Universidad Rey Juan Carlos

# DESARROLLO DE UN VECTORSCOPIO PARA DISPOSITIVOS MÓVILES ANDROID

**Grado en Ingeniería de Sistemas Audiovisuales y Multimedia**

**Autor: Elena María del Río Galera**

**Tutor: David Gualda Gómez**

Escuela de Ingeniería de Fuenlabrada

# Índice de contenidos


1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

# Índice de contenidos


1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

Universidad Rey Juan Carlos

# 1. Introducción

Creación de una aplicación *Android* que emula un vectorscopio.

Punto de partida → *Script* en *Matlab.*

- Motivación: Aprender a programar en *Android*.

**¿Por qué un vectorscopio?**

1. Proyecto innovador en *Android.*
2. Conocer cómo funciona.
3. Conocer más el mundo audiovisual.

# Índice de contenidos

1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

Universidad Rey Juan Carlos

# 2. Objetivos y Fases de desarrollo

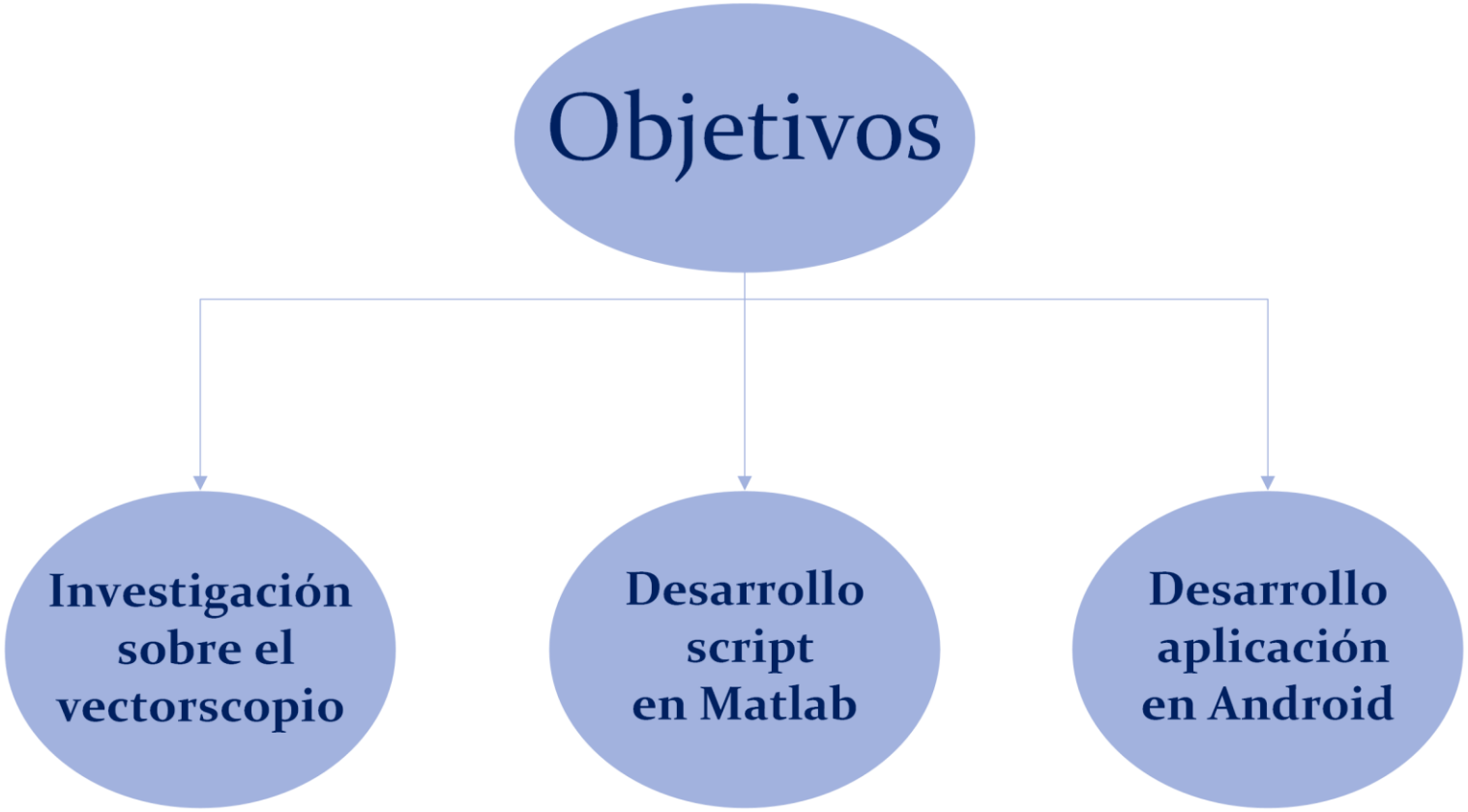



**Diagrama de Objetivos.**

Universidad Rey Juan Carlos

# 2. Objetivos y Fases de desarrollo

1. Fase de investigación.
2. Fase de desarrollo:
   - Desarrollo en *Matlab*.
   - Desarrollo en *Android.*
3. Fase de evaluación de resultados.
4. Fase de documentación.

