[[1]](#footnote-1)

Aplicación para *Aumentar* en Tiempo Real Videos Existentes y en Livestream con Contenidos Multimedia

Liu Fong, Andrés David, Ingeniería Multimedia, Carrillo Acosta, Kevin Alexis, Ingeniería Multimedia y Giraldo Londoño, Leonel, Ingeniería Multimedia

*Abstract*—This paper’s primary objective was to showcase the process and operations followed for creating a simple graphic-user-interface-based augmented reality application that could overlay images, videos and sounds, and could be used by choosing to load either a pre-recorded video or a livestream from a camera. For this, *Template Matching* was used for comparing a region of interest previously chosen from a frame of the video source, with each frame of itself or the livestream video’s; and *Seamless Cloning* for positioning and placing the augmented element by creating a mask that was drawn on top of the source video frames. By applying these methods, the final functionality of the application was presented and explained together with its current limitations. The development process of this application could be further improved by implementing the recognition and tracking of multiple elements in a frame at the same time, although the cost of more processing time and resources should also be accounted for. Thinking on another approach for addressing this problem could prove for better results.

*Index Terms*—AR, audio, augmented reality, image recognition, livestream, multimedia, record, region of interest, seamless cloning, template matching, tracking, video.

# INTRODUCCIÓN

R

ealidad aumentada es, en general, un término que busca englobar los procesos y aplicaciones que intentan combinar la realidad física y una capa virtual extra con el fin de, precisamente, *aumentar* y mejorar en distintos niveles y a través de diversas maneras, nuestra visión del mundo real ya sea para fines de entretenimiento, turismo, educación y capacitación, médicos, investigación, publicidad, o para aplicaciones desarrolladas a la medida.

Comúnmente se han venido desarrollando un gran número

de aplicaciones orientadas a la detección en tiempo real de rostros, enfocándose principalmente en el campo de la computación móvil y embebida, y que han logrado desarrollar e implementar robustos algoritmos para este caso particular de tracking.

Sin embargo, para todos los campos de acción mencionados, las aplicaciones basadas en reconocimiento facial, tradicionalmente no logran salir de ámbitos relacionados a la seguridad y al entretenimiento, y requieren de una configuración o entrenamiento previos. Aplicaciones comerciales y no comerciales que puedan ser configuradas en tiempo real para detectar cualquier tipo de objeto componen un campo de investigación que aún posee potencial de exploración.

# MARCO TEÓRICO

Para el desarrollo del proyecto se empleó el lenguaje de programación Python 2.7, y la librería de OpenCV en su versión 3.0, que permite acceder a ciertas funciones no disponibles en versiones anteriores y que fueron empleadas como parte fundamental para el procesamiento. Estas operaciones se definen a continuación:

## Template matching

Es empleado como método para comparar la plantilla en la que se encuentre el objeto seleccionado contra cada uno de los frames de la fuente del video, en escala de grises. Se basa en una operación de vecindad de correlación en el que se define como ventana a la plantilla, y cuyo centro y bordes son posicionados *dentro* del frame. Como resultado de esto, los píxeles de los bordes nunca serán procesados para la imagen resultante. Posteriormente, en base a un umbral especificado por el usuario, se determina el punto con mayor intensidad como el objeto de interés.

## Seamless cloning

Es empleado como método para posicionar el elemento multimedia de imagen o video a partir de una máscara del tamaño de la región de interés seleccionada. Ésta trata de igualar las condiciones de iluminación y características propias del frame del video fuente aplicando una operación de vecindad. Finalmente, a partir del tamaño de la región de interés, posiciona el elemento en su centro por cada frame.

## TKinter

Aprovechando que este módulo se incluye por defecto con Python 2.7, es empleado como método para generar ventanas de selección de archivos.

## PyQT

Es empleado como wrapper para Python de QT, y usado para desarrollar la GUI más compleja a partir de la cual se pueden configurar los parámetros de la aplicación en tiempo real.

