# Coloquio de Investigación Multidisciplinaria

VOLÚMEN 8 Núm.1 OCTUBRE 2020

Revista Periódica

latindex

OpenAIRE

## JOURNAL CIM - REVISTA DIGITAL

ISSN: 2007 - 8102

DIFUSIÓN VÍA RED DE CÓMPUTO
https://www.cim-tecnm.com/articulos

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

 ISSN 2007-8102

# Reconocimiento de gestos de una mano para la interacción en tiempo real con un modelo tridimensional.

*E. A. Cuellar Cortés 1, A. Hernández Cruz 2.*
*[1]Tecnológico Nacional de México – Instituto Tecnológico de Orizaba, Oriente 9 No. 852, Orizaba Veracruz.*
*[2]Universidad Tecnológica del Centro de Veracruz, Avenida Universidad 350, Cuitláhuac, Veracruz.*
*acuellarc@orizaba.tecnm.mx, alan.he.cruz@gmail.com*
**Área de participación:** Ingeniería Mecánica y Mecatrónica



## Resumen

Este articulo presenta un algoritmo y su metodología para el reconocimiento de gestos. En este caso particular gesto a identificar será el número de dedos que se encuentran levantados en una mano, es decir seis gestos, mano cerrada y cinco dedos levantados, todo esto utilizando técnicas de visión computacional, para posteriormente lograr una interaccion en tiempo real con un modelo tridimensional de una mano que copiara dichos gestos.

**Palabras clave:** *Python, OpenCV, Unity.*

## Abstract

This article presents an algorithm and methodology for gesture recognition. In this particular case, the gesture to be identified will be the number of fingers that are raised in a hand, (six gestures), closed hand and five fingers raised, all this using computer vision techniques, and get a real-time interaction with a three-dimensional model of a hand that will copy these gestures

**Key words:** *Python, OpenCV, Unity.*

## Introducción

Como producto y conclusión de mi residencia profesional se generó este este artículo científico que muestra la metodología de un algoritmo capaz de identificar una mano, las articulaciones, y el movimiento de las mismas mediante el visón computacional, una vez logrado el reconocimiento se reflejaran los movimientos en un ambiente virtual, para su posterior interacción. Este algoritmo fue desarrollado en conjunto de varios lenguajes de programación, como lo son Python y su librería OpenCV, C# y herramientas de modelado 3D como lo es Blender y Unity 3D. También se incluye él envió de datos entre los diferentes programas para realizar la interacción virtual. El programa principal fue codificado en Python, un lenguaje multiplataforma, de código organizado, de fácil lectura, estilo elegante, multiparadigma y una de sus grandes ventajas es que permite automatizar procesos, trabajar grandes volúmenes de datos y sus aplicaciones son muy variadas, gracias a la gran diversidad de librerías y frameworks que se encuentran para su libre uso. Una de estas librerías es OpenCV, es una librería de código abierto para visión por computadora, (Open Source Computer Vision), la cual cuenta a la fecha con mas de 500 algoritmos relacionados con el procesamiento de imágenes y visión computacional, es una las librerías con mas soporte y con constantes actualizaciones en los algoritmos. Otro lenguaje utilizado es C # (Sharp) es un lenguaje compatible con Unity para poder realizar la comunicación entre el mundo real y el mundo virtual, esté fue desarrollado por Microsoft y forma parte de su plataforma de lenguajes .NET. Es una mejora a los lenguajes C y C++. En la parte de modelado 3D, el primer software es Blender, un programa multiplataforma dedicado renderizado, animación, gráficos y al modelado tridimensional, una de las grandes ventajas es que al igual que Python es multiplataforma. Por último, tenemos Unity 3D, principalmente es un motor de videojuegos capaz de soportar varias plataformas, trabaja muy de la mano con Blender, y es capaz de crear interacciones en mundos virtuales, mapeados, iluminaciones, texturas, etc., todo lo relacionado con ambientes virtuales tridimensionales.

 ISSN 2007-8102

## Metodología

### Equipo utilizado

Para el desarrollo del proyecto se utilizó una computadora tipo laptop de la marca HP, modelo PAVILION con sistema operativo Windows 10 de 64 Bits, dicho dispositivo contaba con procesador AMD A9-9410 Radeon R5, 5 Compute Cores 2C+3G a 2.90 GHz, 12 GB de memoria RAM DDR3, 1 TB de disco duro mecánico.

