INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

Modelación de la ingeniería con matemática computacional (Gpo 1)

Prototipo de Alarma

HUPROSY - Human Protection System

PROFESOR (es):

Andrés Abundis Serrano

Isaac Juárez Acosta

Realizó:

David Balleza Ayala | A01771556 | A01771556@tec.mx

Rodrigo Antonio Benítez De La Portilla | A01771433 | A01771433@tec.mx

Jesus David Depardon Jasso | A01369816 | A01369816@tec.mx

Diego Alejandro Montelongo García | A01772149 | A01772149@tec.mx

Carlos Iván Fonseca Mondragón | A01771689 | A01771689@tec.mx

Fecha de entrega:

13 marzo, 2023

Resumen (Abstract):

En el siguiente documento hablaremos acerca de la implementación de una cámara de seguridad en áreas restringidas o de peligro en fábricas para de esta manera evitar accidentes que pueden llegar a ser mortales. Hablaremos sobre el programa MATLAB en donde realizamos la programación de la cámara y a su vez hablaremos del funcionamiento del programa mediante matrices para que la cámara de seguridad funcione de manera adecuada, mediante el procesamiento de imágenes. Tocaremos temas acerca de cómo se realizó el código del sistema de detección de intrusos y se realizará una muestra de los resultados obtenidos y una conclusión acerca del proyecto.

In the following document we will talk about the implementation of a safety camara on restricted or dangerous areas in factories in order to avoid accidents that can be fatal. We will talk about the MATLAB program where we do the programming of the camara, and we are also going to talk about the program operations by means of matrices so that the security camara works properly, by means of image processing. We will show the topics about how the code was made for the intrusion detection system and an analysis of the results that we obtained, with a conclusion about all the project.

Introducción:

La tecnología ha sido fundamental para resolver distintos problemas de la actualidad, pero en este reporte nos enfocaremos en una alarma que funciona mediante el procesamiento de imágenes, este tipo de tecnología ha sido muy útil, ya que nos ha beneficiado como individuos o comunidades en aspectos de seguridad, identificar individuos o para monitorear áreas restringidas.

En este reporte se mostrará el cómo se creó y el funcionamiento de una alarma de seguridad que brinda una alerta al acercarse a una zona peligrosa, esto se enfoca principalmente en fábricas, ya que es muy importante el resguardar la seguridad de los trabajadores de las mismas, para que estos no sufran de lesiones, heridas o amputaciones por maquinaria peligrosa, por lo que una alarma que avisé cuando alguien entre a una zona peligrosa es de mucha utilidad.

Para poder realizar esto es necesario conocer varios conceptos como lo son las matrices,***"****Las matrices son un* ***conjunto bidimensional de números o símbolos*** *distribuidos de forma rectangular, en líneas verticales y horizontales, de manera que* ***sus***

elementos se organizan en filas y columnas. *Sirven para* ***describir sistemas de ecuaciones lineales o diferenciales,*** *así como para representar una aplicación lineal. "(Ferrovial, 2022)*, este concepto es útil para realizar la alarma, debido a que las imágenes son vistas como un conjunto de pixeles, en las cuales cada pixel tiene asignado un valor, si la imagen es a color se utilizan los colores rojo, verde y azul, y dependiendo de los valores se le da el color al pixel, mientras que por otra parte si la imagen se encuentra en escala de grises el valor del pixel indica que tan brillante u obscuro es el pixel.

Otros conceptos útiles para la realización de programas en general son las tablas de verdad, las cuales muestran las condiciones que debe tener una proposición para que esta sea verdad mediante los elementos que componen a la proposición.

De igual manera se mostrará la creación de una interfaz gráfica la cual facilitará la información que el programa va a brindar sobre la alarma.

Sobre las fábricas y los accidentes:

La implementación de cámaras que alerten cuando un humano se encuentra en una zona restringida o peligrosa de una fábrica puede ser una medida efectiva para prevenir tanto accidentes mortales como no mortales. Según la Organización Internacional del Trabajo (OIT), cada año se producen alrededor de 2,3 millones de muertes relacionadas con el trabajo y se registran alrededor de 317 millones de accidentes laborales no mortales.

