[[1]](#footnote-1)

Reconocimiento de objetos e interacción con el usuario.

Patiño Ricardo, Noreña Juan Pablo.

richypati@gmail.com, juanpanore@gmail.com

Universidad de Antioquía

*Resumen*—En este artículo se pretende explicar cómo se aplicaron los diferentes conceptos que se tienen a la hora de procesar una imagen digital, con el objetivo de implementar el juego BomberHand. Utilizando el IDE Matlab, especializado para software matemático y manipular los diferentes recursos que nos provee el computador, entre los más importantes la video cámara.

*Índice de Términos—*

Mascara: Imagen monocromática, binarizada que permitirá discriminar sectores de la imagen para detectar cambios de color.

Matriz: representación de la imagen con elementos algebraicos, teniendo como referencia la cantidad de pixeles en una determinada posición de la imagen.

Adaptadores de video: Puerto en los cuales el IDE realiza la conexión con los recursos físicos que nos provee nuestros computadores.

Monocromático: denominación que se les da a las imágenes de una sola escala de color.

RGB: colores Red, Green and blue (Rojo, Verde, Azul)

Binarizado: Imagen que contiene valores de 1 o 0, es decir blanco y negro.

# INTRODUCCIÓN

## Problema humano a resolver:

Con la creciente cantidad de datos manejados hoy en día en términos de seguridad, normatividad y otros ámbitos similares, el reconocimiento de objetos es esencial a la hora de identificar ciertos factores de la vida cotidiana con el fin de lograr cierto control de la población y evitar un excesivo desgaste humano a la hora de la obtención de datos. En casos comunes como la administración de un parqueadero de alta concurrencia o la separación de productos defectuosos que pasan por una banda transportadora de alta velocidad, es necesario implementar métodos tecnológicos que permitan el desarrollo de estas actividades de una manera eficiente, teniendo en cuenta las limitaciones de las capacidades humanas.

## Problema técnico a resolver

Nuestro problema es un objeto llamativo que funcione como puntero para permitir interacción entre un usuario y una aplicación, en nuestro caso un juego. Para ello se cuenta únicamente con una cámara y el computador que va a ejecutar dicho programa.

## Experiencias

En la literatura se encuentra altamente documentado el tratamiento, los modos y por consiguiente la detección de los colores o colorimetría. En la actualidad sus usos principales son:

● Las cámaras digitales: Éstas reconstruyen la imagen digitalmente detectando los colores de la imagen real.

● Procesos industriales de clasificación: A través del color de los empaques de cierto conjunto de productos pueden implementarse métodos que junto con un robot separen y organicen los objetos.

● Dispositivos de salud: Principalmente como solución a problemas de visión como el daltonismo.

● Análisis de concentraciones químicas: Exponiendo ciertos químicos a altas temperaturas se logran ciertas pigmentaciones que son características del tipo de químico que es. En este caso la detección de color es usada para automatizar dicho proceso.

Con la información documentada es fácil implementar algoritmos que utilicen técnicas similares como creación de máscaras, comparación de imágenes, separación de capas, etc, con las cuales pueda realizarse un proceso semejante.

# Arquitectura de conceptos

## Detección de color

La cualidad principal del objeto que funcionará como puntero es que resalta con respecto a todos los demás que se verán a través de la cámara. Basados en este concepto, utilizaremos el reconocimiento de colores para identificar el objeto en la imagen y así poder realizar los procesos pertinentes con cada mascara definida.

## Algoritmo

Primero se realiza la abstracción de valores de la imagen representada en una matriz de n filas, m columnas y 3 capas.

Estas 3 capas representan el nivel color de la imagen, la primera capa representa el color rojo, la segunda el color verde y la tercera el azul.

Para extraer el color rojo de la imagen se extrae la capa roja de la matriz, además se monocronomatiza. A esta capa se le aplica una función (medfilt2), el cual realiza un median filtering que nos permite disminuir el ruido, es decir nos permite deshacernos de los valores detectados de rojo no relevantes en la imagen.

Segundo binarizamos la capa definiendo un umbral que permite obtener los valores más representativos y descartar valores que no son de nuestro interés. Esto lo que hace es para que ciertos valores mayores del umbral queden con valores 1 para el blanco, y cero para el negro.

Luego se realiza un filtro sobre la imagen binarizada usando la función bwareaopen que permite descartar sectores conectados dentro de la imagen que sean menores a n pixeles.

Para nuestro objetivo tendremos 4 mascaras las cuales nos permitirá identificar el posición del objeto en la imagen. Cada mascara está representada es un mapa de bits binarizado, es decir solo tiene colores negro y blanco, esta imagen en una matriz, está representada por ceros y unos.

Con las máscaras y la imagen filtrada podemos hacer la fusión de estas dos. Con cada fusión se puede encontrar de correspondiente podemos conocer en qué posición hubo detección de color rojo, es decir si la suma de los valores de la matriz obtenida de la fusión es mayor a cero quiere decir que existe un color rojo en la posición correspondiente a la máscara.

fsi= Fusión de la máscara superior izquierda de la imagen con la imagen tomada.

fsd= Fusión de la máscara superior derecha de la imagen con la imagen tomada.

fii= Fusión de la máscara inferior izquierda de la imagen con la imagen tomada.

fid= Fusión de la máscara inferior derecha de la imagen con la imagen tomada.