### Software utilizado

Como ya se mencionó previamente, Python/ OpenCV es donde se programó la adquisición de la imagen, la etapa de pre procesamiento y la extracción de las características de la mano, además de enviar mediante UDP a Unity la información de la posición de los dedos, para que Unity realizara la animación en tiempo real del movimiento y Blender se utilizó en la exportación del modelado 3D de la mano.

### Algoritmo

El algoritmo siguió la metodología mostrada en la figura 1.

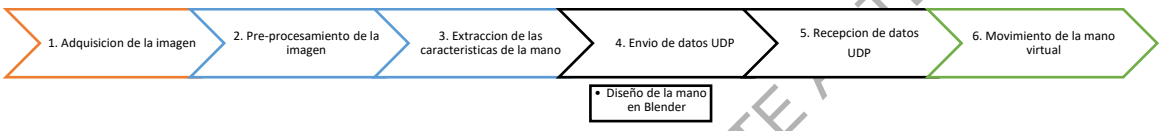


Figura 1. Diagrama a bloques de la metodología empleada.

El primer bloque nos muestra la adquisición de la imagen la cual se realizó mediante la librería OpenCV, con la instrucción VideoCapture(X), esto inicializa la cámara X de nuestra computadora, siendo X un numero entero que puede variar dependiendo del número de cámaras conectadas, en el caso de laptop el valor que normalmente se utiliza es 0 ya que se utiliza la cámara integrada en la pantalla, sin embargo como en este caso se conectó una cámara externa mediante USB se utilizó el número uno. La característica de la cámara externa de marca genérica, tenia una calidad de 1.3 megapíxeles con una resolución de 1280x1024.[9][15]

El segundo bloque expresado como pre procesamiento nos incluye varias etapas, los cuales describiré a continuación. Después de inicializar la cámara, se toma un frame, es decir una imagen, con la cual se llevará acabo todo el algoritmo, esto se repetirá infinitamente mientras el programa este ejecutándose. Posteriormente espejeamos o invertimos la imagen (flip) para facilitar la representación tridimensional, se genera un área de interés (ROI) indicándola en el video con un recuadro verde. En esa área de interés mostraremos la mano, buscando una visualización ideal, con un fondo obscuro. Una etapa mas en el pre procesamiento es la conversión entre espacios de color de RGB a HSV. Favoreciendo la identificación de color, en este caso, el color o tono de mi piel, estos valores pueden ser modificados dependiendo del usuario del programa, necesitando un rango bajo y un rango alto de tonalidad. Lo anterior sirve para segmentar la imagen, quitar el fondo y seleccionar únicamente la mano, mediante la función de inRange, la cual comprueba si los elementos de la matriz se encuentran entre los elementos de otras dos matrices (matriz de tono de piel bajo y matriz de tono de piel alto). Es así como aislamos o segmentamos la mano del resto de la imagen, además de obtener una imagen binaria (tomando los valores de umbral como rango bajo y rango alto de tono de piel, en HSV). Posteriormente se aplica una transformación morfológica de dilatación y un filtro de suavizado Gaussiano, obteniendo como resultado una imagen binaria de fondo negro y una mano de color blanco que gracias a los filtros se debe distinguir perfectamente. [6][1][5*

El tercer bloque es la extracción de características de la mano, esto se hace mediante la jerarquía de contornos, la definición de contornos es una curva que se une con puntos continuos, que tienen un límite de largo y una misma intensidad, los contornos son la herramienta ideal para el reconocimiento de objetos, detección y análisis de formas, como en este caso la mano. Como se menciono para un mejor funcionamiento deben utilizar imágenes binarias (intensidad) que proviene