# ANTECEDENTES

Actualmente existen algunas aplicaciones para *aumentar* contenidos específicos de manera individual, con funcionalidades similares y enfocadas a esta área de investigación. Sin embargo, se ha encontrado que éstas están principalmente orientadas para ser empleadas a modo de tutorial y que sirvan como referente para el desarrollo de otras aplicaciones, y empleando un método de procesamiento particular; es decir, no se ha encontrado una aplicación “completa” que permita escoger tanto el elemento deseado como la fuente del video y emplee un método de procesamiento más general.

# DESARROLLO

Para el desarrollo de la aplicación se tuvieron en cuenta dos requerimientos principales al orientar la investigación del proyecto: el tener que poder seleccionar el objeto o región de interés a partir de un frame del video; y el deber realizar un seguimiento continuo para posicionar el elemento sobre ésta.

Para la captura del elemento deseado en un frame del video fuente se pensó desde el principio en tener que “cortar” la matriz del frame original, a partir de un par de puntos seleccionados por el usuario y emplear sus coordenadas espaciales para extraer las pixeles deseados, y finalmente almacenar esta nueva imagen para su posterior procesamiento. Para esta fase se emplearon eventos del teclado de tal forma que al presionar la tecla ‘i’ se copiara una referencia al frame actual de la fuente de video y se aplicara el proceso anteriormente descrito. Para esto, se empleó un script externo desarrollado por *Adrian Rosebrock* de *PyImageSearch [6].*

Durante la fase de investigación para la detección de elementos se encontraron varios métodos para identificar y realizar seguimiento a una gran variedad de objetos. Sin embargo, algunos de los procesos más comunes y con resultados más significativos se basaban en características muy generales, tales como generar un histograma de las intensidades del objeto para un seguimiento basado en color, similar a una especificación del histograma. Naturalmente, si el fondo del video contenía algún otro elemento de colores parecidos habría una detección errónea, además de ser muy costoso en términos de recursos. Por el contrario, algunos de los métodos requerían características muy específicas tales como la forma del objeto para un reconocimiento a partir de contornos, pero denuevo, si en la escena apareciera otro objeto de forma similar, se presentaría también una detección errónea, además de presentarse la necesidad de extraer los contornos del objeto deseado a partir de la región de interés seleccionada en caso de que ésta contuviera otros elementos indeseados.

Una solución a este problema pudo haber sido una combinación de ambas técnicas, sin embargo, se optó por una solución mucho más directa e incorporada a la librería empleada, pero un poco menos efectiva, como lo fue el procesamiento a partir de *Template matching*.

Una vez se resolvió el método de detección se procedió a analizar de qué forma se podría sobreponer el elemento aumentado sobre el frame en el lugar exacto en que se detectara la región de interés. En primer lugar se intentó copiar la matriz del elemento multimedia a partir del punto determinado como el de mayor coincidencia en la fase de detección y, a partir del cotorno generado por su tamaño, reemplazar los pixeles correspondientes. Este método fue probado como útil pero presentaba algunos inconvenientes al intentar sobreponer imágenes .PNG con transparencia puesto que los nuevos pixeles correspondientes a estas posiciones no se encontraban definidos.

Investigando más a fondo se encontró el nuevo método *Seamless cloning,* propio de la versión 3.0 de OpenCV, que permitía realizar la misma operación descrita anteriormente y además posicionar el elemento de manera más exacta definiendo un punto como pivote a partir del cual copiarlo. Aún más importante, éste método resolvía el problema de las imágenes con transparencia pues empleaba una operación de vecindad a partir de una máscara que fue aplicada de manera muy general alrededor del elemento multimedia y que permitía tener en cuenta los pixeles pre-existentes del frame original de la fuente de video, efectivamente asignando un valor de intensidad adecuado a los pixeles correspondientes a transparencias.