En este sentido, la utilización de cámaras que alerten sobre la presencia de un trabajador en una zona restringida puede contribuir significativamente a reducir estos números. Según un estudio de la Universidad Politécnica de Valencia en España, la implementación de sistemas de alerta temprana en la industria puede reducir los accidentes laborales hasta en un 50%. Además, estas cámaras pueden ayudar a identificar las causas de los accidentes y prevenir futuros incidentes al proporcionar una evidencia clara de lo que sucedió.

Métodos:

Para la realización de la alarma se utilizó el software de MATLAB, MATLAB significa laboratorio de matrices, por lo que este programa funciona principalmente con arreglos de matrices.

Para el funcionamiento de la alarma primero es necesario hacer que una cámara brinde las imágenes al programa, por lo que primero se tomará una imagen de referencia del lugar que se necesita monitorear y después se tomarán más imágenes para comparar si las personas pasan del espacio de seguridad, estas imágenes son representadas en Matlab como un arreglo de una matriz.

MATLAB App Designer es una herramienta de desarrollo de aplicaciones gráficas que se utiliza para crear interfaces de usuario personalizadas en MATLAB. Con App

Designer, los desarrolladores pueden diseñar y construir aplicaciones interactivas y amigables para el usuario en un entorno de desarrollo integrado.

El Human Protection System se creó utilizando MATLAB App Designer para detectar la presencia de humanos en zonas restringidas de una fábrica y alertar al personal de seguridad en caso de que se produzca una intrusión. El sistema funciona de la siguiente manera: primero, se agrega una imagen de referencia de la zona restringida a la aplicación. Luego, se activa la cámara que captura imágenes en tiempo real y las compara con la imagen de referencia.

Si se detecta un cambio notable entre la imagen de referencia y la imagen capturada en tiempo real, la aplicación emite una alarma y alerta al personal de seguridad a través de la interfaz de usuario. La aplicación utiliza técnicas de procesamiento de imágenes para comparar las imágenes en tiempo real y detectar cualquier cambio en la zona restringida.

Para desarrollar la aplicación, se utilizaron técnicas de procesamiento de imágenes y programación en MATLAB junto con su herramienta App Designer. El proceso de desarrollo incluyó el diseño y creación de la interfaz de usuario, la implementación de algoritmos de procesamiento de imágenes para comparar las imágenes en tiempo real con la imagen de referencia y la programación de alertas de seguridad en caso de detección de intrusos.

Código de la Interfaz (UI):

La aplicación consiste en una interfaz dividida en dos paneles, uno para la selección de la imagen de referencia y otro para el monitoreo en tiempo real. La aplicación se divide en componentes: botones, figuras, etiquetas. Algunos de estos tienen interactividad, misma que se programa en funciones *callback*. Se recitan algunos componentes importantes:

- *UIFigure*: el componente padre de toda la interfaz de usuario
- *TomaFotodeReferenciaButton*: un botón para tomar una foto de referencia
- *Image*: un elemento de imagen que muestra la foto de referencia tomada o, cuando se hace clic sobre él, permite seleccionar una imagen del almacenamiento local.
- *EmpezarButton*: un botón para iniciar el monitoreo
- *UIAxes*: un eje para mostrar las imágenes del monitoreo en tiempo real
- *AlertaLabel*: una etiqueta para mostrar una alerta si se detecta un intruso
- *LimpiarButton*: un botón para regresar al estado inicial del sistema