 ISSN 2007-8102

del pre procesamiento. La función findContours necesita tres argumentos el primero es la imagen el segundo el modo de recuperación del contorno y el tercero el método de aproximación del contorno, generando una lista de la jerarquía encontrada en la imagen. Para el tercer parámetro se utilizó CHAIN_APPROX_SIMPLE el cual elimina todos los puntos redundantes y comprime el contorno, ahorrando memoria, el segundo parámetro se estableció RETR_TREE este parámetro recupera todos los contornos y crea una lista jerárquica familiar completa. Seguido se busca el área del máximo contorno encontrado en la imagen, que por lógica deberá ser la mano. Por ultimo en esta etapa se implementa convexHull, son dos palabras en inglés unidad convex – convexo y hull – cascara. La definición establece que cualquier región o forma es convexa si la línea que une dos puntos cualesquiera está contenida completamente en esa región. Es decir que una forma sea convexa todos sus ángulos interiores deben ser menores a 180 grados o todos sus vértices deben abrirse hacia el centro. para una forma determinada o un conjunto de puntos, podemos tener muchas curvas / límites convexos. El límite convexo más pequeño o ajustado se conoce como casco convexo. Con esto en mente lo que procede es encontrar los vértices de los dedos es decir la punta del dedo con la base, generando 4 vértices cuando los dedos se encuentren extendidos, o cero cuando se encuentre la mano cerrada. Además, se agregar líneas de código para facilitar visualmente la detección del numero de dedos y marcarlos. [13][4][3]

La siguiente etapa es el envió de información o datos mediante UDP, el cuarto bloque. El UDP es modelo de transmisión simple, y tiene como desventaja que no se puede garantizar la integridad o la fiabilidad de los datos, por lo tanto, es adecuado para la multidifusión o el envío a todos los suscriptores de una red local. El tráfico UDP, a diferencia del TCP, no requiere necesariamente una respuesta y no es necesario establecer una conexión para ser enviado. En Python existe la librería socket, que sirve para conectar dos nodos en una red y poder comunicarse entre sí. Un socket (nodo) escucha en un puerto particular (IP), mientras que otro socket forma una conexión. Por lo tanto, nuestro código en Python será quien envié las instrucciones a Unity indicando con una cadena de caracteres que dedo es el que se encuentra levantado, Unity estará “escuchando” y dependiendo de la cadena, realizará un movimiento. Lo anterior una vez procesado el número de vértices y mediante una comparación simple de if’s anidados se verifica el número de dedos levantados. [7]

Una etapa adicional es el uso de Blender. Se buscó en internet un modelo ya diseñado de una mano humana que fuera sin costo, para esto se recurrió a la biblioteca Free3D, encontrando un modelo tridimensional de una mano con licencia de uso personal. Se importo a Blender el archivo .obj descargado. Este archivo solo es el modelado tridimensional de la mano, pero para que ese modelo pudiera moverse e interactuar en Unity, fue necesario programar las articulaciones, o como las denomina Blender “Armature” por cada dedo se generaron tres “huesos” o articulaciones, además de la muñeca, ver Figura 2 es decir al menos 15 articulaciones que controlar y programar en Unity. Por lo que se decidió que el control principal estuviera en la base de los dedos, medicamente denominado articulación intermetacarpianas, ya que esta etapa inicial solo nos interesara saber que dedo de la mano se encuentra levanto.[12][10]

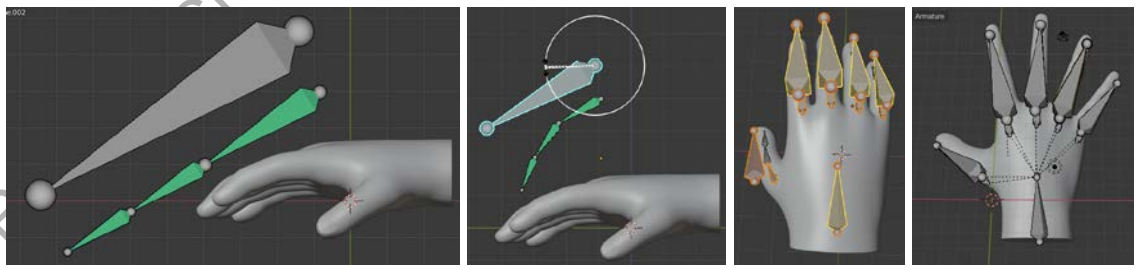

Figura 2. Blender armature: se muestra las tres articulaciones para el dedo índice (derecha) y la base del dedo donde se realizara un giro, para contraer los dedos y cerrar la mano (centro), todas las armature creadas en la mano ( derecha).

