ISSN 1870-4069

Traducción del lenguaje de señas usando visión por computadora

Eduardo Mancilla Morales, Oswaldo Vázquez Aparicio, Pedro Arguijo, Roberto Ángel Meléndez Armenta, Antonio Hiram Vázquez López

Tecnológico Nacional de México/Instituto Tecnológico Superior de Misantla, México
eduardomancillamorales@gmail.com, 192T0035@itsm.edu.mx, pedro_arguijo@excite.com, ramelendeza@itsm.edu.mx, jahvazquezl@itsm.edu.mx

Resumen. Se realiza el análisis del desempeño de redes neuronales multicapa y SVM como clasificador para el reconocimiento de señas del LSM considerando tres conjuntos de características: Momentos de Hu, Momentos de Zernike e Histogramas de orientación del gradiente (HOG). Se creó un dataset de 21 señas y se le aplicó un preprocesamiento a las imágenes antes de extraer el conjunto de características, las técnicas aplicadas son: reducción de ruido, escalamiento de la imagen y eliminación del fondo de la imagen. Así mismo, se analizó el efecto de la reducción de dimensionalidad del conjunto de características con los algoritmos PCA y LDA. Los resultados indican que los momentos de Zernike junto con LDA ofrece la mejor exactitud en los resultados.

Palabras clave: LSM, reconocimiento de imágenes, segmentación de piel, aprendizaje automático, momento de Hu, momento de Zernike, HOG.

Sign Language Translation Using Computer Vision

Abstract. We analyzed the perfomance of multilayer neural networks and SVM as classifier for the Mexican Sign Language (LSM) considering three diferent sets of features: Hu moments, Zernike moments and Histogram of oriented gradients (HOG). A 21-sign dataset was created and in order to extrac the features the images were pre-processed, the applied techniques were: noise reduction, image scaling, and image background removal. Likewise, we analyzed the dimensionality reduction of the set of features with PCA and LDA. The obtained results show that the moments of Zernike together with LDA offers the best accuracy in the classification using the features set.

Keywords: LSM, image recognition, skin segmentation, automatic learning, Hu moments, Zernike moments, HOG.

1. Introducción

La comunicación es fundamental para el desarrollo social del ser humano. Entre las diferentes formas de comunicación, la oral es la más común. Cuando el habla se ve impedida por algún motivo, se disminuye la capacidad de interacción social del individuo, lo que limita sus oportunidades de inclusión. Las personas con problemas del habla y auditivos se comunican utilizando señas que incluyen movimientos de las manos o cuerpo y expresiones faciales. Con este fin, en México se ha desarrollado el lenguaje de señas mexicano (LSM) cuyo alfabeto consta de 27 señas, de las cuales 21 son estáticas y las restantes dinámicas. Rautaray y Agrawal [9] mencionan que las señas estáticas se definen por la orientación y posición de la mano, durante algún tiempo sin mover ésta. Si la mano varía durante ese tiempo, se denomina gesto dinámico. El reconocimiento de gestos dinámicos requiere de secuencias de imágenes en el tiempo, debe detectarse que las secuencias tengan un patrón de movimiento, lo cual implica un conocimiento de las secuencias anteriores.

Se han tenido grandes avances en el desarrollo de sistemas de traducción automatizada de señas a texto utilizando técnicas de visión computacional. Sin embargo, dicho avance se ha dado principalmente para el lenguaje de señas americano, el japonés y el de la India. Con este fin se propuso un sistema un sistema que utiliza un sensor de movimiento para capturar imágenes de profundidad y el esqueleto de la mano humana para rastrear movimientos considerando la altura y profundidad de la mano [1]. También se ha presentado un sistema basado en visión artificial [11] que realiza el reconocimiento de veintiuna señas estáticas cuya descripción está dada por momentos centrales normalizados invariantes a transformaciones de escala y rotación. Obtienen un 93 % de exactitud con un perceptrón multicapa para la etapa de reconocimiento. El sensor Kinect se utilizó para la interpretación de palabras [12], consideran los momentos geométricos como extractor de características y modelos ocultos de Markov como clasificador para encontrar las relaciones entre la secuencia de imágenes, obteniendo tanto una sensibilidad como especificidad del 80 %. Montaño y Rodríguez-Aguilar [5] proponen un sistema para la traducción automática de lenguaje de señas mexicano a texto reportando un porcentaje de exactitud del 80 % comparando el vector de características obtenido con vectores previamente almacenados.

