Detección de posición y decodificación del número inscrito en un TAG impreso dentro del cuadro de enfoque de una cámara, por medio del entorno SIMULINK.

Autor: Javier A. Chauvin

Abstracto: El desarrollo de este artículo tiene por objeto la decodificación y detección del número inscrito en un TAG impreso cuadrado. Para lograr la detección de posición y decodificación de dicho TAG se ha desarrollado una forma de algoritmo basado en la obtención de bordes del mismo, y la dilatación de estos además de la aplicación de filtros que permitirán tener como resultado final la decodificación y lectura del numero inscrito en un tag, además de la representación grafica del cuadro que contiene a este.

I. INTRODUCCIÓN

Hasta ahora, el desarrollo de Realidad Aumentada se ha basado en la detección de patrones específicamente establecidos, como marcas dentro de recuadros en un plano definido. Se marca el lugar donde insertar, o sobreponer la información que se aumenta a la realidad física.

Es por esto que mediante el presente artículo se explica de manera concisa en que se basa la creación de un lector de tags a través de Simulink. El siguiente diagrama muestra la lógica utilizada para el desarrollo del proyecto.

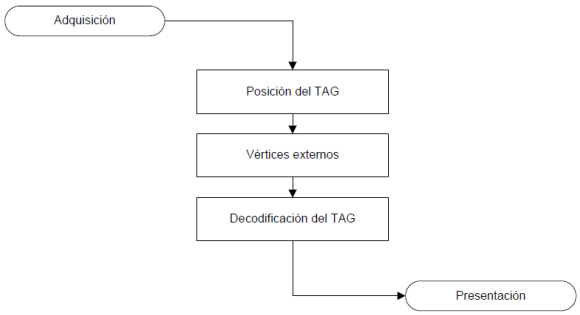

Figura.1. Algoritmo Lector de Tags

Como ilustra la Figura 1 el algoritmo general del lector de tags consta de algunos subalgoritmos, los mismos que permiten al algoritmo general cumplir algunas condiciones para su funcionamiento.

La presente investigación se originó frente a la necesidad de estar a la vanguardia de las nuevas exigencias tecnológicas y comprender cuál es la base de la que parte la realidad aumentada.

II. DESARROLLO DE ALGORITMOS

En este punto se explica el desarrollo de todos los subalgoritmos que conforman el algoritmo Lector de Tag, a lo largo de esta descripción se presentan imágenes correspondientes a cada uno de los pasos representativos de los algoritmos.

A. ADQUISICION

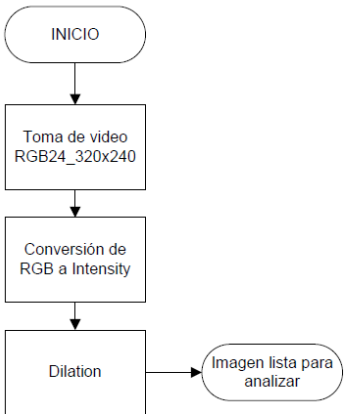

Figura.2. Diagrama de Flujo de Adquisición

Este algoritmo recibe una imagen de dimensiones de (240,320), a través de la cámara enlazada con Simulink y tiene como producto final una matriz binaria con las mismas dimensiones de la fuente (240,320). Se elige la adquisición de video RGB24_320x240, y se toma la menor resolución que permite visualizar los detalles del tag a una distancia de 1 [m] aproximadamente desde el lente de la cámara. Se determina entonces que el tamaño de las matrices con las que se trabaja es de 240 filas por 320 columnas..

Se debe realizar la conversión de RGB a intensity la cual es necesaria, porque la función de detección de bordes solo puede trabajar con imágenes de intensidad o binarias.

Después de la transformación de la imagen, la dimensión de la matriz no cambia, pero ahora contiene números flotantes en un rango de 0 a 1, debido a que la ecuación de transformación por pixel es la siguiente:

$$P_{mn} = 0.2989 * R + 0.5870 * G + 0.1140 * B \qquad (1)$$

P es el pixel analizado, m y n son sus coordenadas en filas y columnas mientras que R, G y B representan los tres colores básicos.

La Figura 3 muestra la imagen analizada después de ser transformada de RGB a imagen de intensidades.

Figura.3. Transformación a imagen de Intensidades.

Una vez realizado esto se procede a eliminar aun más la información a procesar, por tal motivo se extrae los bordes de dicha imagen.