 ISSN 2007-8102

Con todo lo anterior listo se procedió a generar las animaciones del movimiento de los dedos. En el modo Pose de Blender se grabaron los movimientos de cada una de las articulaciones (antes mencionadas) para el desplazamiento de cada uno de los dedos, ver figura 3. Estas animaciones son los dedos moviéndose de arriba a bajo y la mano cerrada.[11]

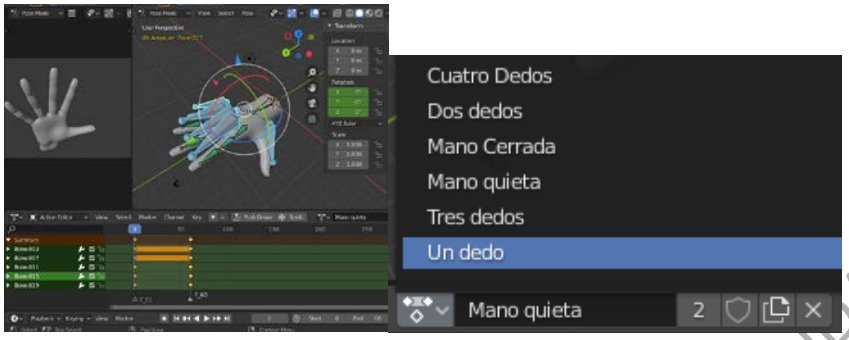


Figura 3. Modo Pose, se observan las grabaciones del desplazamiento de los dedos y a la derecha los nombres asignados a las grabaciones.

Teniendo las animaciones grabadas, procederemos a exportar del software Blender a Unity 3D, el motor de desarrollo acepta diferentes tipos de archivos compatibles para ser utilizados en su entorno, en este caso Blender nos permite exportar el modelo junto con todas las animaciones grabadas en formato “FBX” para que se exporte el modelo 3D y las animaciones, como también el ambiente (escenarios, enfoque de cámara, etc.).

En la etapa 5 recepción de datos UDP es necesario crear un script en C# dentro de Unity el cual recibirá los datos provenientes de Python, ver figura 4.

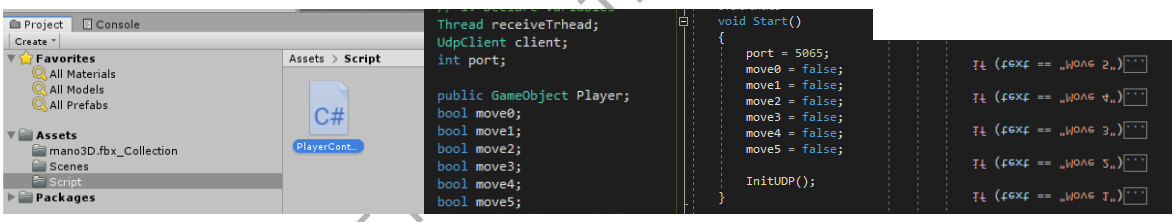


Figura 4. Se muestra el archivo del script de C # dentro de Unity y parte de su programación.

La recepción de los paquetes datos de Python al Script de C# se realiza través del protocolo UDP, y también el envío de los pulsos Trigger del Script de C# a las variables de entrada de tipo Trigger en Unity. Habiendo enviado la cadena de tipo Trigger a Unity 3D se procede a crear variables del mismo tipo en Unity que corresponderán a accionar los movimientos predeterminados, para accionar los movimientos se crean estados, esto corresponde como su nombre lo menciona al estado al cual pasara el modelo 3D cuando se detecte que se recibió el dato correspondiente a dicha acción, para asegurarnos que se cambió de acción al recibir el Trigger correspondiente al movimiento asignado se crean transiciones condicionadas con los datos obtenidos. Todas y cada uno de los estados creados deberán tener un movimiento predeterminado para que el modelo 3D hago dicho movimiento y deben estar conectados entre sí, dicho proceso se muestra en la Figura 5, esto para que no importando el estado en el que este el modelo pueda hacer la transición al estado siguiente y así sucesivamente hasta llegar al estado de reposo que sería tener la mano abierta.[10][16]