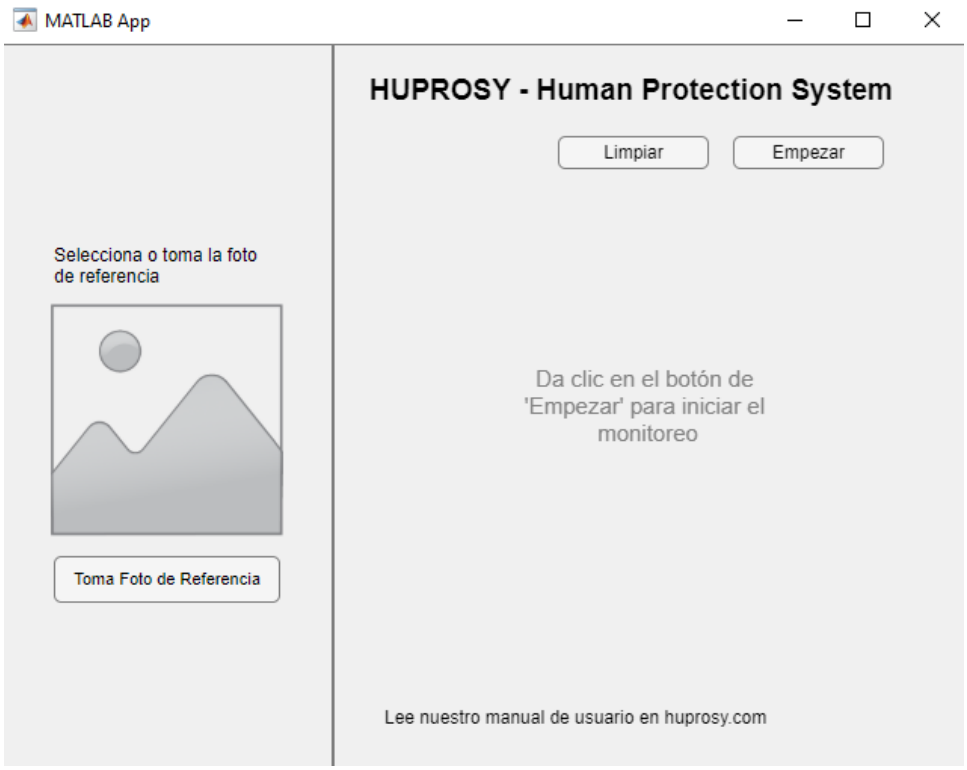

Figura 1. Interfaz gráfica de Huprosy

También se cuenta con un archivo estático de audio *"alarm.wav",* el cual contiene una pista de audio que suena cuando se detecta un intruso. Primero, utiliza la función *audioread()* para leer un archivo nuestro archivo de audio y almacenar los datos de audio en la variable *a* y la frecuencia de muestreo en *fs*. Luego, utiliza la función *audioplayer()* para crear un objeto de reproductor de audio *p* que puede reproducir los datos de audio almacenados en *a* con una frecuencia de muestreo *fs*.

Si se detecta un intruso, se reproduce el archivo de audio alarm.wav utilizando el objeto de reproductor de audio *p.* Si no se detecta ningún intruso, se detiene la reproducción del archivo de audio alarm.wav utilizando la función *stop()* en el objeto de reproductor de audio *p*.

Código de detección de intrusos:

La función *Segmentation_Fn* es una función que se utiliza para realizar la segmentación de una imagen actual en comparación con una imagen de referencia. Esta función se llama dentro de la función *EmpezarButtonPushed* cada vez que se captura una nueva imagen de la cámara.

La función recibe dos imágenes como entrada: la imagen actual capturada por la cámara y la imagen de referencia, y devuelve dos salidas: una imagen resaltada con la ubicación del objeto detectado en rojo y un valor de alerta booleano que indica si se ha detectado un objeto o no.