Para almacenar el nuevo video generado a partir de la sobreposición de la multimedia se definió una simple variable que iría agregando los nuevos frames que resultaran de aplicar *Seamless Cloning* para que finalmente, una vez terminara la ejecución del programa, se “cerrara” el stream de adición y se almacenara como un nuevo archivo si así se escogía. En este proceso se presentó el inconveniente de que, al parecer, OpenCV o el OS en general requería de tener instalados los códecs específicos para el formato de salida del archivo generado. Para el desarrollo de esta aplicación se empleó el formato .AVI a partir del códec XVID.

# PRUEBAS Y RESULTADOS

Como las técnicas empleadas debían ser aplicadas a imágenes individuales, inicialmente se emplearon frames particulares de un video pre-grabado. Una vez se comprobó el adecuado funcionamiento de los métodos descritos anteriormente, se procedió a aplicarlos a una secuencia de imagénes en un ciclo, entendida como un video.

Las pruebas iniciales fueron exitosas, logrando capturar una región de interés del frame seleccionado, almacenándolo en un nuevo archivo de imagen, permitiendo en primer lugar sobreponer una imagen arbitraria en esta región una vez fuera detectada. Sin embargo, inicialmente no se tomó en cuenta que el procesamiento por *Template matching* siempre retornaría un pixel con mayor similitud a la plantilla, lo cual significó que a pesar de que el elemento a detectar ya ni siquiera se encontrara en el frame actual, se seguiría sobreponiendo el elemento en la nueva mejor ubicación. Para resolver este problema se implementó una simple sentencia condicional a modo de umbral de aceptación.

Para el caso de aumentar con un video pre-existente, se extendió el mismo concepto aplicado a una imagen pero empleando un ciclo interno para reproducirlo, pero al terminar el recorrido de todos lo frames se presentaba un error debido a que el programa seguía corriendo pero ya no tendría más frames que sobreponer. Esto se resolvió detectando cuántos frames contenía el video, de tal manera que cuando el frame actual llegara a él se reiniciara a ‘0’, creando un efecto de *loop.*

Para el caso de aumentar con un sonido, se empleó el módulo nativo de sonido de *Windows, “winsound”* incorporado también con Python 2.7, para que una vez se detectara la región de interés en el frame del video fuente, reprodujera el sonido seleccionado. Se usó una variable booleana de control para determinar si volver a reproducir desde el inicio o no el sonido sin éste haber terminado de reproducirse en la anterior detección, en caso de haber ocurrido.

Una de las mayores falencias de *Template Matching* descubiertas fue la no manipulación interna de transformaciones espaciales tanto en la detección de la plantilla en el frame del video fuente, como en el mismo elemento multimedia, refiriéndose sólo a imágenes y videos. Es debido a esto que no es posible detectar elementos si éstos cambian de ángulo durante el video o son muy acercados o alejados con base al punto de referencia empleado como plantilla. La elección de un nivel de umbral correcto es por lo tanto relativo tanto a la plantilla como al video en general.

Del mismo modo, el mayor inconveniente de emplear *Seamless Cloning* es que dependiendo del tipo de método de *overlay* que se aplique, el elemento sobrepuesto tendrá cierto nivel de transparencia que puede no ser ideal para ciertos objetos que se quieran aumentar. Igualmente, al ser una operación de vecindad basada en correlación, las áreas correspondientes a transparencias en el elemento sobrepuesto presentarán un efecto de desenfoque que dependiendo de las características de ambas imágenes, determinarían si éste es ligero o no. Análogamente, el color de ambas imágenes también afectará la forma en la que éstas se mezclen, lo cual puede llegar a producir resultados muy inesperados como la desaparición total o parcial del elemento sobrepuesto.

# CONCLUSIONES Y TRABAJO FUTURO

El proceso de investigación realizado para este proyecto ha permitido presentar una aproximación inicial hacia el desarrollo de una aplicación “universal” que, aunque no muy robusta, permite detectar cualquier elemento en una escena y sobreponer elementos multimedia en él.

