UNIVERSIDAD DE TARAPACÁ

FACULTAD DE INGENIERÍA

Departamento de Ingeniería en Computación e Informática

INFORME DE PROYECTO

“Manejo de daltonismo dicromático mediante esquema de colores RGB”

Autor(es): Diego Berríos

Gustavo Olivares

Kevin Rodríguez

Asignatura: Proyecto II

Profesor(es): Diego Aracena Pizarro

ARICA, 07 de enero 2021

Fecha	Versión	Descripción	Autor(es)
31/10/2020	1.0	Versión preliminar del formato	Gustavo Olivares Diego Berríos
03/10/2020	1.1	Versión final del primer informe de avance	Gustavo Olivares
23/12/2020	1.2	Versión final del segundo informe de avance	Gustavo Olivares
03/01/2021	1.3	Versión preliminar del informe final	Gustavo Olivares Diego Berríos Kevin Rodríguez
07/01/2021	1.4	Versión final del informe final	Gustavo Olivares

Índice

1. Panorama general

1.1 Resumen del Proyecto

- **Introducción**:

En el presente informe se muestra la planificación que se está llevando a cabo para la realización de nuestro proyecto, el cual está realizado en lenguaje Python utilizando una librería OpenCV para realizar una aplicación de asistencia cognitiva principalmente visual, siendo en nuestro caso específico centrándonos en el daltonismo, aquellos casos con protanopía y deuteranopía, también se muestra qué es lo que se quiere hacer junto a una solución propuesta por el equipo. Luego explicando cuales son nuestros objetivos, las suposiciones y restricciones que son parte del proyecto.

Comentado [u1]: Digo que cuando digo, digo, digo diego
Realización realizando realizado realizar ..
Mala redacción

- **Propósito**:

El proyecto propone una forma de asistencia a personas con algún tipo de discapacidad visual, centrándonos principalmente en el daltonismo dicromático, especialmente la protanopía y deuteranopía. Esto se llevará a cabo utilizando la cámara del Smartphone con un software una aplicación de detección y reconocimiento transformación de colores en tiempo real, de tal manera que permita al incapacitado visual daltónico ver de manera natural.

Comentado [u2]: Mal redactado

- **Alcance**:

El software está dirigido principalmente a las personas que sufran algún tipo de daltonismo, dónde se tendrá que apuntar al objeto en cuestión para mostrar al usuario en tiempo real de acuerdo al tipo de patología que tenga. Se utilizará el lenguaje de programación Python, el cual será utilizado una vez que se realice el análisis y diseño del software.

- **Objetivo General**:

Desarrollar una aplicación que permita corregir la deficiencia de la visión cromática a una persona daltónica.

- **Objetivos específicos**:

  - Estudiar y analizar herramientas para reconocer el color en tiempo real

- Desarrollar un software que entregue un apoyo a las personas que sufren protanopía y deuteranopía.
- Probar una herramienta para reconocer el color en tiempo real y que asista a personas que sufren daltonismo.
- Entregar un producto final funcional y testeado.

- **Suposiciones y restricciones**:

  a) **Suposiciones**:

  - Las personas con daltonismo utilizan un Smartphone con capacidad de reconocimiento mediante sus cámaras.

  - Se da por hecho que las personas que utilicen la aplicación son aquellas que tengan algún tipo de daltonismo dicromático

  b) **Restricciones**:

  - El proyecto debe ser realizado en un plazo de un semestre académico.
  - El Smartphone debe apuntar directamente al objeto detectar color y corregirlo en tiempo real.
  - La aplicación funcionará en un Smartphone que tenga cámara.
  - La documentación tiene que estar de forma obligatoria en la plataforma de Redmine.
  - La aplicación debe funcionar para al menos dos tipos de daltonismo.

- **Entregables del proyecto**:

  1. Bitácoras semanales
  2. Informe de avance.
  3. Presentación de avance.
  4. Informe final.
  5. Presentación final.
  6. Wiki del proyecto.
  7. Manual de usuario.
  8. Producto final.

2. Organización del Proyecto

2.1 Personal y entidades

Jefe de proyecto, Diseñador, Programador, Documentador.

2.2 Roles y responsabilidades

Jefe de proyecto: Es la persona que coordina, organiza y representa al equipo de trabajo para qué todo se realice de forma eficiente. El responsable es: Gustavo Olivares

Diseñador: Personal encargado de diseñar diagramas que representen el procedimiento a seguir en el proyecto. Además, los que decidirán el diseño final de la interfaz de la aplicación. Los que tienen este rol son: Diego Berríos y Kevin Rodríguez.

