[[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 FingerBomb. 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 opaca o traslúcida que sirve para darnos referencia sobre los colores de otras imágenes

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 un solo color.

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

# 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

## Concepto a utilizar en prosa

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 la posición del objeto en la imagen y así poder realizar los procesos pertinentes para identificar el objeto en la posición correspondiente.

## Representación matemática

Para nuestro objetivo tendremos 4 mascaras que las cuales nos permitirá identificar el nivel de color en una posición correspondiente de la imagen, de la siguiente manera,

si= Resolución mascara superior izquierda de la imagen.

sd= Resolución superior derecha de la imagen.

ii= Resolucin inferior izquierda de la imagen.

id= Resolución inferior derecha de la imagen.

snap= es el resultado binario de la extracción de la capa roja de la imagen

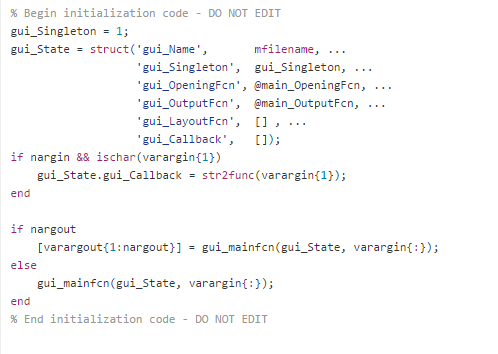
De esta manera para las demás esquinas.

Así la Sumatoria de la resolución de la imagen nos daría el cambio total en pixeles que sufre la imagen después de los snapshots, de la siguiente forma:

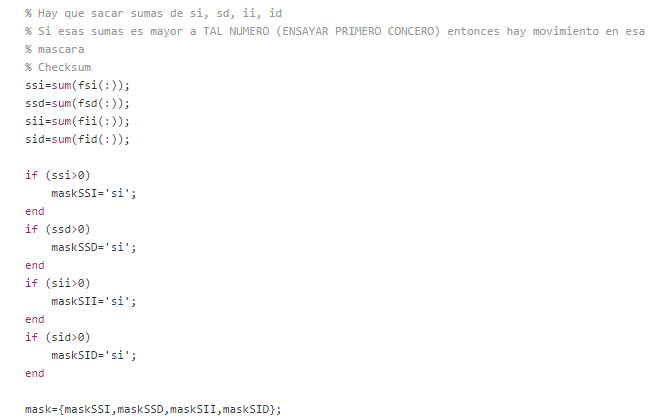
De esta forma obtenemos los valores necesarios para implementar el algoritmo que detecte el color rojo en alguna de las esquinas donde se encuentran ubicadas las bombas y los globos.

## Esquemático o gráfico

A continuación se presentan las funciones más representativas del juego implementadas en matlab



1. GUI fingerBomb



1. 1.1 Obtener Mascaras



3.0 Obtener Datos Camara

## Algoritmo

En fingerBomb, clase principal cual será el orquestador de las funciones que componen el juego, se tiene las siguientes funciones:

getCam():Función con la cual se obtiene los activa la cámara

obtenerMasks(); con la cual se obtiene los datos de las máscaras, cuatro en total, una por cada bomba o globo que se posicionará en la interfaz gráfica.

ObjetosEnMask(): con la cual se dibujan los respectivos en las respectivas posiciones los objetes del juego, es decir las bombas y los globos.

Luego tendremos un ciclo en el cual nos permitirá obtener snapshots, los cuales analizando con la función detectaMovientoEnMask(), detectaremos los cambios de color en las imágenes, permitiéndonos así identificar donde se está realizando el movimiento.

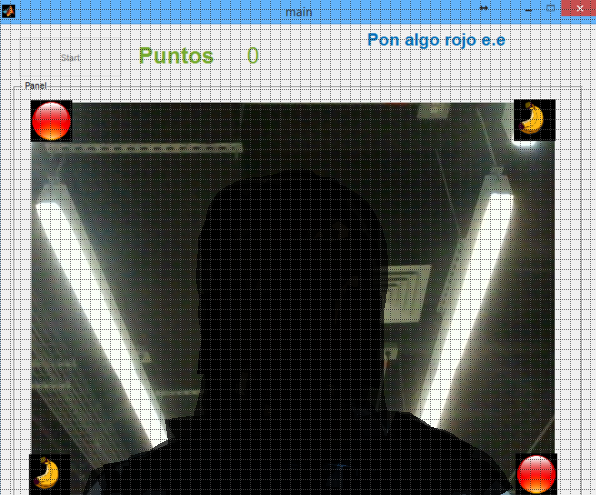
Una vez conocemos el vector RGB del snapshops, comparamos con la posición de las imágenes de la bomba, de tal manera que si en esta posición hay un cambio de color, podemos concluir que existe en un movimiento en la posición que se encuentra algunas de las imágenes de las bombas o globos.

Cuando se detecta que si corresponde el movimiento a la ubicación donde se encuentra las imágenes se llama la función de sumarPuntos(), la cual suma un punto positivo si el movimiento se detecta en la posición de un globo, si es una bomba retorna un -1, con cual determinamos el fin del juego.

# 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)