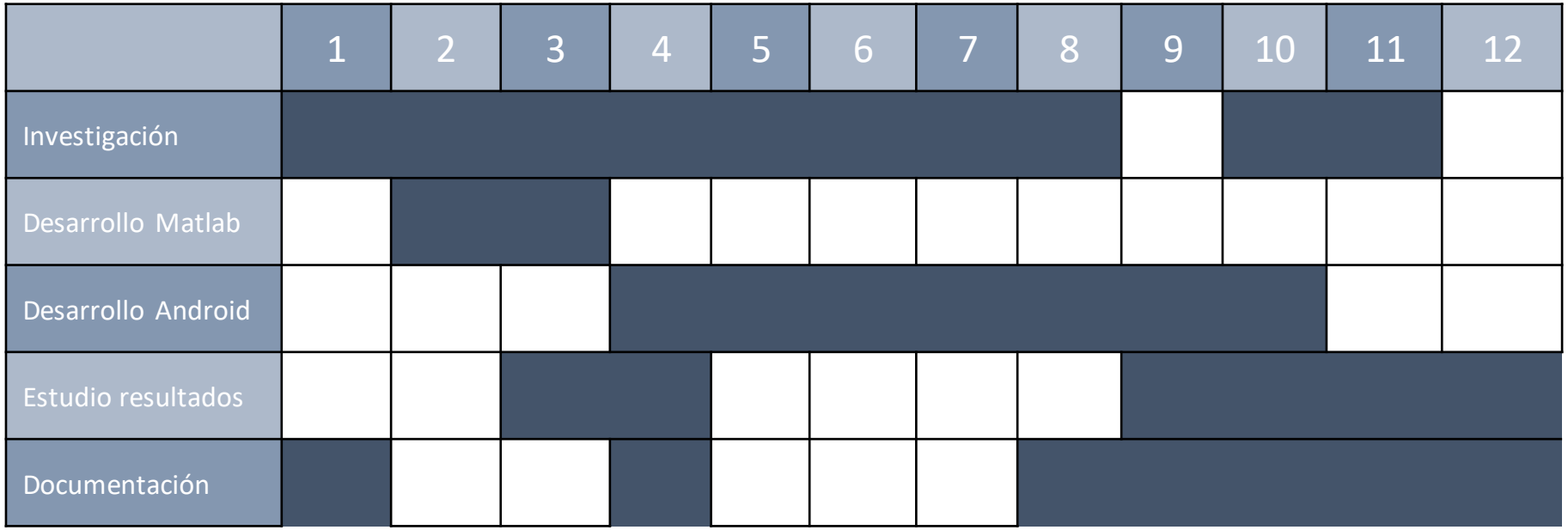



**Diagrama de Gantt relativo al proyecto.**

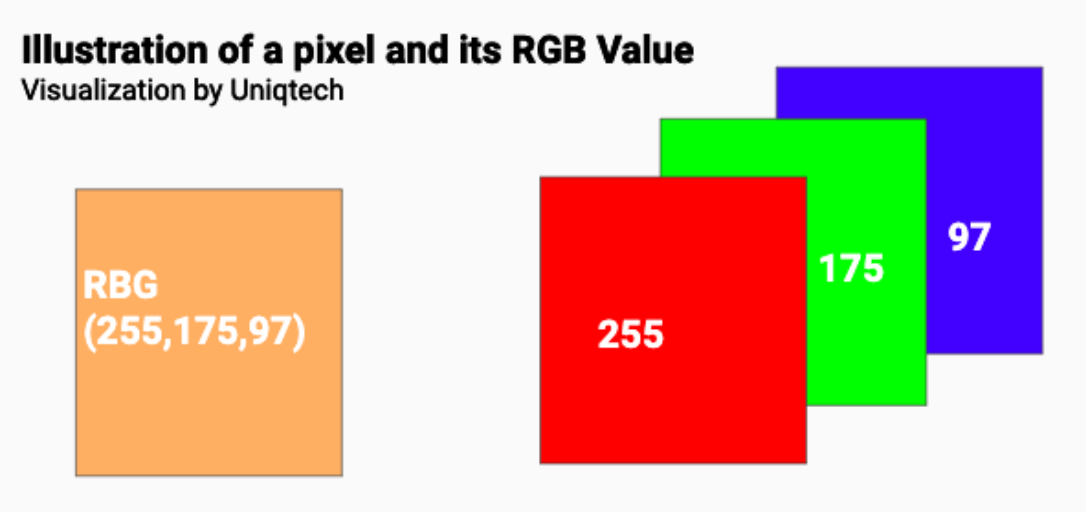
# Índice de contenidos


1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

# 3. Definiciones y Conceptos

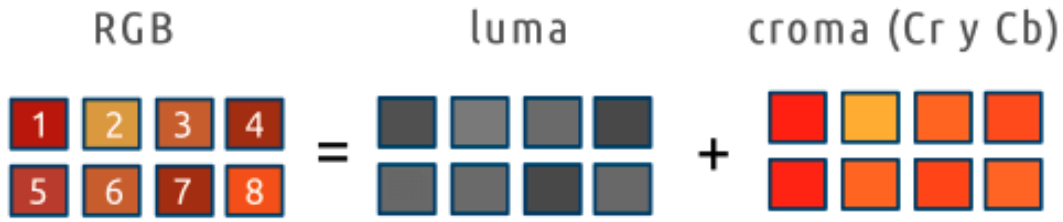
**Píxeles**

- Elementos pequeños que componen la imagen.
- División en subpíxeles RGB (255,255,255 → Blanco).




**Distribución de los valores RGB de un píxel [2].**

- Cada píxel tiene información de luma y croma(color).