Programador: Personal encargado de realizar la programación del software. Los que tienen este rol son: Diego Berríos y Kevin Rodríguez

Documentador: Se encargan de la redacción de los informes y cualquier tipo de documentación que tenga que realizarse como bitácoras, informes de avance, entre otros. El responsable es: Gustavo Olivares

Comentado [u3]: Secretario ejecutivo.. debe demostrar cual fue su rol en el desarrollo

2.3 Mecanismo de comunicaciones

Para la comunicación se utiliza un medio de comunicación llamado Discord el cual cada miembro del equipo utiliza frecuentemente.

Los informes, bitácoras y documentación en general están siendo subido a una carpeta compartida en drive para que cada miembro del equipo pueda verlo cuando sea necesario.

Comentado [u4]:

3. Planificación de los procesos de gestión

3.1 Planificación inicial del proyecto

- **Planificación de estimaciones:**

Tiempo estimado para el proyecto: 3 meses

Recurso	Valor	Cantidad
Notebooks	$800.000	3
Smartphone	$200.000	3
Visual Studio Code	Gratis	3
Microsoft Office	$40.000	3
Sueldo total de cada integrante	$600.000	3

Costo total del proyecto	$4.920.000

Comentado [u5]: 3x3x600= 5400 esto esta deficitario 5400 + 1040 = 6440.. no es 4920

- **Planificación de Recursos humanos:**

Diseñador:3
Programador: 3
Documentador: 3
Jefe de Proyecto: 1.

3.2 Lista de actividades

a) Actividades de trabajos

En la figura 1 se observa la Carta Gantt para mostrar la planificación del trabajo en grupo y el porcentaje de realización de cada actividad.

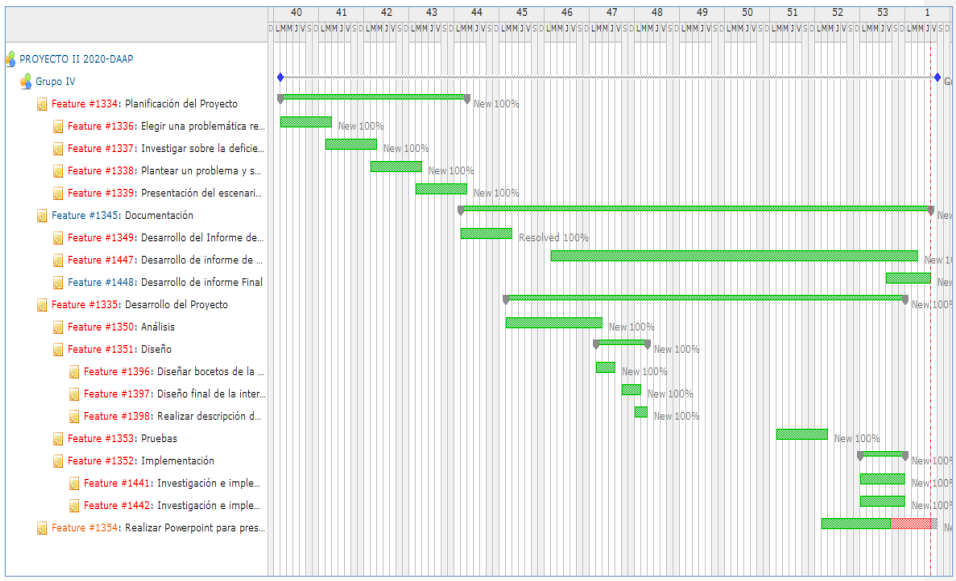

Figura 1 Carta Gantt para la planificación del Proyecto

Elegir una problemática relacionada a una deficiencia visual
Descripción: Se escoge un problema a resolver a través de reuniones en equipo.
Responsable: Gustavo Olivares
Producto: Problema en el cual se trabajará.

Investigar sobre la deficiencia visual escogida
Descripción: Se investiga sobre el problema escogido para poder plantear posibles soluciones en el futuro.
Responsable: Diego Berríos
Producto: Información sobre el daltonismo.