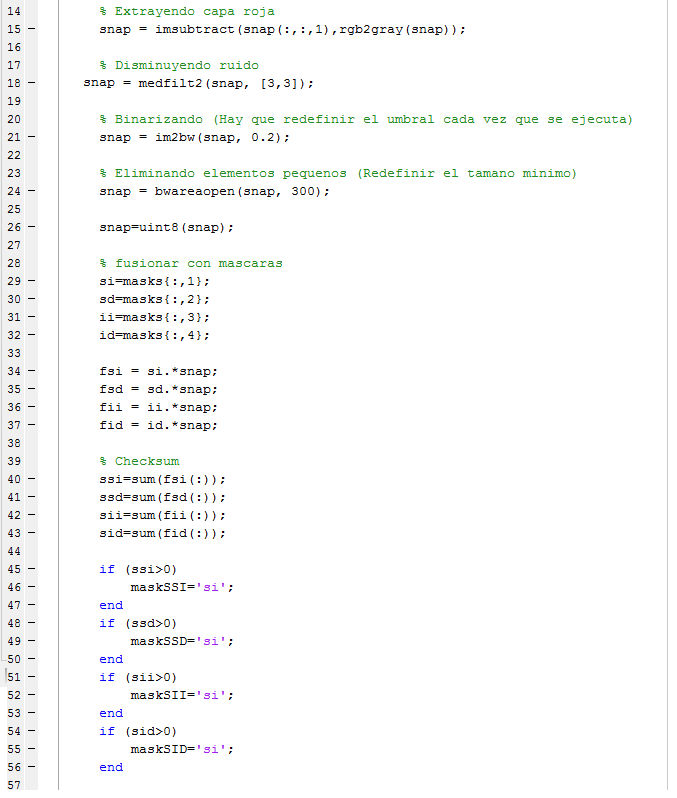
snap= Es la imagen procesada.

Para realizar la fusión de la máscara con la imagen, se realiza el producto término a término entre las dos matrices correspondientes. Esto es, como la máscara y la imagen se encuentran binarizadas, solo existen valores de 0 y 1 en las matrices, por lo tanto, al hacer la fusión de esta forma se asegura que se va a obtener en la posición correspondiente a cada máscara, los valores en 1 si detectó color rojo y en el resto 0.

De esta forma obtenemos los valores y las condiciones necesarias para implementar el algoritmo que detecte el color rojo en alguna de las esquinas (según las máscaras) donde se encuentran ubicadas las bombas y los globos.

## Esquemático o gráfico

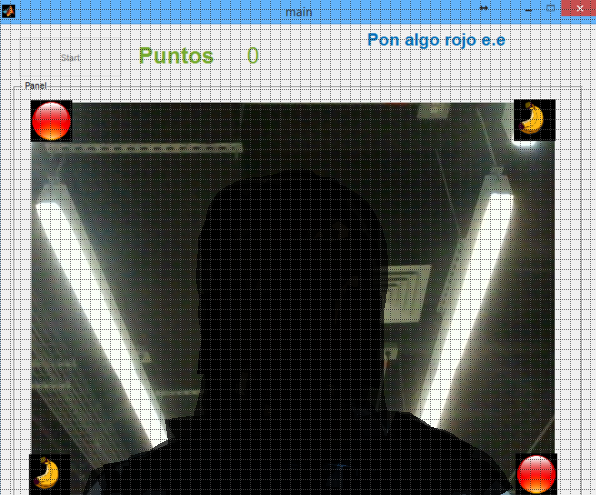
A continuación se presentan las instrucciones más representativas del juego implementadas en MATLAB, que realizan el detectado de color rojo en la imagen según la máscara correspondiente:



# Resultados y líneas futuras

## Resultados obtenidos y su análisis

Corriendo el juego



FingerBomb 1.0

En la imagen podemos visualizar los diferentes elementos que componen nuestro juego. Las bombas representadas por una imagen de color amarillo, los globos con el color rojo. Además de esto tenemos 1 botón de start el cual da inicio al juego, el puntaje total del juego y un label donde se están visualizando textos referentes al flujo del juego, es decir si nos muestra un mensaje dependiendo si acertamos o no con la posición de un globo o de una bomba.

## Líneas futuras.

La detección de color es base para distintos procesamientos y detecciones a realizar sobre una imagen, por ejemplo se puede hacer extracción de color para así crear una máscara y obtener elementos con colores específicos; se podría observar predominancia de color de una imagen, o ya realizar seguimientos de objetos por los colores que tiene.

# Conclusiones

Para detectar mejor el movimiento se quiere obtener un fondo en el cual los colores sean lo menos cambiantes posibles, para brindar algoritmo una mejor nivel de precisión de detectar los cambios que se requieren.

Se identificó que detectar el movimiento con los cambios de colores sin identificar el fondo era muy difícil detectar el movimiento, lo que comprometía el funcionamiento correcto del juego. Por esta razón se decide optar por identificar el color rojo en el fondo, para asegurar el flujo normal del juego.

El IDE Matlab nos provee una gran cantidad de ayuda en implementaciones que nos facilita implementar algoritmos como el presentado en el artículo.

referencias

[1] C. Gonzalez. Rafael and E. Wood. Richard, “Digital Image Processing” Prentice Hall

Upper Saddle River, New Jersey 07458. Capítulo 2, 1994, ed. 2.

[2] C. Russ. John, “The Image Processing Handbook” North Carolina State University

Materials Science and Engineering Department

Raleigh, North Carolina. Capítulo 1, 1994, ver. 6.

[3] The MathWorks, Inc. “Analyzing and Enhancing Images” Image Processing Toolbox™User's Guide, 2014.

1. [↑](#footnote-ref-1)