El presente trabajo se centra en el análisis del desempeño de redes neuronales y SVM como clasificador para el reconocimiento de señas del LSM. Se considera el enfoque basado en apariencia con una cámara como medio de captura. En la siguiente sección se describe el dataset utilizado. La solución propuesta se aborda en la sección 3. La sección 4 describe la implementación y en las secciones 5 y 6 se aborda el análisis de los resultados y las conclusiones, respectivamente.

2. Dataset de señas estáticas del LSM

En la búsqueda de un dataset para el lenguaje de señas mexicano, solamente se encontró uno pero a las imágenes de éste les faltaba variación en traslación

y escala [3], por lo cual se creó un dataset con variaciones de traslación, escalamiento y rotación. En la Fig. 1, se muestran variaciones en la posición de la mano en traslación y rotación. La traslación se logra moviendo la mano de un lado hacia otro y la rotación se obtiene al girar la mano, en la Fig. 1 se muestra con una flecha el movimiento que se realizó al capturar las imágenes. El dataset consta de 300 imágenes por cada una de las 21 señas estáticas, con las variaciones mencionadas, dando un total de 6300 imágenes. La Fig. 2, exhibe un ejemplo de las 21 señas estáticas capturadas. Todas las señas se realizaron por una sola persona. El dataset utilizado en este artículo se encuentra disponible, para las personas interesadas en su posterior análisis, en la dirección electronica: `https://1drv.ms/f/s!Ag_D60aA-1xrggXOp-G9eJrs-u3l`

Fig. 1. Imágenes de la letra A con variaciones en traslación y rotación.

3. Solución propuesta

El método propuesto en el presente trabajo incluye una etapa de preprocesamiento, una etapa de extracción de caracteristicas y una etapa de clasificación. El proceso es el siguiente:

- Creación del dataset de las 21 señas.
- Eliminar el ruido de la imagen.
- Escalar la imagen a un tamaño predeterminado.
- Segmentar la imagen por color de piel.

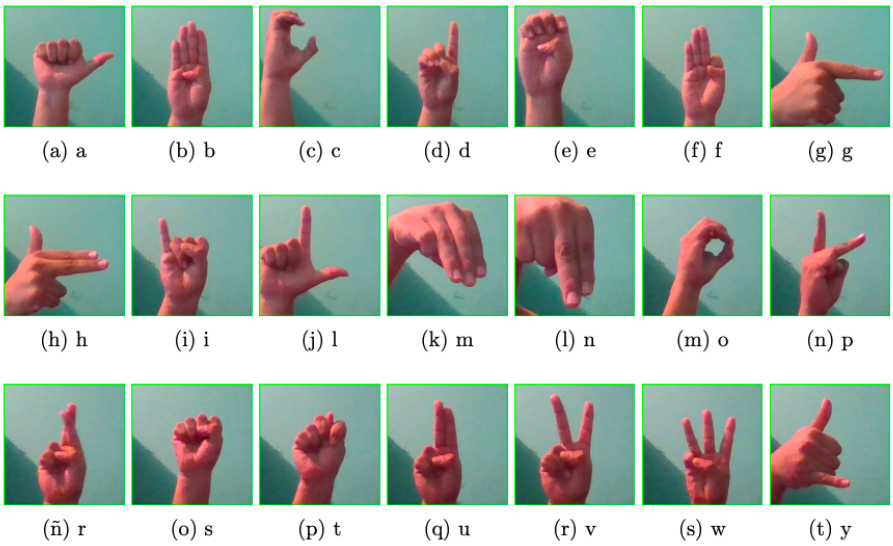

Fig. 2. Ejemplos de las imágenes de todas las señas del dataset creado.. Sólo se incluyen las letras estáticas.

- Transformar la imagen a un vector de características.
- Reducir la dimensionalidad del vector.
- Entrenar el clasificador.

3.1. Reducción de ruido

Se realizó un comparación entre los filtros promediador, gaussiano y bilateral [7] concluyendo que este último permite conservar los bordes sin una pérdida significativa de la mano.

3.2. Escalamiento de la imagen

Con la finalidad de aumentar la velocidad de procesamiento se escalan la imágenes a un tamaño de 200×200 pixeles, con lo cual se conserva suficiente información. Este tamaño de imagen da un total de 40,000 píxeles en comparación con los 777600 de la imagen original (1080×700).