Plantear un problema y solución mediante un escenario experimental.
Descripción: Se prepara un escenario mostrando el problema a resolver junto con una posible solución a este.
Responsable: Gustavo Olivares
Producto: Un escenario experimental

Presentación del escenario experimental.
Descripción: El equipo presenta a la clase el escenario experimental que se realizó junto a una presentación en powerpoint mostrando el mismo junto a los objetivos del proyecto.
Responsable: Kevin Rodríguez
Producto: Presentación del escenario

Desarrollo del Informe de avance I
Descripción: Se desarrolla junto al equipo el primer informe de avance para la planificación del proyecto.
Responsable: Gustavo Olivares
Producto: Informe de avance I

Desarrollo del Informe de avance II
Descripción: Se desarrolla junto al equipo el segundo informe de avance para la planificación del proyecto.
Responsable: Gustavo Olivares
Producto: Informe de avance II

Desarrollo del Informe Final
Descripción: Se desarrolla junto al equipo el último informe de avance para la planificación del proyecto.
Responsable: Gustavo Olivares
Producto: Informe Final de proyecto

Diseñar bocetos de la interfaz
Descripción: Se desarrollan varios bocetos de cómo podría ser la interfaz de la aplicación.
Responsable: Diego Berríos
Producto: Bocetos de interfaz

Diseño final de la interfaz
Descripción: Se desarrollan diseños preliminares para ser utilizados en las primeras pruebas de la aplicación.
Responsable: Gustavo Olivares
Producto: Diseños preliminares de la interfaz.

Realizar descripción de la interfaz
Descripción: Con el diseño claro de la interfaz se realiza una descripción de esta de forma detallada de cada componente.
Responsable: Gustavo Olivares
Producto: Descripción de la interfaz.

Investigación e implementación del algoritmo para la corrección de colores
Descripción: Se investiga y desarrolla una aplicación de prueba utilizando un algoritmo de corrección de colores.
Responsable: Kevin Rodríguez
Producto: Programa prototipo para corrección de colores.

Investigación e implementación de la interfaz
Descripción: Se investiga y desarrolla una versión de la interfaz implementada en Python usando librerías Kivy.
Responsable: Gustavo Olivares
Producto: Interfaz implementada.

b) Asignación de tiempo

Comentado [u6]: Es parte de la carta gantt

Planificación del proyecto: 2-3 semanas.
Ejecución del proyecto: 6 a 7 semanas.
Cierre de proyecto: 1 semana.

3.3 Planificación de la gestión de riesgos

RIESGOS	PROBABILIDAD DE OCURRENCIA	NIVEL DE IMPACTO	ACCIÓN REMEDIAL
El cliente cambiará los requisitos.	80%	2	Presentar una planificación el cual el cliente acepte y no se hagan cambios críticos
Falta de formación en las herramientas.	60%	3	Investigar y experimentar con las herramientas que se van a utilizar.
La estimación del tamaño del proyecto es errónea.	80%	2	Realizar un análisis y diseño de una forma minuciosa para evitar un error en la estimación de gran margen.
Un compañero de equipo no esté disponible	20%	3	Tener planes en caso de ser necesario repartir el trabajo entre los integrantes restantes.
La tecnología disponible no cubre las necesidades del proyecto.	30%	1	Volver a replantear todo el proyecto primero analizando la tecnología existente para el problema que se quiere solucionar.

Niveles de impacto:

- 1: Catastrófico
- 2: Crítico
- 3: Marginal
- 4: Despreciable

4. Análisis de arquitectura

4.1 Especificación de requerimientos

Requerimiento Funcional	Descripción
1. La aplicación debe de utilizar la cámara del celular para captar imagen.	La aplicación debe utilizar la cámara del celular para captar principalmente semáforos y otros elementos que sea importante a diferenciar su color.
2. La aplicación debe poder comunicar el color al usuario.	La aplicación debe comunicar el color del objeto apuntado en cuestión ya sea en forma de texto o voz.
3. La aplicación debe poder utilizarse en Smartphone de bajas especificaciones.	La aplicación debe tener una opción que reduzca los requisitos de procesamiento para usuarios con Smartphone de bajas especificaciones.

Requerimiento no Funcional	Descripción
1. Límite de iluminación	La aplicación solo podrá funcionar si hay buena iluminación o puede no capturar imagen correctamente.

4.2 Descripción de la arquitectura y diseño de interfaz.

En la figura 2 se observa el diseño de la interfaz a la vez que tiene una explicación del uso de cada elemento de esta.