La importancia del desarrollo de este tipo de aplicaciones yace en la necesidad de poder crear algoritmos que permitan la detección de objetos arbitrarios y no sólo de plantillas específicas tipo *chessboard* o *códigos QR* de los cuales ya existen muchas implementaciones muy eficientes, por ejemplo.

Asimismo, emplear *Template matching* no es la mejor solución a menos que se modifique el algoritmo original para que tenga en cuenta cambios en el ángulo de comparación entre la plantilla y el frame, y poder transformar acordemente el elemento sobrepuesto; pero aún así, realizar esta operación sobre todo un frame treinta veces en un solo segundo sería muy costoso per se, e inclusive algo idílico para una computadora promedio actual, esto sin mencionar una plataforma embebida.

Sin embargo, dejando de lado el método de procesamiento, un buen ejemplo de trabajo a futuro sería implementar la detección y seguimiento de múltiples objetos junto con la sobreposición de múltiples elementos de distintos tipos de multimedia. Igualmente, podría implementarse la sobreposición de objetos 3D animados o inclusive combinar dispositivos *weareables* externos con una base de datos almacenada en un servidor web y *HUDs* virtuales completos que presentaran información relevante de cualquier tipo de objetos predefinidos y en tiempo real acerca de su estado. Es inclusive posible también pensar en interacciones entre los elementos aumentados para generar nuevas formas de comunicación no sólo entre los objetos físicos del mundo sino también con otras personas, como es ya posible observar en algunas empresas dedicadas a las industrias de los juguetes y el entretenimiento infantil con sus productos que cada vez se inclinan más hacia la electrónica.

# REFERENCIAS BIBLIOGRÁFICAS

1. OpenCV dev team. (2016, Mayo). Template Matching. *OpenCV 2.4 docs.* [Online]. Available at source: http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template\_matching/template\_matching.html
2. OpenCV dev team. (2015, Diciembre). Template Matching. *OpenCV 3.1.0 docs.* [Online]. Available at source: http://docs.opencv.org/3.1.0/d4/dc6/tutorial\_py\_template\_matching.html#gsc.tab=0
3. Múltiples autores. (2015, Diciembre). Template Matching. W*ikipedia.* [Online]. Available at source: https://en.wikipedia.org/wiki/Template\_matching
4. OpenCV dev team. (2014, Noviembre). Seamless Cloning. *OpenCV 3.0.0-dev documentation.* [Online]. Available at source: http://docs.opencv.org/3.0-beta/modules/photo/doc/cloning.html
5. Mallick. Satya. (2015, Marzo). Seamless Cloning using OpenCV (Python, C++). *Learn OpenCV.* [Online]. Available at source: http://www.learnopencv.com/seamless-cloning-using-opencv-python-cpp/
6. Rosebrock. Adrian. (2015, Marzo). Capturing mouse click events with Python and OpenCV. *Pyimagesearch.* [Online]. Available at source: http://www.pyimagesearch.com/2015/03/09/capturing-mouse-click-events-with-python-and-opencv/

1. Documento presentado el día martes 17 de mayo del 2016 para revisión en la asignatura de Procesamiento Digital de Imágenes de la Universidad Autónoma de Occidente para el período 2016-01.

   Liu Fong, Andrés David, actualmente estudia Ingeniería Multimedia en la Universidad Autónoma de Occidente, cursando séptimo semestre. (email: andres.liu@uao.edu.co.)

   Carrillo Acosta, Kevin Alexis, actualmente estudia Ingeniería Multimedia en la Universidad Autónoma de Occidente, cursando séptimo semestre. (email: kevin.carrillo@uao.edu.co.)

   Giraldo Londoño, Leonel, actualmente estudia Ingeniería Multimedia en la Universidad Autónoma de Occidente, cursando séptimo semestre. (email: leonel.giraldo@uao.edu.co.) [↑](#footnote-ref-1)