3.3. Eliminación del fondo de la imagen

Se necesita eliminar el fondo de la imagen, para evitar tener información innecesaria, que puede interferir en los resultados de la detección de la mano. Consecuentemente, se utiliza la segmentación por color de piel. La cual consiste en eliminar las partes de la imagen que no tengan un color similar al color de la

piel, eliminando el fondo, en la Fig. 3 se muestra el resultado de la segmentación. Se utiliza el espacio de color YCbCr debido a que puede ser aplicado a imágenes con colores complejos y con iluminación irregular. Los valores que se consideran son [10]:

$$150 < Cr < 200 \tag{1}$$

$$100 < Cb < 150 \tag{2}$$

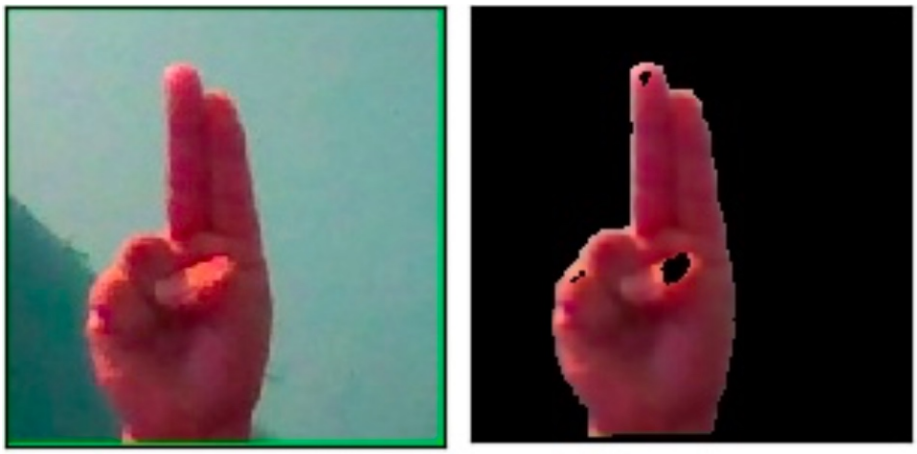

Fig. 3. Segmentación basada en el color de la piel.

3.4. Transformar la imagen a un vector de características

Se utilizaron tres algoritmos de extracción de características: momentos de Hu [6], momentos de Zernike [6] e histograma de orientación del gradiente [4].

El algoritmo de Hu da como resultado un vector de longitud igual a 7, la longitud del vector de características que produce el algoritmo de Zernike depende del grado del polinomio usado, y la longitud del vector de características para el algoritmo de Histograma de Orientación del Gradiente depende del número de píxeles por celda y el número de orientaciones seleccionadas. El algoritmo que da como salida el mayor número de características, es el Histograma de Orientación del Gradiente.

Los parámetros considerados para la obtención del vector de características correspondiente a los momentos de Zernike fueron: grado 5 y radio 80, con lo cual se obtuvieron 12 características. Para HOG se consideraron 8 orientaciones, con celdas de 16×16 píxeles, obteniendo un total de 1153 características.

Debido a la gran dimensionalidad del vector de características de HOG se emplearon técnicas de reducción a fin de lograr una mejor eficiencia del clasificador, los métodos utilizados fueron: PCA y LDA.

3.5. Entrenamiento del clasificador

Se consideraron dos clasificadores para el reconocimiento de las señas, una red neuronal multicapa [11] y SVM [8]. En ambos caso es necesario reajustar los valores mediante el escalamiento de los valores. En el presente trabajo se utiliza la estandarización para reescalar las características. El proceso de estandarización se expresa como:

$$x(i)_{std} = \frac{x(i) - \mu_x}{\sigma_x}, \tag{3}$$

donde:

- $x(i)$ es un ejemplo en particular.
- μ_x es el promedio de una columna de características.
- σ_x es la desviación estándar de la columna de características.

El presente trabajo se evalúa usando validación cruzada, porque es uno de los métodos que dan una mejor evaluación del clasificador, usando $k = 4$, debido al tamaño del dataset. Por lo tanto primero se dividen los datos en 70 % para entrenamiento y 30 % para prueba, este 70 % de entrenamiento posteriormente es dividido en k particiones y evaluado con validación cruzada [2].