1. Primero se observa en la primera pantalla se encuentra las opciones que tendrá el usuario para escoger.
2. Luego al momento de seleccionar el tipo de daltonismo el sistema realiza internamente la corrección de colores adecuada a la selección.
3. La aplicación muestra la imagen a tiempo real de la cámara con su debida corrección de colores.

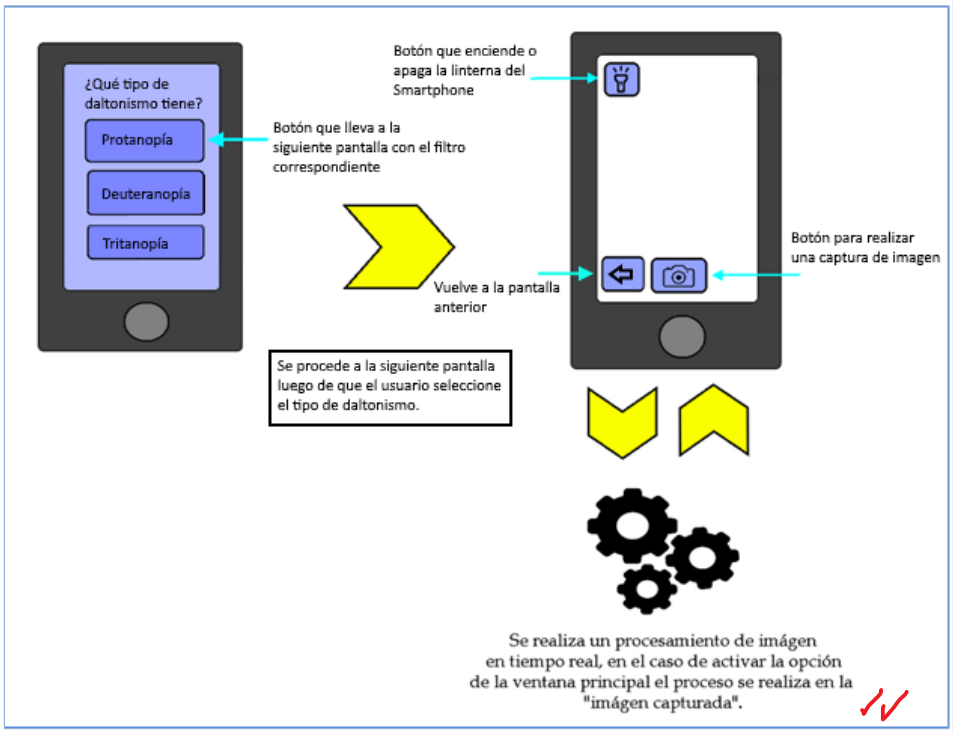

Figura 2 Diseño preliminar de la Interfaz de la Aplicación

4.3 Diagrama de caso de uso

En la figura 3 ~~observamos~~ se observaun diagrama de casos de uso que nos explica como es el proceso de la aplicación.
Primero el usuario inicia la aplicación, luego se incluye el siguiente caso de uso que viene de parte de la aplicación, donde solicita una opción a escoger al usuario. Luego incluye un caso de uso de capturar la imagen para después pasar al procesamiento de esta de acuerdo a la selección de la opción del caso de uso anterior “Solicita escoger opción”. Por último, el usuario recibe la imagen resultante en su pantalla con la corrección de colores correspondiente.