Para poder comparar las imágenes de una mejor manera primero hay que convertirlas en escala de grises, para que sea más fácil de encontrar los contrastes, esto es posible al crear 2 nuevas variables que representen a las imágenes y utilizar la función rgb2gray que de acuerdo con MathWorks(s.f) "*La función rgb2gray convierte imágenes RGB a escala de grises mediante la eliminación de la información de tonalidad y saturación y la retención de la luminancia"*, por lo que las imágenes nuevas tienen menos parámetros que controlar y es son más sencillo encontrar cambios en estas, por lo que con imgdiff obtenemos las diferencia entre 2 imágenes para poder obtener la diferencia entre las 2 imágenes.

```
img1BW = rgb2gray(img1);

img2BW = rgb2gray(img2);

imgDiff = abs(img2BW - img1BW);
```

Posteriormente se hace un filtro relacionado con el threshold, de acuerdo con mathworks(2022) "*a partir de la imagen de escala de grises y empleando el método de Otsu [1]. El método de Otsu elige un umbral que minimiza la varianza interclase de los píxeles blancos y negros pasados por el umbral. El umbral global T se puede usar con imbinarize para convertir una imagen en escala de grises en una imagen binaria."*, por lo que este filtro establece los parámetros que tiene que tener la imagen para que su área se resalte, por lo que se obtiene una imagen binaria, en este caso todo lo que coincide con la imagen original se queda en negro y lo que cambia se muestra en blanco.

```
imgThresh = imgDiff > 8;
```

Posteriormente se añade otro filtro, el cual es bwareaopen, esta función elimina los componentes que se encuentran unidos con una cantidad determinada de pixeles, para que la cámara no presente alarmas ante cualquier pequeño cambio que se presente en las imágenes, por lo que si un pequeño objeto o cosa se mueve por el ambiente la alarma no sonara.

```
imgFilled = bwareaopen(imgThresh, 15);
```

Después, se combina la imagen actual con la máscara binaria utilizando la función *imoverlay* para resaltar el objeto detectado en rojo y superponerlo en la imagen de referencia proporcionada por el usuario. Esta imagen será la imagen resaltada que devolverá la función.

```
imgBoth = imoverlay(img2,imgFilled,[1 0 0]);

highlighted_img = imgBoth;
```

La función *regionprops* se utiliza para calcular las propiedades de los objetos en la imagen binaria, en este caso, la longitud del eje mayor de cada objeto. La función filtra los objetos con una longitud del eje mayor inferior a 100 píxeles y devuelve un valor de alerta booleano que indica si se ha detectado un objeto o no.

```
imageStats = regionprops(imgFilled, 'MajorAxisLength');
```

En caso de que los requisitos se cumplen con la función imoverlay se resaltarán los cambios que se presentan con respecto a la imagen seleccionada o tomada previamente con respecto a la imagen actual.

```
imgBoth = imoverlay(img2,imgFilled,[1 0 0]);

highlighted_img = imgBoth;
```

El último filtro revisa si la anchura es mayor a 100 para poder resaltar el cambio y que este se muestre en rojo, pero para que esto sucede los 2 filtros pasados debieron de haberse activado igualmente.

```
imgLengths = [imageStats.MajorAxisLength];

idx = imgLengths > 100;

imageStatsFinal = imageStats(idx);
```

El último aspecto del código hace que al mostrar el cambio en la imagen activa la alarma, lo que hace que se active un audio que indica que alguien se encuentra en peligro, para que todos estén alertas o para qué la persona que entre en la zona peligrosa se aleje de inmediato.

```
                if ~isempty(imageStatsFinal)

                    alert = true;

                else

                    alert = false;

                end

            end

        end
```

Finalmente se utilizó la interfaz de usuario, que se explicó previamente para que facilitase a los operadores utilizar esta herramienta, la interfaz de usuario tiene una ventana la cual te permite tomar una foto de referencia o seleccionar una de los archivos, además cuenta con un botón que te ayuda a empezar el monitoreo, para cuando las maquias están en uso por personal preparado, pero si no es el caso, al presionar empezar la cámara comienza a monitorear y en caso de que alguna persona se encuentre en el cuarto comenzara a sonar un audio de alerta, debido a esto el sistema se llama HUPROSY, el cual significa “Human Proteccion System”.

Para la elaboración del sistema de detección se utilizó como referencia un codigo realizado por Mathwork el cual es *“Introduction to MATLAB with Image Processing Toolbox – Video"*, (2014).