Los mejores parámetros para la configuración de los clasificadores se encontraron empleando grid search, que consisten en buscar las combinaciones de los diferentes parámetros, y mediante la técnica de validación cruzada, seleccionar la mejor combinación de parámetros.

La entrada de la red neuronal depende del algoritmo de extracción de características, en el caso de los momentos de Hu, la red neuronal tiene como entrada 7 características, y para los otros algoritmos dependerá de la configuración que tengan, porque la salida depende de los parámetros de configuración del algoritmo de extracción de características. La capa de neuronas de salida de la red neuronal para todos los algoritmos será de 21 neuronas, ya que son 21 tipos de señas, que corresponden a las diferentes letras del alfabeto. El número de neuronas en las capas internas depende de los algoritmos y cambia para cada combinación de técnicas.

Para evitar el sobre-entrenamiento de la red se utilizó Dropout, que consiste en bloquear algunas neuronas, para permitir que las otras neuronas aprendan patrones independientes. La función de activación de la última capa de las neuronas tiene la función de activación softmax, debido a la clasificación del tipo multiclase. La función de activación de las demás neuronas es RELU. La función de optimización que se utiliza es Adam, utilizando decay = learning rate/número de épocas, permitiendo a la red converger más rápido, si se ajusta de la manera correcta. La métrica que se utilizan para evaluar el clasificador es exactitud.

Cabe mencionar que se debe configurar de acuerdo al problema el número de capas de neuronas ocultas, así mismo se debe establecer la cantidad de neuronas en cada capa oculta, existen varias formas de configurar la red, pero las que dan mejores resultados es la configuración escalonada, es decir poner 5 neuronas en

la primera capa, 4 en la segunda y así sucesivamente, por lo tanto se utiliza esta configuración en la búsqueda de los parámetros.

Para la máquina se soporte vectorial (SVM) se utilizó un kernel de función de base radial (RBF). Se empleó el enfoque uno contra el resto para realizar la clasificación multiclase, debido a que este clasificador está diseñado para la clasificación binaria. Los modelos SVM tienen un parámetro de costo C que controla la compensación entre permitir errores de entrenamiento y forzar márgenes estrictos. El parámetro gamma define hasta dónde llega la influencia de un sólo ejemplo de entrenamiento, con valores bajos que significan lejos y valores altos que significan cercanos. Los parámetros a buscar mediante grid search para el clasificador SVM son C y gamma.

4. Implementación

Para la implementación de las configuraciones mencionadas junto con la etapa de preprocesamiento se utilizó Python 2.7, con distribución Anaconda. Las librerías empleadas para la implementación fueron:

- Pandas: Para analisis de datos, guardar y leer archivos CSV.
- Numpy: Para operaciones con arreglos.
- OpenCV: Procesamiento de imagenes.
- Scikit-learn: Para preprocesamiento y clasificación de los datos con SVM.
- Mahotas: Para extraer características con el algoritmo de momentos de Zernike.
- Scikit-image: Para extraer características con el algoritmo de momentos de HOG.
- Keras: Para preprocesamiento y clasificación de los datos con la red neuronal.
- Matplotlib: Para visualizar las imagenes.
- Pickle: Para guardar los clasificadores despues del entrenamiento.

El equipo de cómputo utilizado fue un ordenador portátil con un procesador Intel core i5, 8gb de memoria RAM con sistema operativo macOs Mojave 10.14.

5. Análisis de resultados

Los resultados de todas las configuraciones se muestran en la Tabla 1 y Tabla 2. Como era de esperarse, el tamaño del vector de características influye en el resultado obtenido por los clasificadores. El algoritmo de Hu obtiene la menor exactitud, aún así el entrenamiento es más rápido, debido a que el vector de características es menor. El algoritmo de HOG obtiene mejores resultados que el algoritmo de Hu, sin embargo, el entrenamiento y el cálculo de las características son procedimientos muy lentos, esto se debe a que el número de características es mayor que los otros algoritmos de extracción de características. El algoritmo de Zernike obtiene una mejor exactitud en algunos casos, comparado a HOG y Hu, también tiene menor número de características, y tarda menos tiempo

de entrenamiento que HOG. Los algoritmos de reducción de dimensionalidad ayudan al algoritmo en algunos casos, pero en otros disminuyen la exactitud, en algunas situaciones se obtiene mejores resultados con PCA y en otras situaciones se obtiene mejores resultados con LDA, por este motivo es mejor evaluar la combinación completa y no sólo al clasificador. El algoritmo de clasificación SVM y la red neuronal obtienen resultados parecidos en exactitud de la clasificación, sin embargo, la red neuronal requiere un configuración mayor de sus parámetros y mayor tiempo de entrenamiento, y sólo para obtener una exactitud parecida.