Para extraer los bordes de la imagen obtenida se utiliza el algoritmo de Canny, en el cual se debe incluir el umbral alto y bajo de un pixel para determinar como parte de un borde fuerte o débil, es necesario recordar que los valores de la matriz analizada tienen un rango de 0 a 1, siendo 1 blanco y 0 negro. Por lo tanto se eligió 0.02 para los bordes débiles y 0.20 para los fuertes.

La Figura 4 muestra los bordes extraídos de la matriz anterior, esta es una imagen binaria. Es una matriz con dimensiones (240,320) cuyos elementos solo pueden tomar el valor de 1 o 0.

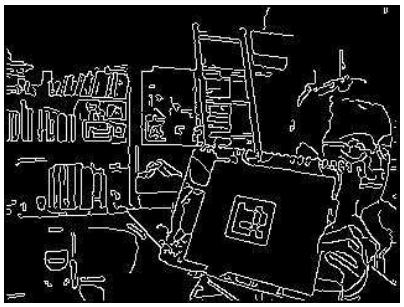

Figura.4. Detección de Bordes

A continuación se realiza la operación morfológica dilation la cual requiere de una figura geométrica que genera la dilatación de la imagen, para esta operación se eligió un disco de 5 pixeles de diámetro para la dilatación. Con esto es necesario que el tag se encuentre impreso en una hoja blanca con un borde de 0.5 veces el tamaño del tag por cada lado.

La Figura 5 ilustra la imagen digital analizada después de ser sometida a la operación morfológica dilation.

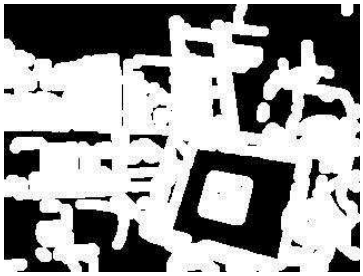

Figura.5. Dilatación de Bordes

Este algoritmo entrega al siguiente una matriz binaria de 240 filas por 320 columnas que es la representación de los bordes dilatados de la imagen que captura la cámara.

B. POSICION DEL TAG EN EL CUADRO DE ENFOQUE

Este algoritmo tiene como fin determinar en qué posición se encuentra el tag, para ello se entrega al siguiente algoritmo una matriz de intensidades con dimensiones (200,200), que contiene un segmento proporcional al tamaño del tag, tomado de la matriz de intensidades adquirida al inicio.

Para iniciar el siguiente proceso se debe recibir la matriz con la representación de bordes dilatados de realidad física, y analizar las manchas obtenidas de este proceso, una mancha es un borde dilatado que no tiene conexión con otro borde.

La función que hace el análisis de manchas trabaja solo en imágenes binarias y lo que busca específicamente es la interconexión entre los pixeles blancos, o que contienen 1 como valor. La Figura 6 ilustra cómo se sigue dicha conexión. Una vez que se tiene un perímetro formado por un borde o una mancha como tal formada por la unión de bordes, se analizan las características geométricas de la misma, es decir se hallan el alto, ancho y las coordenadas del vértice superior izquierdo de cada cuadro que contenga una mancha.

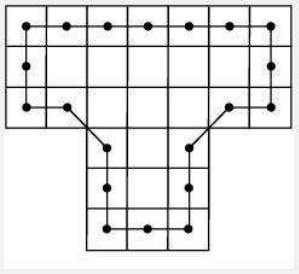

Figura.6. Unión entre pixeles

La Figura 7 muestra todos los cuadros que contienen manchas graficados en azul, en rojo es el resultado final, el cuadro que contiene el tag.

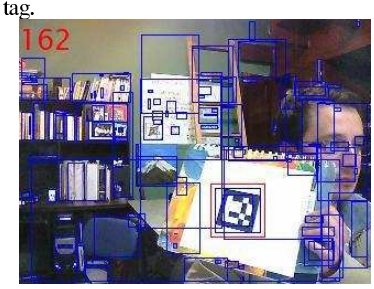

Figura.7. Recuadros de Manchas

Una vez con la información de los cuadros que contienen manchas se analizan estos segmentos de imagen de uno en uno en busca de las características del tag, esto se realiza con el propósito de

eliminar una a una las manchas obtenidas y obtener como resultado final la imagen que corresponde al tag que se va a decodificar.