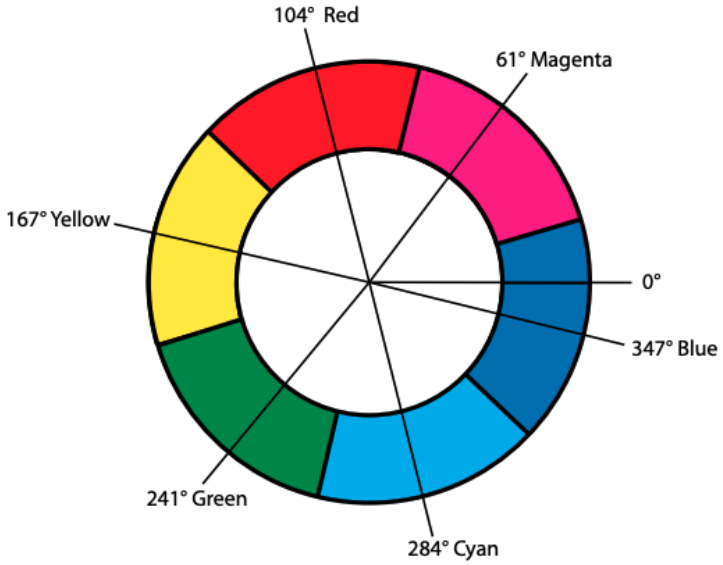



**Luma y croma en píxeles [3].**

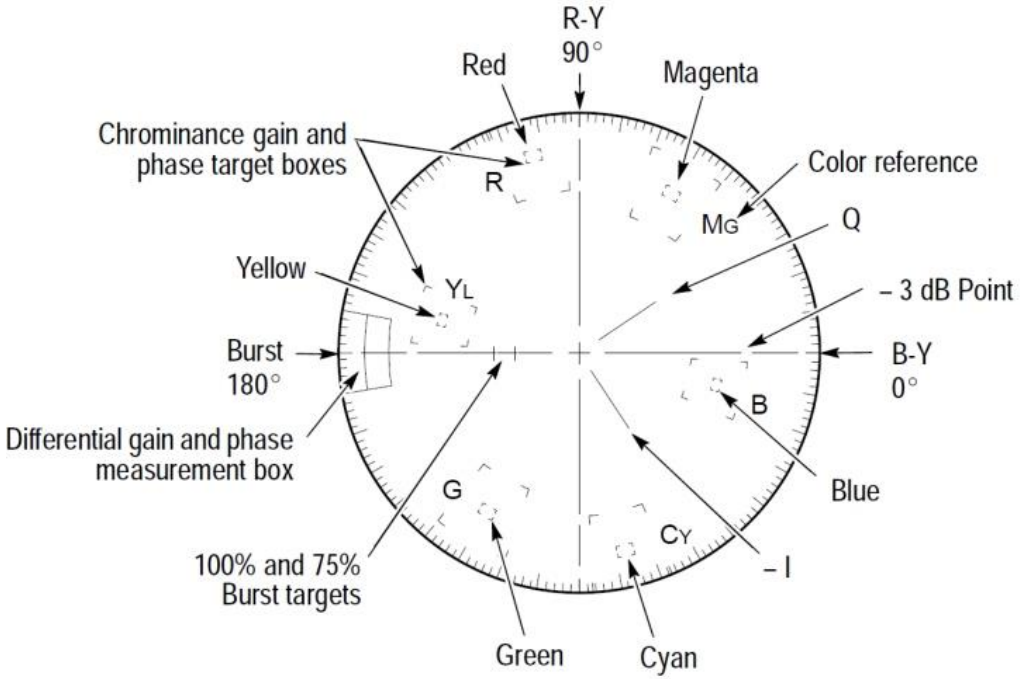
# 3. Definiciones y Conceptos

## Vectorscopio

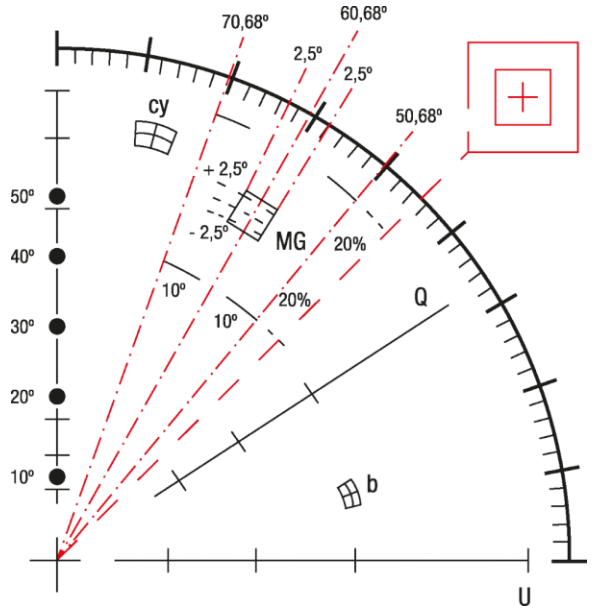
- Osciloscopio, especializado en la representación de la parte de crominancia de la señal de vídeo.