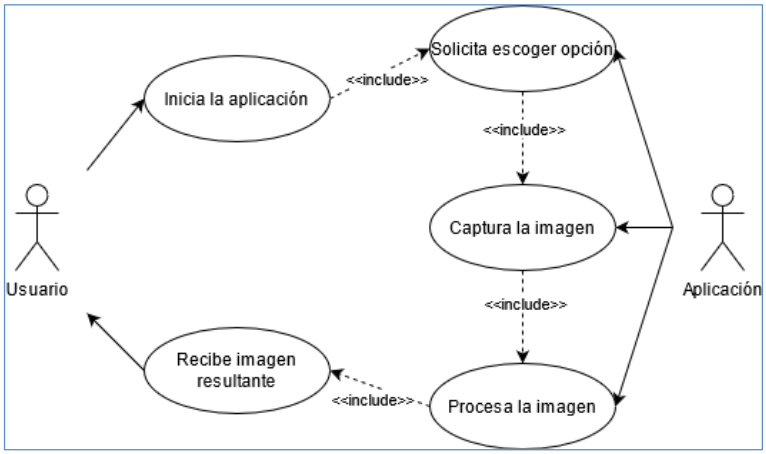

Figura 3 Diagrama de casos de uso

4.4 Descripción de casos de uso

Inicia la aplicación	
Fecha: 21/12/2020	
Descripción: Permite al usuario iniciar la aplicación móvil.	
Actores: Usuario	
Precondiciones: La aplicación debe tener acceso a ~~a~~ la cámara del celular.	
Flujo normal: Se elige una opción directamente para utilizar la aplicación	
Usuario: 1.- Inicia la aplicación. 3.- Selecciona la opción que le corresponde.	Sistema: 2.- Muestra opciones al usuario para elegir tipo de daltonismo. 4.- Incluye el caso de uso “Solicita escoger opción”.
Postcondiciones: La aplicación debe tener los filtros de imagen correspondientes a las opciones.	

Solicita escoger opción	
Fecha: 21/12/2020	
Descripción: Permite a la aplicación saber qué proceso debe realizar después.	
Actores: Aplicación	
Precondiciones: La aplicación debe tener así como a la cámara y las opciones de daltonismo correspondientes.	
Flujo normal: Muestra tipos de daltonismos disponibles	
Aplicación: 1.- Muestra opciones a escoger.	Sistema: 2.- Configura aplicación de acuerdo a la selección 3.- Incluye el caso de uso “Capturar la imagen”.
Postcondiciones: La aplicación obtendrá selección del usuario.	

Capturar Imagen
Fecha: 21/12/2020
Descripción: Permite obtener información de la cámara.
Actores: Aplicación

Precondiciones: Tener acceso a la cámara del Smartphone.	
Flujo normal: Mostrar imagen de la cámara	
Aplicación: 1.- Realiza un cambio de pantalla.	Sistema: 2.- Envía información de la cámara. 3.- Incluye caso de uso “Procesar Imagen”
Postcondiciones: Se tendrá la imagen para ser procesada.	
Procesar Imagen	
Fecha: 21/12/2020	
Descripción: Permite realizar el procesamiento de imagen.	
Actores: Aplicación	
Precondiciones: Tener información de la cámara.	
Flujo normal: Procesar imagen de acuerdo a la selección inicial.	
Aplicación: 3.- Recibe imagen procesada para mostrar.	Sistema: 1.- La imagen se procesa para el tipo de daltonismo seleccionado inicialmente. 2.- Envía la imagen procesada.

4.- Incluye caso de uso “Recibe imagen resultante”.	
Postcondiciones: Se tendrá imagen resultante en la aplicación	

Recibe imagen resultante	
Fecha: 23/12/2020	
Descripción: Permite confirmar que el usuario recibe la imagen resultante	
Actores: Usuario	
Precondiciones: La aplicación debe la imagen resultante.	
Flujo normal: El usuario puede ver la imagen resultante de acuerdo al daltonismo seleccionado.	
Usuario: 2.- Recibe la información de la imagen resultante.	Sistema: 1.- Muestra la imagen resultante en la pantalla de la cámara.
Postcondiciones: El usuario podrá ver la imagen con el procesamiento correspondiente al daltonismo seleccionado.	

4.5 Diagrama de secuencia

En la figura 4 observamos la secuencia en cómo se realiza la interacción del usuario con el sistema. Primero se observa que el usuario inicia la aplicación del sistema, luego el sistema muestra las opciones, el usuario escoge una opción, el sistema captura imagen, después la procesa y por ultimo entrega la imagen procesada al usuario.

