida el rendimiento. El enfoque de los conos de fricción implica tener un tipo de material asociado para especificar el coeficiente de fricción para el contacto. Por ejemplo, cuando los objetos detectados son plásticos impresos en 3D (PLA o ABS) el coeficiente de fricción utilizado es $\mu = 0.3$. Al cambiar el material de los enlaces, podemos variar el coeficiente de fricción y, en consecuencia, el algoritmo de agarre encontrará más o menos agarres disponibles. El coeficiente de fricción entre las superficies plásticas y metálicas se define como 0.2, y entre el caucho y el metal es 1.0 [28].

En la Fig. 5 mostramos el algoritmo de agarre en ejecución. Los contornos detectados y la ubicación del centroide en el objeto se muestran a la izquierda y el paso de reconocimiento del objeto a la derecha. También pueden verse las coordenadas del objeto en el plano, un paso fundamental para determinar el movimiento del manipulador. Probamos el método de agarre con un conjunto mixto de objetos y asumiendo que un objeto se puede agarrar desde cualquier dirección. Por lo tanto, el brazo del robot debe moverse a lo largo del camino hacia la posición de agarre, recoger el objeto y transportarlo a la posición de destino. Para los experimentos, utilizamos una GUI para que el operador pueda seleccionar el objeto deseado. Además, se puede establecer una secuencia predefinida de agarre de objetos para ensamblaje.

Una vez que la ConvNet identifica el objeto de interés y se ejecuta el algoritmo de agarre. Posteriormente, necesitamos

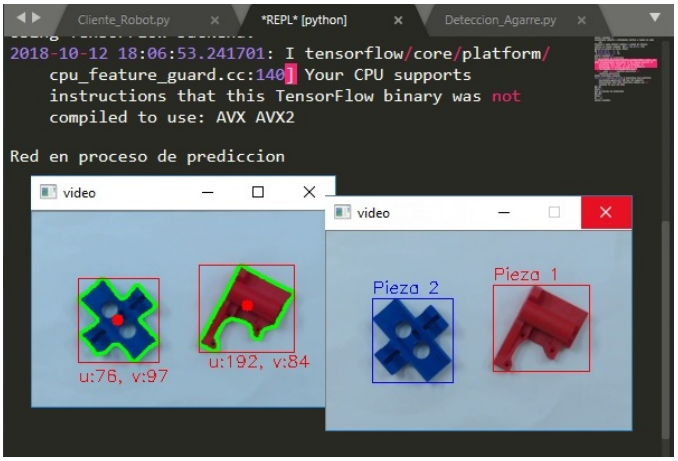


Fig. 5. El enfoque de agarre en ejecución: el contorno de la imagen a la izquierda y el reconocimiento de los objetos a la derecha.

encontrar la postura del brazo que permita lograr el agarre computado. Al resolver la cinemática inversa del robot a través de las rotaciones de las matrices indicadas en la Ec. 3, se genera una ruta hacia la ubicación de la pieza teniendo en cuenta la posición y la orientación de la pinza. Finalmente, la pieza se sujeta, se levanta y se coloca en una dada posición de destino.

En la Fig. 6 se muestra el resultado de comparar el desempeño del algoritmo bajo diferentes condiciones de iluminación. En particular, se hicieron pruebas a 15 lux sobre el plano de trabajo (Fig. 6a), a 60 lux (Fig 6b) y a 170 lux (Fig. 6c). Se debe tener en cuenta que la mayoría de las cámaras tienen un fotómetro interno para compensar la exposición de acuerdo a la iluminación de la escena, agregando ruido cromático en casos de poca luz al amplificar la baja señal del sensor. Sin embargo, como se puede ver, el algoritmo ha demostrado ser robusto incluso en situaciones de muy baja iluminación, encontrando siempre un agarre correcto.