**Rueda de colores en la que se basa el vectorscopio [4].**




**Explicación del Vectorscopio NTSC [5].**

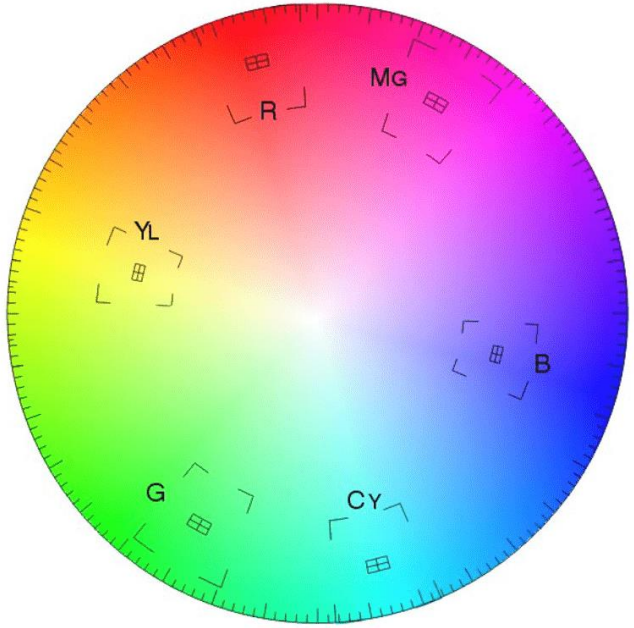



**Detalle de un cuarto del vectorscopio [6].**

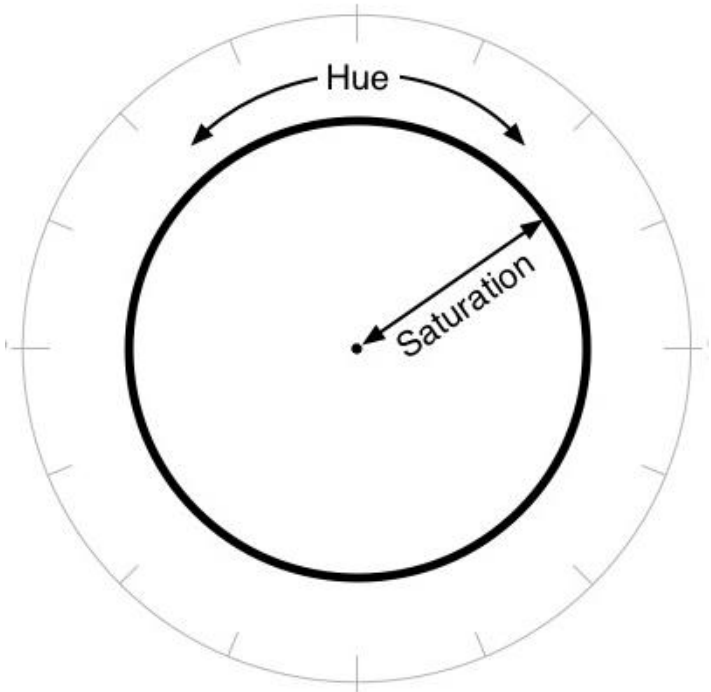
# 3. Definiciones y Conceptos

**Interpretación del vectorscopio**

- Cada punto en el vectorscopio representa los tonos puros.
- El gris neutro se sitúa en el centro.
- El ángulo alrededor del centro representa el tono del color.
- La distancia que separa un punto del centro representa la saturación.