Para validación cruzada se utilizó el número de particiones es 4. Primero se dividió el conjunto en una parte de entrenamiento y una de test. Para los resultados de la Tabla 1 se usó el 30 % de entrenamiento y 70 % de prueba, dando buenos resultados incluso usando más datos de prueba que de entrenamiento.

Se realizó otra prueba con en 70 % de entrenamiento y 30 % de prueba, como se esperaba al aumentar el numero de imágenes de entrenamiento, se obtuvo mejores resultados, como se muestra en la Tabla 2.

Tabla 1. Resultados de la exactitud con 30 % de entrenamiento y 70 % de prueba.

Extracción de características	Preprocesamiento de datos	Reducción de dimensionalidad	Clasificación	Exactitud
Zernike	std scaler		SVM	95.07 %
Zernike	std scaler	PCA	SVM	95.39 %
Zernike	std scaler	LDA	SVM	96.50 %
HU	std scaler		SVM	89.88 %
HU	std scaler	PCA	SVM	85.70 %
HU	std scaler	LDA	SVM	87.87 %
HOG	std scaler		SVM	89.30 %
HOG	std scaler	PCA	SVM	92.90 %
HOG	std scaler	LDA	SVM	67.97 %
Zernike	std scaler		NN	94.37 %
Zernike	std scaler	PCA	NN	94.69 %
Zernike	std scaler	LDA	NN	95.89 %
HU	std scaler		NN	91.15 %
HU	std scaler	PCA	NN	90.0 %
HU	std scaler	LDA	NN	90.3 %
HOG	std scaler		NN	88.5 %
HOG	std scaler	PCA	NN	40.5 %
HOG	std scaler	LDA	NN	71.0 %

El tiempo de procesamiento para la extracción de características fue el indicado:

- Momentos de Hu: 68.64 segundos en crear dataset.
- Histograma de orientación del gradiente: 1319.21 segundos en crear dataset.
- Momentos de Zernike: 134.29 segundos en crear dataset.

Tabla 2. Resultados de la exactitud con 70 % de entrenamiento y 30 % de prueba.

Extracción de características	Preprocesamiento de datos	Reducción de dimensionalidad	Clasificación	Exactitud
Zernike	std scaler		SVM	98.35 %
Zernike	std scaler	PCA	SVM	98.5 %
Zernike	std scaler	LDA	SVM	98.7 %
HU	std scaler		SVM	93.43 %
HU	std scaler	PCA	SVM	92.0 %
HU	std scaler	LDA	SVM	93.06 %
HOG	std scaler		SVM	95.44 %
HOG	std scaler	PCA	SVM	99.1 %
HOG	std scaler	LDA	SVM	93.3 %
Zernike	std scaler		NN	98.04 %
Zernike	std scaler	PCA	NN	97.35 %
Zernike	std scaler	LDA	NN	97.77 %
HU	std scaler		NN	92.75 %
HU	std scaler	PCA	NN	93.59 %
HU	std scaler	LDA	NN	93.17 %
HOG	std scaler		NN	94.97 %
HOG	std scaler	PCA	NN	81.37 %
HOG	std scaler	LDA	NN	91.48 %

Analizando los resultados del clasificador en conjunto con los diferentes algoritmos de clasificación y los algoritmos de reducción de dimensionalidad, se obtiene que el clasificador SVM obtiene mejores resultados que la red neuronal, incluso en un tiempo menor de entrenamiento.

La red neuronal requiere una mayor cantidad de parámetros a configurar para una mayor exactitud en el problema específico, por lo tanto requiere una mayor cantidad de combinaciones a probar, esto produce que la cantidad de tiempo requerido para entrenar una red neuronal sea mayor que el tiempo que requiere el entrenamiento de una máquina de soporte vectorial.

La combinación del algoritmo de reducción de dimensionalidad LDA junto con el algoritmo de extracción de características de los momentos de Zernike, y el clasificador SVM aumentó la exactitud aún con 30 % de los datos como entrenamiento.