 ISSN 2007-8102

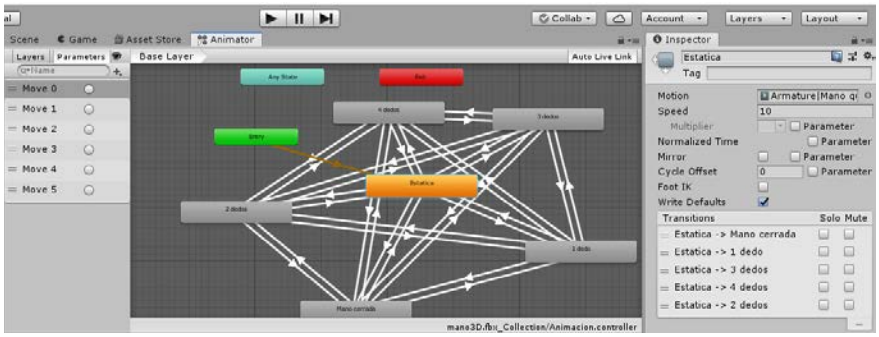


Figura 3. Mapa de transiciones para cambio de estado de modelo 3D.

## Resultados y discusión

En la primera etapa de visión computacional se logró satisfactoriamente el reconocimiento de los dedos levantados, como se muestra en la Figura 4. Claramente bajo condicione ideales o ambiente controlado, es decir iluminación y fondo, además de un determinado tono de piel, pero como ya se había dicho esto puede ser modificado según el usuario.

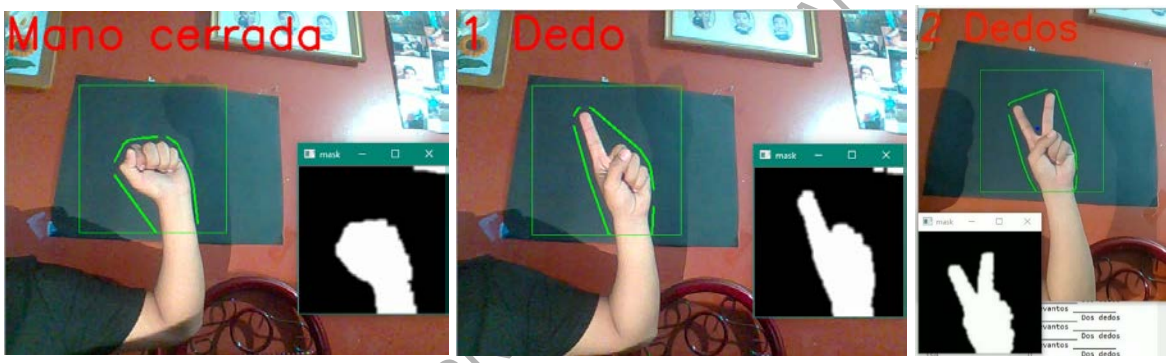


Figura 4. Captura de imagen en tiempo real y filtros aplicados a imagen.

Se logro comunicar de manera satisfactoria Python hacia C# y Unity, esto con la ayuda del protocolo de comunicación UDP y el envió de datos tipo Trigger que son paquetes de datos fiables para enviar por dicho protocolo, cabe mencionar que este enlace se realizó con el fin de mejorar la velocidad en cuanto a la comunicación. El resultado final al ejecutar los dos programas se muestra en la Figura 5, donde se muestra el algoritmo corriendo en tiempo real y Unity 3D, esto para que Python envié que dedo esta levantado y Unity 3D la reciba la cadena, y dependiendo de eso realice las animaciones echas en Blender.