Para este proceso se debe aplicar los siguientes filtros:

1. Relación entre el alto y el ancho de las manchas tiene que ser tendiente a uno porque sabemos que el tag es cuadrado.
2. El tamaño de la mancha debe ajustarse a un límite inferior y superior que permita decodificar sin problemas el número inscrito en el tag. El inferior tras ciertas pruebas de funcionamiento se determinó en 35 pixeles y el superior fue determinado en 200 pixeles, se tiene aquí las dimensiones de la matriz resultado de este subalgoritmo.
   En este punto se ha reducido el número de manchas a procesar, con el número de áreas a procesar reducido no es problema utilizar un filtro que utilice los valores mismos que trae la matriz.

3. Se encuentra la fila del medio, y de esta se obtiene la desviación estándar. La desviación estándar indica cuan variados están los elementos dentro de un universo. Como sabemos que nuestro tag tiene solo blanco y negro, entonces su variación de valores debe ser alta es decir su desviación estándar debe ser alta.
   Se obtiene la desviación estándar más alta entre todas las manchas que han sido analizadas y no eliminadas. Si el tag no está dentro del cuadro de enfoque, este algoritmo puede entregar otra mancha que supere a los filtros anteriormente expuestos, aunque tenga una desviación estándar baja, pero siendo la máxima será presentada. Por tal razón es necesario un filtro más. Un filtro que certifique una variación grande entre sus valores, que se pueda semejar al de blancos y negros. Ya que se analiza las manchas en una imagen de intensidades, los colores son susceptibles a variación por cambio en la cantidad de luz, por lo que se decidió tomar el valor de 50 como desviación estándar mínima para encontrar el tag.

C. DETERMINACION DE VERTICES

Una vez que se ha recibido del algoritmo anterior el área donde se encuentra el Tag, se ubicará las posiciones en las que se encuentra los vértices externos del tag, para lo cual primeramente se debe tomar en cuenta que la visión por computadora de Simulink trabaja con unos es por esto que se debe realizar la inversión de dicha imagen.

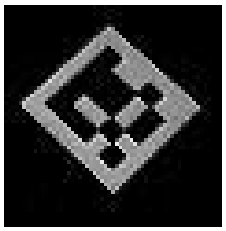

Figura.8. Imagen inversa del tag real

Luego se realiza el llenado de blancos el cual consiste en llenar toda la figura con blancos como se muestra en la figura 9.

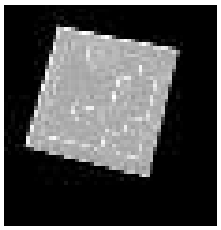

Figura.9. Llenado de blancos del tag

Gracias al llenado de blancos se puede reducir gran cantidad de vértices aumentado la posibilidad de que cada uno de los cuatro vértices a encontrar se encuentre en el espacio llenado de blancos.

Aún a través de este resultado todavía no se ha determinado los 4 vértices del tag, es por eso que para hallar estos vértices se debe considerar un principio que dice que las esquinas deben contener cómo coordenadas a los valores de filas y columnas de los extremos. Tomando estas consideraciones también se debe tener en cuenta que se puede dar el caso en el cual tengamos un vértice repetido, en este caso se debe encontrar el ultimo vértice al realizar el cálculo con los otros tres vértices obtenidos sabiendo que es una matriz cuadrada. Una vez realizado este procedimiento este algoritmo entrega al algoritmo de decodificación del tag una figura como la que se muestra en la figura 10.

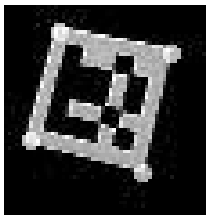

Figura.10. Vértices externos

También cabe mencionar que una vez terminado el anterior proceso finalmente se ordena a los vértices en forma horaria.

D. DECODIFICACION DEL TAG

Este es el algoritmo final y entregará como respuesta el número contenido en el tag, empieza su funcionamiento cuando recibe la posición de los 4 vértices entregados por el algoritmo anterior.

Para empezar a construir este algoritmo se tomó en cuenta que la forma más fácil de leer la información contenida en el tag es analizar a la imagen en una posición totalmente recta a la cámara ya que así se eliminará el efecto proyección producido por la inclinación del tag.

Es así que se toma los vértices obtenidos y se trabaja en base a estos es decir relaciona la posición de un punto en el trapecio y lo ubica en un cuadrado como se muestra en la figura 11.

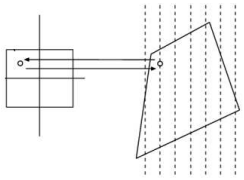

Figura.11. Cuadrilátero a rectángulo

Se ingresa un vector que contienen las coordenadas en fila y columna de los cuatro vértices y devuelve una matriz igual a la que ingresa.