**Vectorscopio integrado con la rueda de colores [7].**

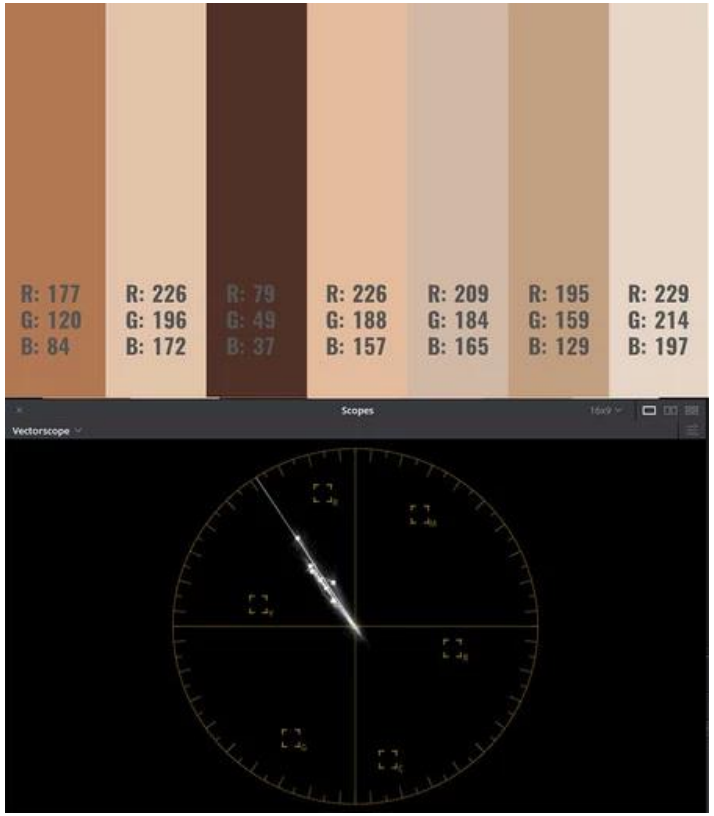



**El tono y la saturación en el vectorscopio [8].**

# 3. Definiciones y Conceptos

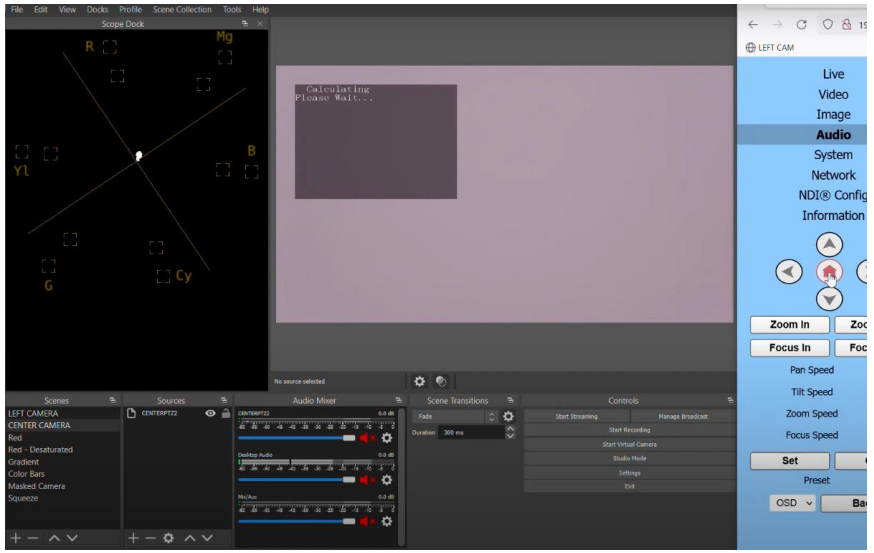
## Usos del vectorscopio

### Corrección de tonos de piel.



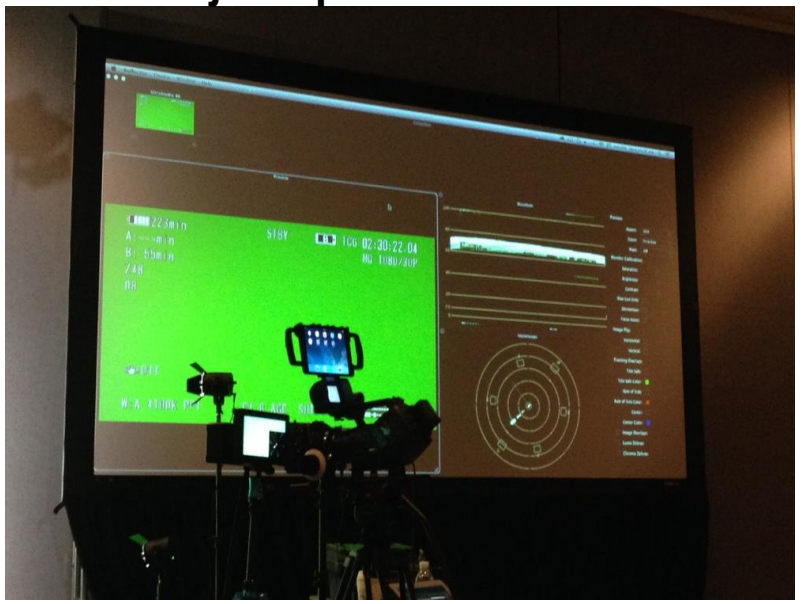

Representación de barras de tonos de piel en *Davinci Resolve* [11].

### Balance de blancos.




Balance de blancos de cámara fotográfica [12].

### Ajuste pantalla verde.



Uso del vectorscopio para pantalla verde en estudio [13] .