La Fig. 7 muestra varios agarres exitosos e incorrectos procesados por el método. Un agarre se considera exitoso si durante el proceso de agarrar y levantar el objeto, no se resbala de la pinza. La Fig. 7a muestran varios agarres precisos para objetos con diferentes formas. En la Fig. 7b se muestran dos casos aislados de agarres incorrectos, originados debido a que la detección de bordes falla en la determinación de bordes externos e internos en los orificios. El algoritmo implementado analiza en tiempo real una cierta cantidad de cuadros de la secuencia de imágenes de la cámara para detectarlos, y en algunos casos aislados, hemos detectado una duplicación de algunos de estos bordes. El conteo de intersecciones entre el segmento de agarre y los bordes de las caras internas del objeto se ve afectado, y esto provoca que se seleccione un agarre incorrecto. Esto podría evitarse implementando un algoritmo de detección de segmentos de líneas duplicados, o paralelos a una distancia muy pequeña.

V. Conclusiones

Este artículo presenta el diseño de un sistema robótico completo de agarre mediante un efector final que incluye

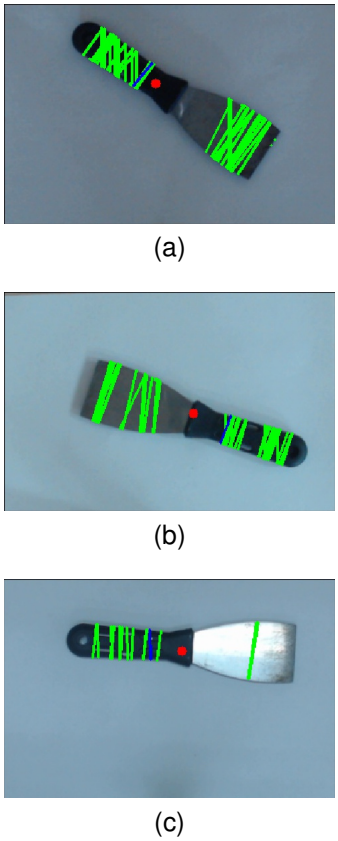
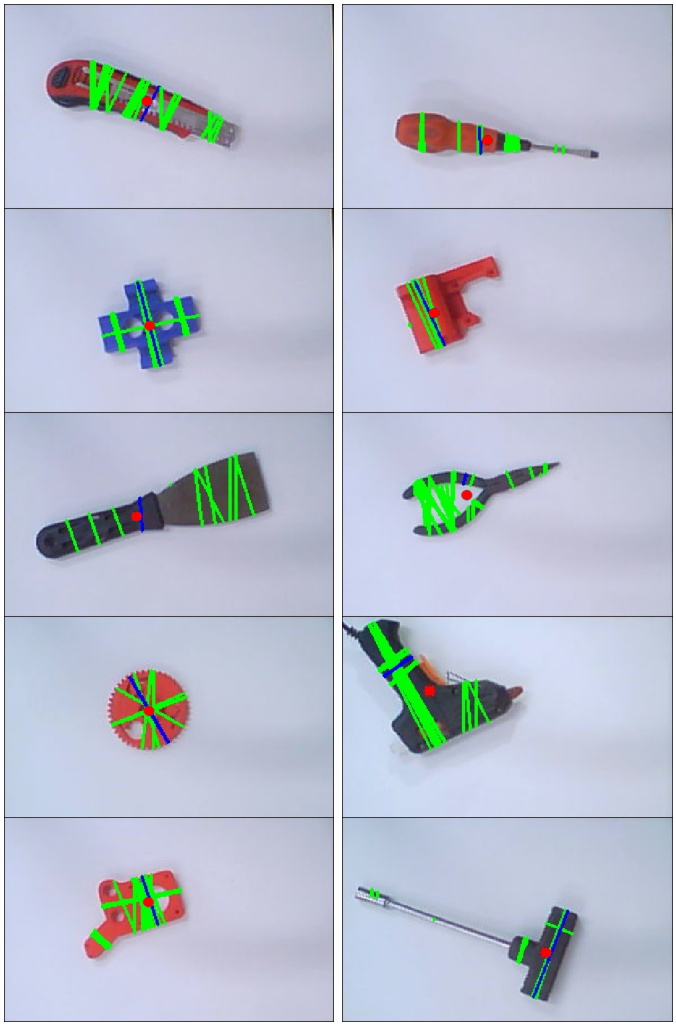


Fig. 6. Desempeño del algoritmo a diferentes niveles de iluminación: a) 15 lux sobre el plano de trabajo, b) 60 lux, c) 170 lux.