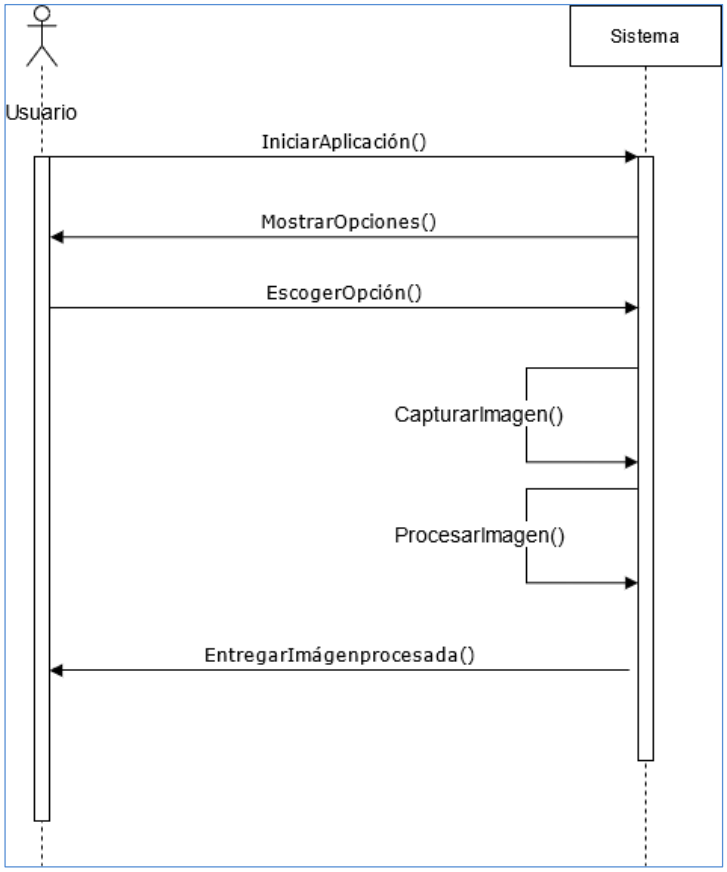

Figura 4 Diagrama que muestra la secuencia de funciones de la aplicación

4.6 Diagrama de clase

En el siguiente diagrama de la figura 5 se observa cómo funciona la aplicación con un diagrama clases. Se observa una clase Interfaz la cual almacena un string que es la opción escogida por el método escogerOpción(). Luego utiliza Cámara para la captura de imagen la cual se guarda como una matriz de colores en imagen_capturada con el método capturarImagen() esta imagen está representado como una clase Imagen que tiene una matriz llamada matriz_colores la cual después pasa a ser utilizada por la clase Conversión y esta utiliza los datos de la matriz de Imagen para convertirla usando el método convertirImagen() y se obtiene la matriz_resultante con la corrección de colores la cual es la que se muestra en la clase Interfaz.

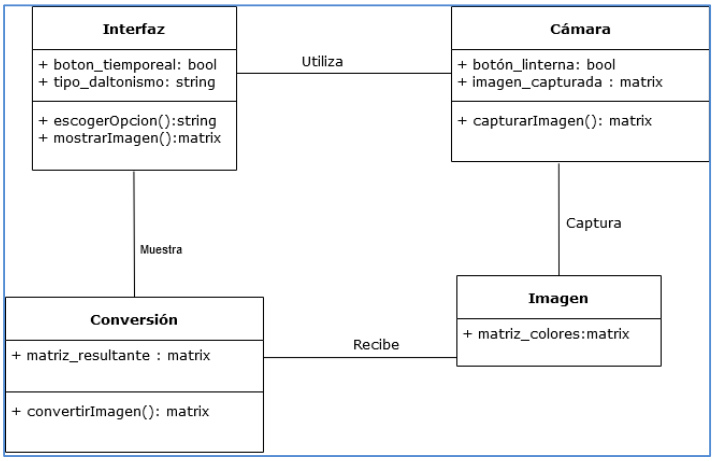

Figura 5 Diagrama de clases de la aplicación

5. Plan de integración

El proyecto consiste en la corrección de colores para daltónicos, para lograrlo se divide el trabajo en los puntos:

1. **Interfaz para que el usuario escoja su tipo de daltonismo.**

   Para esto se utiliza las librerías de interfaz llamada kivy.uix donde se utilizan los componentes Label, Button, BoxLayout. Estos utilizados de forma que se muestre una etiqueta señalando “que hacer” y, debajo de esta, tres botones cada uno con un tipo de daltonismo diferente, Protanopía, deuteranopía y tritanopía.

2. **Captura y corrección de color de la imagen.**

   Para la captura y corrección de imagen se utiliza la librería kivy.uix donde se utiliza principalmente el componente Camera, la librería numpy para controlar matrices de forma más compleja y kivy.graphics con su componente Texture. Utilizando estas librerías se realiza primero la captura de imagen con la cámara, luego utilizando la matriz de colores obtenida se hace el cálculo matemático utilizando el componente numpy, y luego se le aplica la corrección correspondiente con el componente Texture.