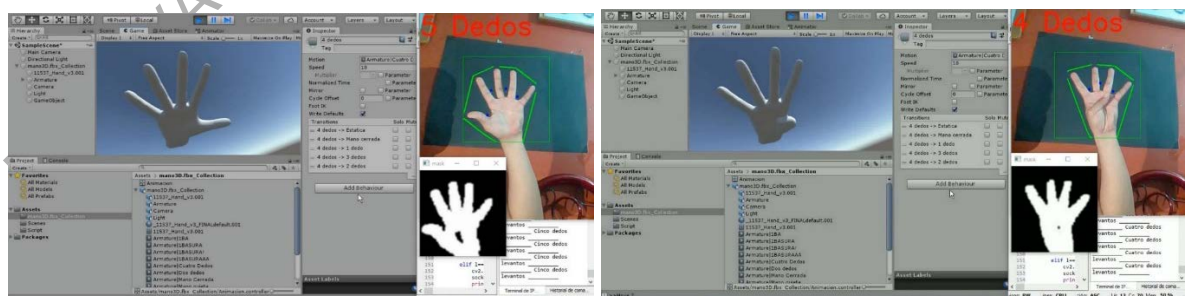


Figura 5. Resultado final del algoritmo funcionando con 5 y 4 dedos levantados.

## Trabajo a futuro

El algoritmo empleado en este proyecto cuenta con un amplio potencial de adaptación y mejora, tales como la implementación de movimientos de mano más elaborados para satisfacer otro tipo de necesidades más específicas. La idea inicial es implementar esto en rehabilitación motriz de mano, sin que sea invasivo, solo utilizando unas gafas de realidad virtual. Además de poder ser utilizado en juegos educativos.

## Conclusiones

Utilizando la visión computacional para la identificación de la mano se pudo realizar una interconexión con un ambiente virtual y tener interacción sin el uso de sensores de flexión o algún otro elemento invasivo en la mano. El algoritmo cumple el propósito inicial del reconocimiento de la mano y dedos en movimiento.

## Referencias

[1] Siddharth S. Rautaray, Anupam Agrawal, “Real Time Hand Gesture Recognition system for dynamic applications”, International Journal of UbiComp, Vol 3, No 1. January 2012

[2] Roy Sirui Yang, Anthony Lau, Yuk Hin Chan, Patrice Delmas, Christof Lutteroth, “Real Time 3D hand tracking for 3d modelling applications”, ResearchGate 2015

[3] Ali A. Abed, Sarah A. Rahman, “Python-based raspberry pi for hand gesture recognition”, International Journal of Computer Applications, September 2017

[4] M. Ali Qureshi, Abdul Aziz, “ Implementation of an efficient algorithm for human hand gesture identification”, IEEE, 2011

[5 ]Mr. D. K. Ray, P. Johri, M. Soni, A. Gupta. Hand Gesture Recognition using Python. International Journal on Future Revolution in Computer Science & Communication Engineering., Vol. 4, 59 – 62, 2018.

[6] Nayana P. B., S. Kubakaddi. Implantation of Hand Gesture Recognition Technique for HCI Using Open CV. International Journal of Recent Development in Engineering and Technology, Vol. 2, 17 – 21, 2014.

[7] Y. Gu, R. L. Grossman. Optimizing UDP-based Protocol Implementations. National Science Foundation, Vol. 1, 1 – 6, 2005.

[8] Hocine Ouhaddi, Patrick Horain, “ 3d Hand Gesture Tracking my model registration”

[9] Mohd Baqir Khan, Kavya Mishara, “Gesture Recognition using OpenCV”, International conference on communication system and network technologies, 2017.

[10 ]. A. Okita. Learning C# Programming with Unity 3D. New York: CRC press, 2015.

[11]. I. Ouazzani. Manual de creación de videojuego en Unity3D. Madrid, España: Universidad Carlos III, 2012.

[12]. R. Hess. The Essential Blender: Guide to 3D Creation with the Open Source Suite Blender. San Francisco, CA, USA: No Starch Press, 2007.

[13 ]. T. B. Patil, A. Jain, Supriya C. Sawant, D. Bhattacharyya and H. Kim. Virtual Interactive Hand Gestures Recognition System in Real Time Environment. International Journal of Database Theory and Application, Vol. 9, pag 39 – 50, 2016.

[14] T. H. Maung. Real-Time Hand Tracking and Gesture Recognition System Using Neural Networks. World Academy of Science, 26, pag. 466 – 470, 2009.

[15] Sajjad Ur Rahman, Zeenat Afroze, “Hand gesture recognition techniques for human computer interactions using OpenCV”, International Journal of Scientific and research publications, Vol. 4, December 2014

[16] Paul Smith, Thomas P. Hartley, Qasim H. Mehdi, “C # interpreter and Unity 3D for educational programming games”, The 18th international conference on computer games, 2013