Resultados y Discusión:

Después de probar extensamente nuestra herramienta, se logró crear un programa capaz de detectar cambios con alta precisión dentro de un entorno.

Para poder entender las situaciones en las que se activa la alarma del sistema de seguridad, se implementó una tabla de verdad, que detalla las instancias en las que se activa una alerta dentro del programa, que se muestra a continuación

P	Q	R	P^Q^R
V	V	V	V
V	V	F	F
V	F	V	F
V	F	F	F
F	V	V	F
F	V	F	F
F	F	V	F
F	F	F	F

Tabla 1:Condiciones para que se active la alarma

Las variables de la tabla de verdad representan lo siguiente:

P: La diferencia de la imagen imgThresh
Q: La longitud de los pixeles unidos debe ser mayor a 15
R: La anchura debe ser mayor a 100

La tabla de verdad indica que la alarma solo se activara en un escenario, en ese escenario todos los requisitos de los filtros debieron cumplirse para que la alarma sonase, por lo que logicamente es el siguiente caso P^Q^R.

Dentro de las pruebas realizadas encontramos posibles áreas de mejora para las próximas iteraciones de la herramienta, tales como la implementación de un manual de usuario, para facilitar y extender su uso, para que no solo sea utilizado por estudiantes del Tecnológico de Monterrey, pero posiblemente en versiones futuras pueda ser usado por cualquier persona, así mismo, una falla que se busca remediar lo más pronto posible es una falla en la detección de ciertos colores, o con un cierto nivel de transparencia, esto sucede porque el valor que da los colores de los cambios en la imagen presentan valores similares al pixel de fondo, por lo que en estos casos, es posible que la alarma no se accione como debería.

Conclusiones:

Con la investigación y desarrollo previo podemos concluir que las operaciones matriciales aportan muchos datos a la hora de programar, como es nuestro caso, que desarrollamos una alarma de intrusión, ya que, con esto, pretendemos por medio de una diferencia de matrices, observar los diferentes píxeles que existen entre una matriz A y una matriz B para así darnos cuenta lo que es diferente a nuestra imagen de referencia y así poder decir con certeza si hay un intruso en el entorno de la foto tomada.

Además, es importante tener en cuenta que la efectividad del sistema de alarma también depende de otros factores, como la ubicación, factores que la alteren como la luz y la rapidez con la que se puede responder a una alerta. Por lo tanto, es importante realizar una evaluación cuidadosa de los requisitos y limitaciones del proyecto antes de seleccionar los métodos y herramientas adecuados para el diseño del sistema de alarma de intrusión.

De igual manera es muy importante la realización de tablas de verdad a la hora de analizar los códigos de los programas, ya que estas nos permiten comprender de una mejor manera bajo que circunstancias el programa va a funcionar y en que otras no.

Referencias:

World Health Organization. (2021). WHO/ILO joint estimates of the work-related burden of disease and injury, 2000–2016: global monitoring report. Link: https://apps.who.int/iris/bitstream/handle/10665/345242/9789240034945-eng.pdf?sequence=1

Qué son las matrices, conceptos asociados, tipos y aplicación - Ferrovial. (2022, November 2). recuperado 13 de marzo 2023, de Ferrovial sitio web: https://www.ferrovial.com/es/stem/matrices/

Convertir una imagen RGB en una imagen en escala de grises. (2022). recuperado 13 de marzo 2023, de Mathworks.com sitio web: https://la.mathworks.com/help/matlab/ref/rgb2gray.html

Convertir imágenes de intensidad en imágenes binarias empleando el umbral de nivel. (2022). recuperado 13 de marzo 2023, de Mathworks.com sitio web: https://la.mathworks.com/help/images/ref/graythresh.html

Introduction to MATLAB with Image Processing Toolbox - Video. (2014). Retrieved March 14, 2023, from Mathworks.com website: https://la.mathworks.com/videos/introduction-to-matlab-with-image-processing-toolbox-90409.html