6. Pruebas de la aplicación

6.1 Problemas implementando cambio de pantalla:

Es una parte importante de la aplicación ya que de acuerdo a la forma en que se tiene organizada la interfaz se requiere que pueda hacerse la transición, el problema se presenta al tratar de probar la aplicación utilizando el archivo .kv para diseñar la interfaz, la solución principal y rápida que se realiza fue la de utilizar la libreria kivy.lang y su componente Builder para cargar este archivo como un string.

6.2 Problemas al sacar una foto con el filtro correspondiente:

En la aplicación es esencial que capture imagen en tiempo real y que la corrección se haga de igual manera, pero también que sea capaz de sacar una foto y se guarde en el dispositivo, lo cual generaba problemas al comienzo. Probando distintas maneras se logró que funcionara a través de un componente cámara y un componente de la interfaz con una id.

7. Modelo de implementación

7.1 Algoritmo que realiza la corrección de colores.

Como se muestra a continuación en la Figura 6 el algoritmo utilizado para realizar la corrección de colores utiliza una matriz para tener primero una simulación de cómo ve un daltónico en cada caso. Luego se realiza la corrección de color correspondiente utilizando otra matriz y finalmente realizando el cálculo final entre ambas.

```python
def simulate(rgb, color_deficit="d"):
    # matrices para transformar el espacio de colores
    cb_matrices = {
        "d": np.array([[1, 0, 0], [1.10104433, 0, -0.00901975], [0, 0, 1]], dtype=np.float16),
        "p": np.array([[0, 0.90822864, 0.008192], [0, 1, 0], [0, 0, 1]], dtype=np.float16),
        "t": np.array([[1, 0, 0], [0, 1, 0], [-0.15773032, 1.19465634, 0]], dtype=np.float16),
    }
    # matriz para transformar rgb a lms (simular lo que capta la retina)
    rgb2lms = np.array([[0.3904725, 0.54990437, 0.00890159],
                        [0.07092586, 0.96310739, 0.00135809],
                        [0.02314268, 0.12801221, 0.93605194]], dtype=np.float16)
    # inversa precomputada
    lms2rgb = np.array([[2.85831110e+00, -1.62870796e+00, -2.48186967e-02],
                        [-2.10434776e-01, 1.15841493e+00, 3.20463334e-04],
                        [-4.18895045e-02, -1.18154333e-01, 1.06888657e+00]], dtype=np.float16)
    # pasamos del espacio RGB al LMS
    lms = transform_colorspace(rgb, rgb2lms)
    # Calcular la imagen como la ve un daltonico
    sim_lms = transform_colorspace(lms, cb_matrices[color_deficit])
    # Volvemos a RGB
    sim_rgb = transform_colorspace(sim_lms, lms2rgb)
    return sim_rgb
```

Figura 6 Algoritmo de corrección de colores

7.2 Implementación final de la interfaz y Pruebas del funcionamiento del algoritmo de corrección de colores.

- **Interfaz Principal.**

En la Figura 7 podemos observar la interfaz finalmente implementada en Python utilizando las librerías de kivy.uix.

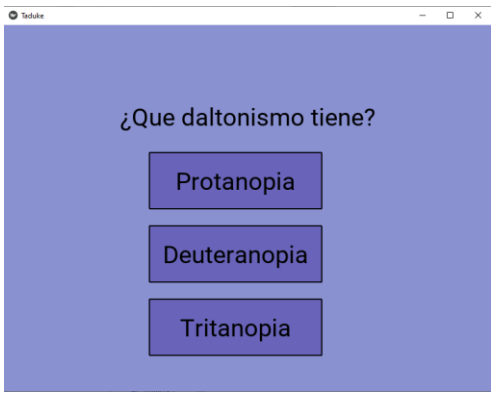

Figura 7 Interfaz implementada

- **Protanopía.**

En la Figura 8 ~~podemos~~ se puede observar una prueba de la corrección de colores para el tipo de daltonismo dicromático Protanopía.