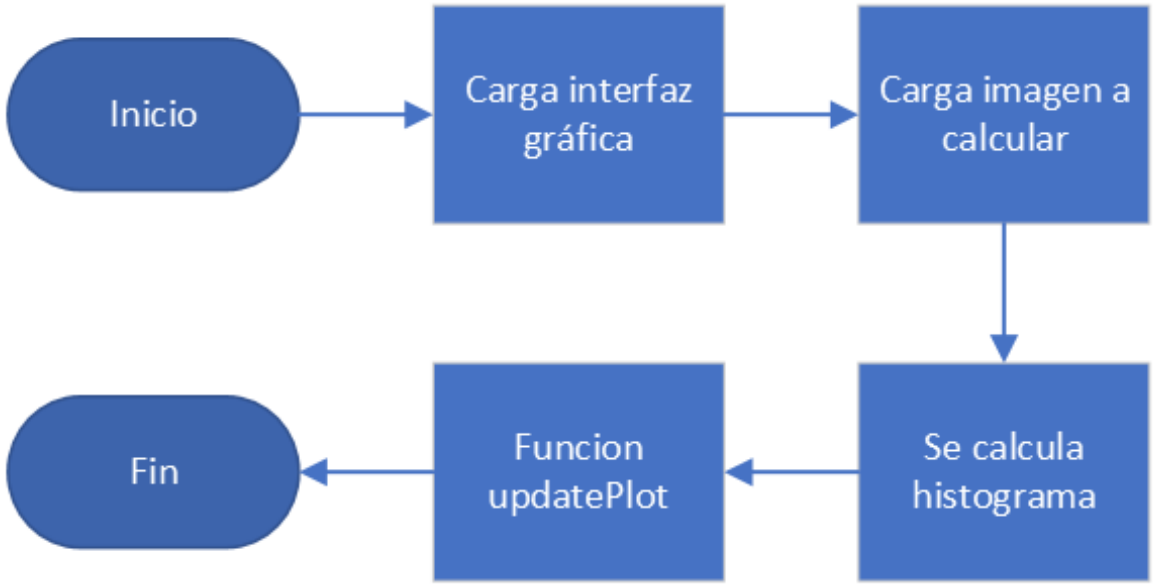
# Índice de contenidos

1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

Universidad Rey Juan Carlos

# 4. Desarrollo del trabajo realizado

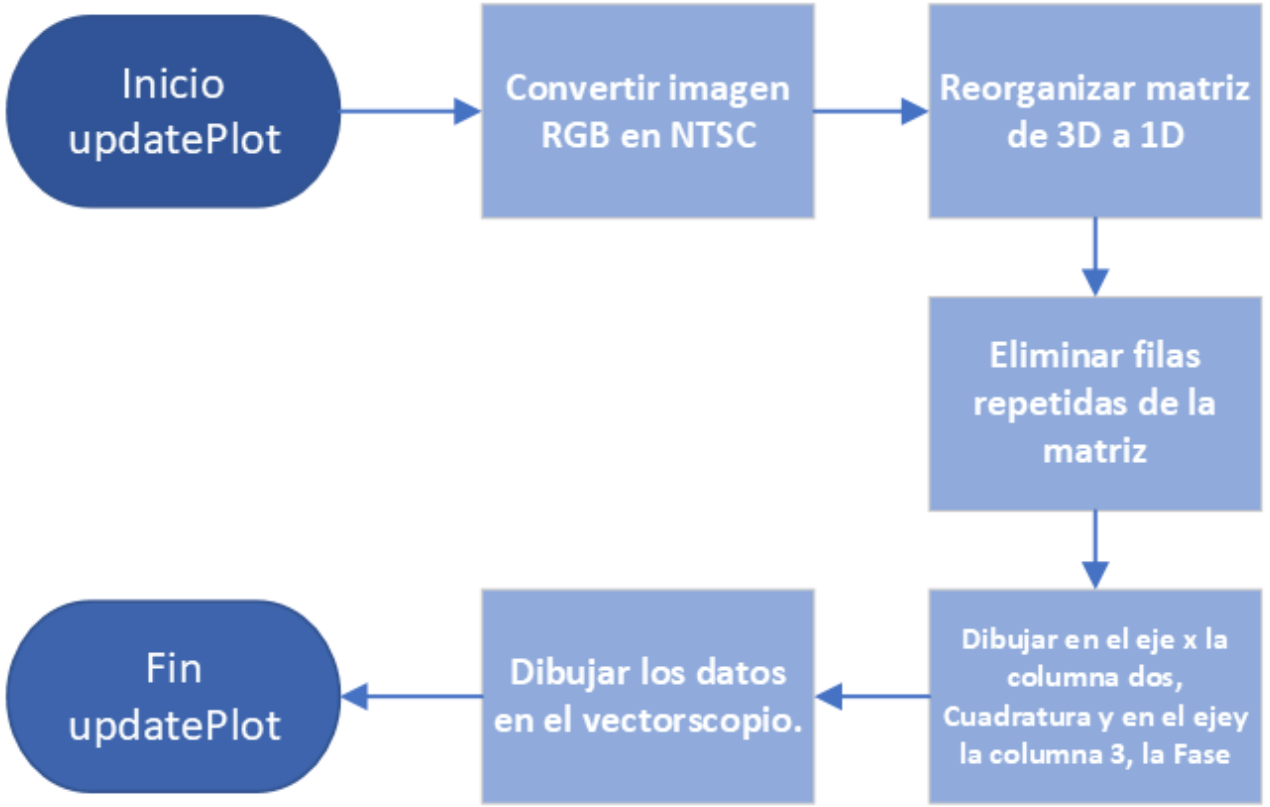