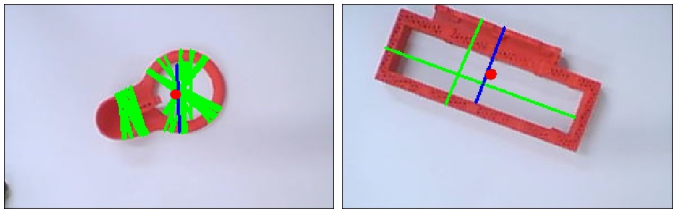
el reconocimiento de imágenes con una cámara monocular y un algoritmo de planificación de agarre para recoger una variedad de objetos en el plano de trabajo. La implementación se dividió en dos tareas, primero, se utilizó una ConvNet para el reconocimiento de objetos y luego se introdujo un método de agarre basado en la transformación de Hough y conos de fricción para permitir la manipulación de objetos.

Dado que el reconocimiento de objetos y el problema de agarre son independientes entre sí, sería posible implementar diferentes técnicas en cada etapa. La principal ventaja de nuestro enfoque es que no es necesario entrenar un modelo con las posiciones de agarre correctas, por lo que es posible generalizar a otros objetos nunca antes vistos. Para el método de agarre se asumió una línea de ensamblaje industrial donde los parámetros de los objetos manipulados se conocen a priori. Por lo tanto, una vez que el robot ha identificado el objeto solicitado, se pueden recuperar de la memoria parámetros como la forma y las dimensiones en el eje z. En general, encontramos que nuestra implementación funciona bien y que el robot pudo recoger una variedad de objetos. De forma concisa, el sistema de agarre presentado ha demostrado la viabilidad del enfoque.

Una de las ventajas de nuestro enfoque es que dado que el reconocimiento de objetos y el problema de detección del mejor agarre son independientes entre sí, sería posible la utilización de diferentes técnicas en cada etapa. Esto, sumado a la existencia de métodos que permiten re-entrenar una red profunda para incorporar nuevos objetos a clasificar sin necesi-



(a)



(b)

Fig. 7. Resultados de pruebas de agarre: a) Posiciones de agarre exitosas estimadas por el algoritmo, b) Ejemplo de agarre de fallas.

dad de partir desde cero (mediante transferencia de aprendizaje por ejemplo), hace posible que nuevos objetos sean fácilmente agregados a la base de datos del sistema de reconocimiento. Frente a otros enfoques actuales, esta implementación tiene además la ventaja de no utilizar un modelo a priori con las posiciones de agarre correctas por lo que es posible generalizar a otros objetos nunca antes vistos. Si bien esta corresponde a una primera aproximación al problema más complejo de predicción del agarre en 3 dimensiones, es clave aclarar que en una línea de ensamblaje industrial los parámetros de los objetos manipulados se conocen a priori. Por lo tanto, una vez que el robot ha identificado el objeto solicitado, se pueden recuperar de la memoria parámetros como la forma y las dimensiones en el eje z.

El siguiente paso es extender el enfoque propuesto al dominio de los datos RGB-D, que no solo plantea el desafío de manejar los datos de profundidad de la escena en una red profunda, sino también predecir los agarres no planos. Esto permitirá un aumento del número de objetos a reconocer. También es necesario evaluar el comportamiento del sistema frente a condiciones mas complejas para el sensor (cámara) a fin de verificar la robustez del mismo. Un video del sistema completo funcionando se ha subido al link https://youtu.be/atxsZzPNup4.

REFERENCIAS

[1] W. Wan, *Using Intelligent Robots to Assemble Automobile Parts*, Adv. Automob. Eng., vol. 06, no. 01, pp. 1–2, Feb. 2017.

[2] O.-H. Alex, *Robot Vision vs Computer Vision: What's the Difference?*, 2016. [Online]. Available: https://blog.robotiq.com/robot-vision-vs-computer-vision-whats-the-difference. [Accessed: 27-Aug-2018].

[3] A. Saxena, J. Driemeyer, and A. Y. Ng, *Robotic Grasping of Novel Objects using Vision*, Int. J. Rob. Res., vol. 27, no. 2, pp. 157–173, Feb. 2008.