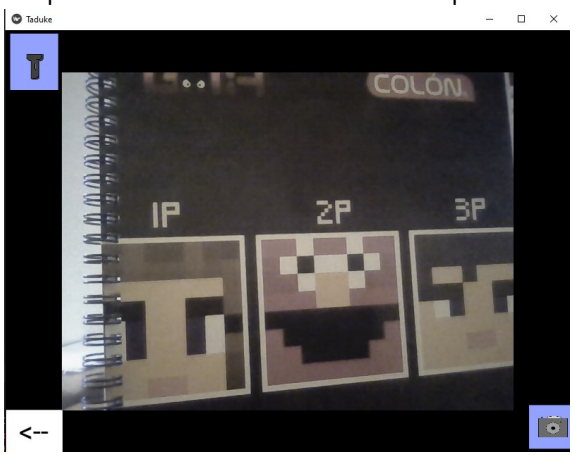

Figura 8 Prueba opción Protanopía

- **Deuteranopía.**

En la Figura 9 podemos observar una prueba de la corrección de colores para el tipo de daltonismo dicromático Deuteranopía.

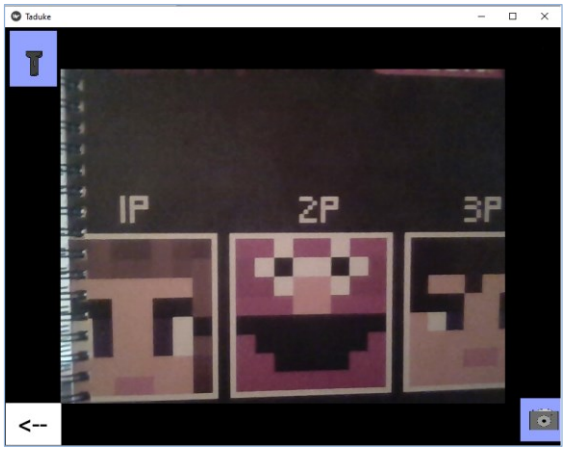

Figura 9 Prueba opción Deuteranopía

- **Tritanopía.**

En la Figura 10 podemos observar una prueba de la corrección de colores para el tipo de daltonismo dicromático Tritanopía.

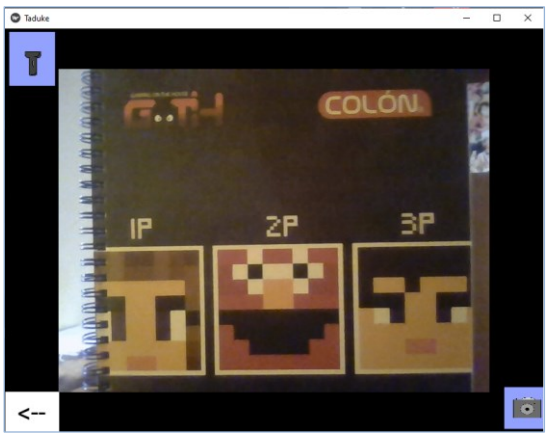

Figura 10 Prueba opción Tritanopía

8. Conclusiones

En el proyecto se realiza un desarrollo de una aplicación centrada en asistir a personas con deficiencia cognitiva visual, a diferencia de cursos anteriores esta vez nos centramos en un sector en específico y un tipo de tecnología llamada “Open Computer Vision” (OpenCV) para el procesamiento de imágenes, para ello se utiliza Visual Studio Code, lenguaje de programación Python y librerías Kivy principalmente, se logra crear una aplicación funcional donde ~~se crea~~ se produce una imagen con su debida corrección de colores ~~a~~ en tiempo real, utilizando cálculos matemáticos ~~con matrices~~matriciales, para obtener una matriz resultanmte ~~de~~ con colores ~~corregida~~ corregidos para asistir a los tres tipos de daltonismo dicromático, Protanopía (carencia de sensibilidad al color rojo), Deuteranopía (carencia de sensibilidad al color verde) y Tritanopía (carencia de sensibilidad al color azul). Por último, también la aplicación contiene una función de sacar una foto con la corrección de colores correspondiente al usuario.

9. Referencias

Básica:

- S Poret, R D Dony y S Gregori, School of Engineering, University of Guelph, Guelph, ON Canada, “Image Processing for Colour Blindness Correction” obtenido de https://ieeexplore.ieee.org/document/5444442
- JINJIANG LI, XIAOMEI FENG y HUI FAN, School of Computer Science and Technology, Shandong Technology and Business University, Yantai 264005, China, “Saliency Consistency-Based Image Re-Colorization for Color Blindness” obtenido de https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9090216
- Documentación de kivy obtenido de https://kivy.org/doc/stable/

Complementaria:

- Apuntes del professor.