Otra combinación buena es el algoritmo de extracción de características HOG junto al algoritmo de reducción de dimensionalidad PCA y el algoritmo de clasificación SVM, obtuvieron una mejor exactitud que las demás combinaciones, pero a un costo de entrenamiento y memoria mayor a las demás combinaciones de algoritmos.

6. Conclusiones

En esta investigación se propuso un sistema de reconocimiento de las señas estáticas del LSM basado en imágenes a color. Se generó una base de conocimiento (dataset) el cual consiste de 21 señas con 300 imágenes cada una, dando un total de 6300 imágenes en las cuales están representadas las señas estáticas.

Se muestra la comparación de dos clasificadores: redes neuronales y SVM junto con una base de datos que consta de 6300 imágenes en las cuales se consideraron variaciones de escala, rotación y traslación.

Se consideraron varios conjuntos de características obtenidas: Momentos de Hu, momentos de Zernike e Histogramas de orientación del gradiente. Así mismo, se consideraron LDA y PCA.

La tasa de reconocimiento para las imágenes estáticas en las condiciones descritas exhibieron una exactitud del 98.7 % considerando los momentos de Zernike y LDA para SVM con $gamma = 0{,}05$ y $C = 19$.

Los pasos para resolver la detección del alfabeto de lenguaje de señas son dependientes uno del otro. Por lo tanto, si un paso no se realiza correctamente puede afectar el rendimiento de los demás pasos. La segmentación basada en el color resulta aceptable, ya que remueve gran parte del fondo de una manera muy eficiente, sin embargo, cuando en el fondo se encuentran objetos de color parecidos a la piel, surgen problemas, para solucionar esto, se debe ajustar el rango de valor para la detección del color o usar alguna manera de diferenciar el fondo.

Se pretende que un un futuro se reconozca todo el abecedario del lenguaje de señas mexicano incluyendo los números, esto implica que se debe implementar el seguimiento de la mano y el reconocer los signos dinámicos no considerados en esta investigación.

Referencias

1. Galicia, R., Carranza, O., Jiménez, E., Rivera, G.: Mexican sign language recognition using movement sensor. In: Industrial Electronics (ISIE), 2015 IEEE 24th International Symposium on. pp. 573–578. IEEE (2015)
2. Hackeling, G.: Mastering Machine Learning with scikit-learn. Packt Publishing Ltd (2017)
3. Ibarra, E.: Generación de dataset para problema de visión computarizada (2017), `https://medium.com/inteligencia-artificial-itesm-cq/generaci%C3%B3n-de-dataset-para-problema-de-visi%C3%B3n-computarizada-a90c77a0dc9a`
4. Joshi, G., Gaur, A., et al.: Interpretation of indian sign language using optimal hog feature vector. In: International Conference on Advances in Computing and Data Sciences. pp. 65–73. Springer (2018)
5. Montaño, L.A.F., Rodríguez-Aguilar, R.M.: Automatic translation system from mexican sign language to text. Advances in Computational Linguistics p. 53 (2011)
6. Otiniano-Rodríguez, K., Cámara-Chávez, G., Menotti, D.: Hu and zernike moments for sign language recognition. In: Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV). p. 1.

The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) (2012)
7. Paris, S., Kornprobst, P., Tumblin, J., Durand, F., et al.: Bilateral filtering: Theory and applications. Foundations and Trends® in Computer Graphics and Vision 4(1), 1–73 (2009)
8. Patil, A., Subbaraman, S.: A review on vision based hand gesture recognition approach using support vector machines. Journal of Electronicsl and Communication Engineering (2012)
9. Rautaray, S.S., Agrawal, A.: Vision based hand gesture recognition for human computer interaction: a survey. Artificial Intelligence Review vol. 43(1), 1–54 (2015)
10. Shaik, K.B., Ganesan, P., Kalist, V., Sathish, B., Jenitha, J.M.M.: Comparative study of skin color detection and segmentation in hsv and ycbcr color space. Procedia Computer Science 57, 41–48 (2015)
11. Solís, F., Martínez, D., Espinoza, O.: Automatic mexican sign language recognition using normalized moments and artificial neural networks. Engineering 8(10), 733 (2016)
12. Sosa-Jiménez, C.O., Ríos-Figueroa, H.V., Rechy-Ramírez, E.J., Marin-Hernandez, A., González-Cosío, A.L.S.: Real-time mexican sign language recognition. In: Power, Electronics and Computing (ROPEC), 2017 IEEE International Autumn Meeting on. pp. 1–6. IEEE (2017)