[4] D. Hossain, G. Capi, M. Jindai, and S. Kaneko, *Pick-place of dynamic objects by robot manipulator based on deep learning and easy user interface teaching systems*, Ind. Robot An Int. J., vol. 44, no. 1, pp. 11–20, Jan. 2017.

[5] I. Lenz, H. Lee, and A. Saxena, *Deep learning for detecting robotic grasps*, Int. J. Rob. Res., vol. 34, no. 4–5, pp. 705–724, Apr. 2015.

[6] J. Redmon and A. Angelova, *Real-time grasp detection using convolutional neural networks*, in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 1316–1322.

[7] E. Johns, S. Leutenegger, and A. J. Davison, *Deep learning a grasp function for grasping under gripper pose uncertainty*, in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016, pp. 4461–4468.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*. pp. 1097–1105, 2012.

[9] C. Silva, D. Welfer, F. P. Gioda, and C. Dornelles, *Cattle Brand Recognition using Convolutional Neural Network and Support Vector Machines*, IEEE Lat. Am. Trans., vol. 15, no. 2, pp. 310–316, Feb. 2017.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*. pp. 779–788, 2016.

[11] S. Ren, K. He, R. Girshick, and J. Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. pp. 91–99, 2015.

[12] J. Illingworth and J. Kittler, *A survey of the hough transform*, Comput. Vision, Graph. Image Process., vol. 44, no. 1, pp. 87–116, Oct. 1988.

[13] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, *Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition*, in 2007 IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[14] G. R. Bradski and A. Kaehler, *Learning OpenCV: computer vision with the OpenCV library*. O'Reilly, 2008.

[15] J. Heikkila and O. Silven, *A four-step camera calibration procedure with implicit image correction*, in Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997, pp. 1106–1112.

[16] Z. Zhang, *A Flexible New Technique for Camera Calibration*, IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, Dec. 2000.

[17] J. Craig, *Introduction to Robotics: Mechanics and Control*. Peason Education. Inc., publishing as Prentice Hall, 2005.

[18] L. Shapiro and G. Stockman, *Computer vision*. Prentice Hall, 2001.

[19] S. Sural, Q. Gang, and S. Pramanik, *Segmentation and histogram generation using the HSV color space for image retrieval*, in Proceedings International Conference on Image Processing, 2002, vol. 2, p. II-589-II-592.

[20] M. Sezgin and B. Sankur, *Survey over image thresholding techniques and quantitative performance evaluation*, J. Electron. Imaging, vol. 13, no. 1, p. 146, Jan. 2004.

[21] P. Mukhopadhyay and B. B. Chaudhuri, *A survey of Hough Transform*, Pattern Recognit., vol. 48, no. 3, pp. 993–1010, Mar. 2015.

[22] R. O. Duda and P. E. Hart, *Use of the Hough transformation to detect lines and curves in pictures*, Commun. ACM, vol. 15, no. 1, pp. 11–15, Jan. 1972.
[23] G. J. Bergues, L. Canali, C. Schurrer, and A. G. Flesia, *Sub-pixel Gray-scale Hough Transform For An Electronic Visual Interface*, IEEE Lat. Am. Trans., vol. 13, no. 9, pp. 3135–3141, Sep. 2015.
[24] J. Matas, C. Galambos, and J. Kittler, *Robust Detection of Lines Using the Progressive Probabilistic Hough Transform*, Comput. Vis. Image Underst., vol. 78, no. 1, pp. 119–137, Apr. 2000.
[25] V.-D. Nguyen, *Constructing Force- Closure Grasps*, Int. J. Rob. Res., vol. 7, no. 3, pp. 3–16, Jun. 1988.
[26] G. Smith, E. Lee, K. Goldberg, K. Böhringer, and J. Craig, *Computing Parallel-Jaw Grips*, in IEEE International Conference on Robotics and Automation, 1999.
[27] J. Canny, *A Computational Approach to Edge Detection*, IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
[28] R. Pelossof, A. Miller, P. Allen, and T. Jebara, *An SVM learning approach to robotic grasping*, in Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, 2004, pp. 3512–3518.
[29] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
[30] D. Kingma, and J. Ba, *Adam: A Method for Stochastic Optimization*. International Conference on Learning Representations, 2014.