Para aumentar el área en la cual se realiza la lectura de cada pixel se elige una matriz de (280,280) como respuesta a esta transformación.

Consiguiendo que cada una de las 7 columnas y filas, incluyendo el borde, sean representadas por 40 pixeles. Con esto se reduce la probabilidad que se entregue una lectura errónea provocada por una imagen borrosa. En este punto todavía se tiene una matriz de intensidades, la cual por facilidad de análisis es transformada a binaria, tal como muestra la Figura 12.

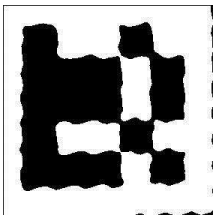

Figura.12. Eliminación de la proyección

Si el borde es eliminado de la imagen que se ilustra en la Figura 12, se obtiene básicamente una matriz de (5,5), que contiene toda la información del tag, lista para ser leída. Es necesario generar una malla como la Figura.13, con el fin de determinar el valor del pixel en ese punto.

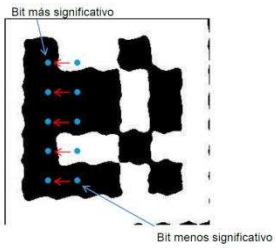

Figura.13. Lectura del tag

Con todo lo necesario para obtener el número requerido se hace la lectura de las dos primeras columnas de la izquierda. Se toma como bit menos significativo el que se encuentra en la esquina inferior derecha y más significativo en la esquina superior izquierda. Se lee de manera sucesiva hasta encontrar un número binario de 10 bits, para luego hacer la transformación a decimal.

Después de todo el proceso para la obtención del número inscrito en un tag se presenta la fusión de realidad física con la realidad virtual (número inscrito y borde del tag) en una imagen a color RGB con las mismas dimensiones que la fuente (240,320) tal como ilustra la Figura 14

Figura.14. Realidad Aumentada

III. PRUEBAS DE FUNCIONAMIENTO

Dentro del proceso de pruebas intervienen tres variables, dos físicas, el computador y el tag, y la variable virtual correspondiente al tamaño en pixeles que tiene la representación del tag en la computadora.

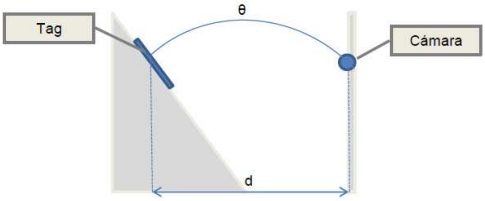

Figura.15. Representación de distribución de elementos de ensayo

A. DISTANCIA AL TAG

El objetivo de este punto es medir la distancia máxima a la cual el algoritmo es capaz de determinar la posición del tag, y si el número inscrito, es decodificado de manera correcta.

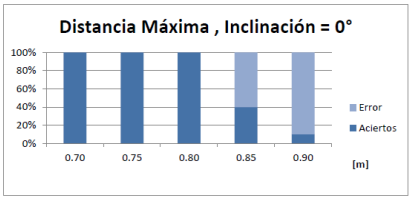

Figura.16. Distancia Máxima Inclinación 0°

La Figura 16 muestra que a 0.85 [m], el nivel de aciertos baja de 100% a 40%, algo inaceptable para el proceso, por lo tanto se decide tomar la primera distancia que logra 100% de aciertos, 0.80 [m], con dimensión del tag aproximadamente igual a 35 [pixeles].

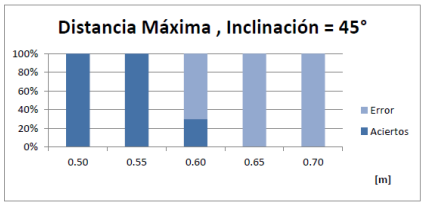

Figura.17. Distancia Máxima Inclinación 45°

Según la Figura 17 el 100% de aciertos ocurre en distancias menores a 0.55 [m].

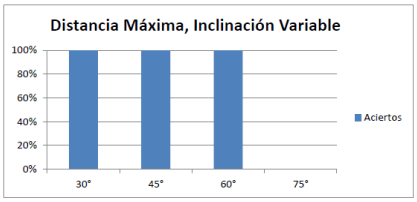

Figura.18. Distancia Máxima, Inclinación variable.

La Figura 18, ilustra que a la distancia ideal de funcionamiento el reconocimiento se encuentra entre 30º a 60º de inclinación del tag. Al ampliar el rango del ángulo de inclinación a 75º el reconocimiento es nulo.

IV. CONCLUSIONES

- ✓ Se desarrolló un algoritmo con la capacidad de encontrar la posición, y el número codificado en un tag impreso que se encuentre por completo dentro del cuadro de enfoque de la cámara utilizada. Por medio del entorno de simulación y diseño Simulink.
- ✓ Si el objeto a analizar se encuentra alejado del lente de la cámara en el momento de capturar las imágenes, todo este objeto será representado por una menor cantidad de pixeles, ósea sus detalles serán burdos. Por lo tanto en el caso en que se quiera encontrar un punto en un video, el error en la posición de dicho punto es inversamente proporcional a la distancia entre el objeto a analizar y la cámara.
- ✓ Dividir el algoritmo general en pequeños sub algoritmos, facilita la resolución de problemas. Esto ayuda al desarrollador a enfocarse en pequeños objetivos. Dando un paso a la vez se culmina cualquier proyecto.

V. BIBLIOGRAFIA

- ✓ Wagner, Daniel. HistoryOfMobileAR. icg. [En línea] [Citado el: 14 de Diciembre de 2011.] https://www.icg.tugraz.at/~daniel/HistoryOfMobileAR/.
- ✓ Integration of ADD into Matlab Simulink. [En línea] Visu itVIsual Information Technologies GmbH, 2010. [Citado el: 16 de Diciembre de 2011.] http://www.visuit.de/index.php?content=ADD/ADD_Interfaces.php?content2=ADD_Interface_Simulink.php.
- ✓ Canny Adge Detection. 2009.
- ✓ Chen, David. Shape'Based Identification of VIsual Code Markers. s.l. : Stanford University.
- ✓ Akhter, Engr. Masaud. MATLAB Setup & Tutorials. Free Civil Engineering Softwares Tutorials,Ebooks and Setups. [En línea] 1 de Abril de 2010. [Citado el: 18 de Diciembre de 2012.] http://mosttutorials.blogspot.com/2010/04/matlab-setup-tutorials.html.
- ✓ 17. MathWorks. deconvwnr . [En línea] MathWorks. [Citado el: 16 de Diciembre de 2012.] http://www.mathworks.com/help/toolbox/images/ref/deconvwnr.html.
- ✓ Product Documentation. Overview of the MATLAB Environment. [En línea] MathWorks. [Citado el: 16 de Diciembre de 2012.] http://www.mathworks.com/help/techdoc/learn_matlab/f0-14059.html.
- ✓ Vision via Matlab. TTU Advanced Robotics. [En línea] 13 de Mayo de 2010. [Citado el: 3 de Junio de 2011.] http://ttuadvancedrobotics.wikidot.com/vision-via-matlab.
- ✓ Viterbi, USC. USC Computer Vision. [En línea] University of South California. [Citado el: 18 de Diciembre de 2011.] http://iris.usc.edu/USC-Computer-Vision.html.
- ✓ Chen, David. Shape'Based Identification of VIsual Code Markers. s.l. : Stanford University.
- ✓ Elliott, Amy-Mae. 10 Amazing Augmented Reality iPhone Apps. Mashable Tech. [En línea] 5 de Diciembre de 2009. [Citado el: 10 de Diciembre de 2011.] http://mashable.com/2009/12/05/augmented-reality-iphone/.
- ✓ wikia. Computer vision. wikia. [En línea] [Citado el: 15 de Diciembre de 2011.] http://computervision.wikia.com/wiki/Main_Page.
- ✓ Vision via Matlab. TTU Advanced Robotics. [En línea] 13 de Mayo de 2010. [Citado el: 3 de Junio de 2011.] http://ttuadvancedrobotics.wikidot.com/vision-via-matlab.
- ✓ Akhter, Engr. Masaud. MATLAB Setup & Tutorials. Free Civil Engineering Softwares Tutorials,Ebooks and Setups. [En línea] 1 de Abril de 2010. [Citado el: 18 de Diciembre de 2012.] http://mosttutorials.blogspot.com/2010/04/matlab-setup-tutorials.html.
- ✓ MathWorks. deconvwnr . [En línea] MathWorks. [Citado el: 16 de Diciembre de 2012.] http://www.mathworks.com/help/toolbox/images/ref/deconvwnr.html.