## i. Implementación en Matlab




**Flujograma programa en *Matlab* a alto nivel.**

# 4. Desarrollo del trabajo realizado

## i. Implementación en Matlab




1. *rgb2ntsc*
2. *permute (mxnxd= nxmxd)*

*3D to 2D*

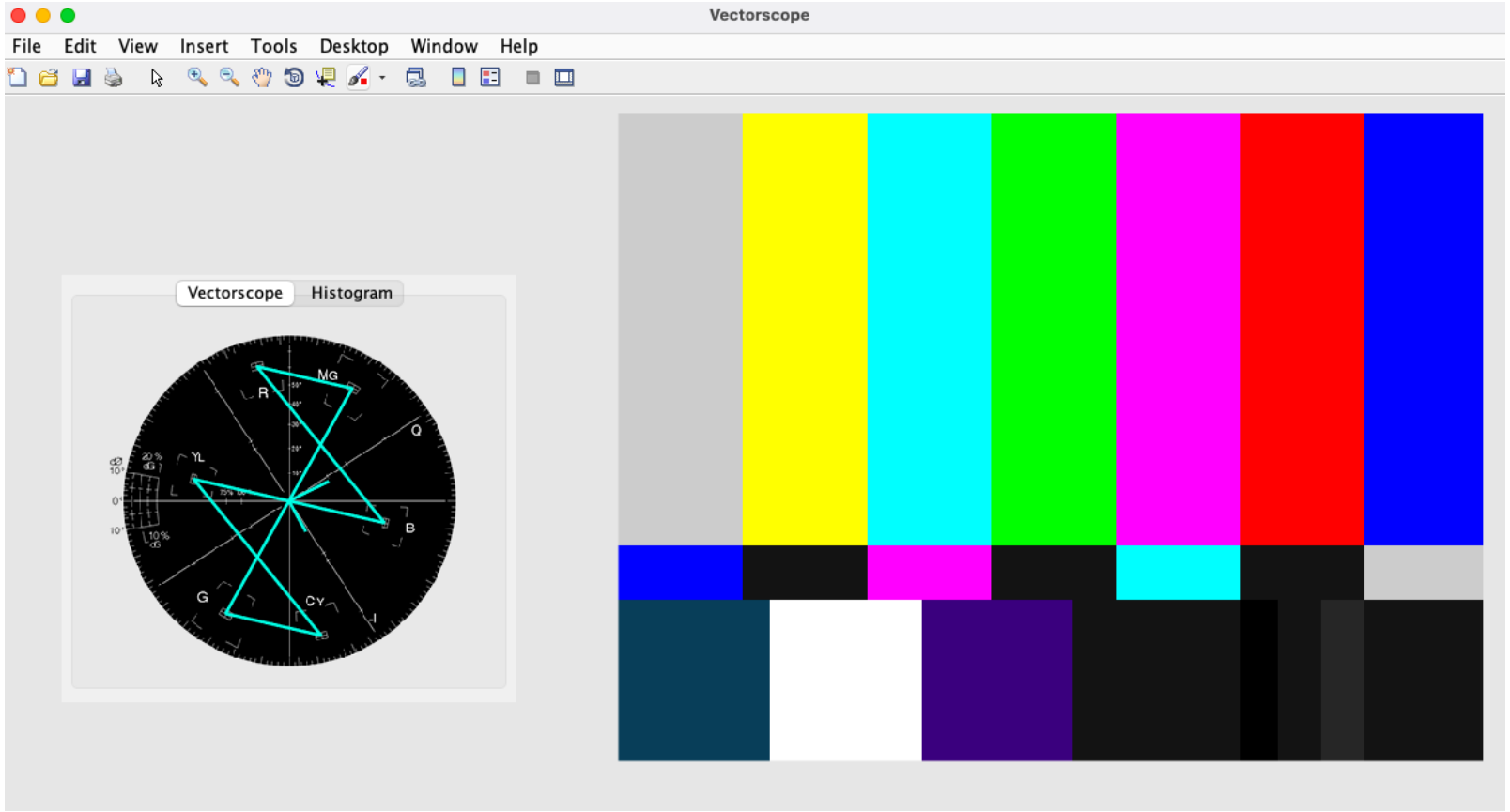
1. *map = reshape (3D matrix (mxn, d))*
2. *points = 2D matrix (mxn, d) zeros*
3. *points = unique(map)*
4. *Eje x = 2ª fila (cuadratura)*
5. *Eje y = 3ª fila (fase)*

**Flujograma función *updatePlot().***

Universidad Rey Juan Carlos

# 4. Desarrollo del trabajo realizado

## i. Implementación en Matlab



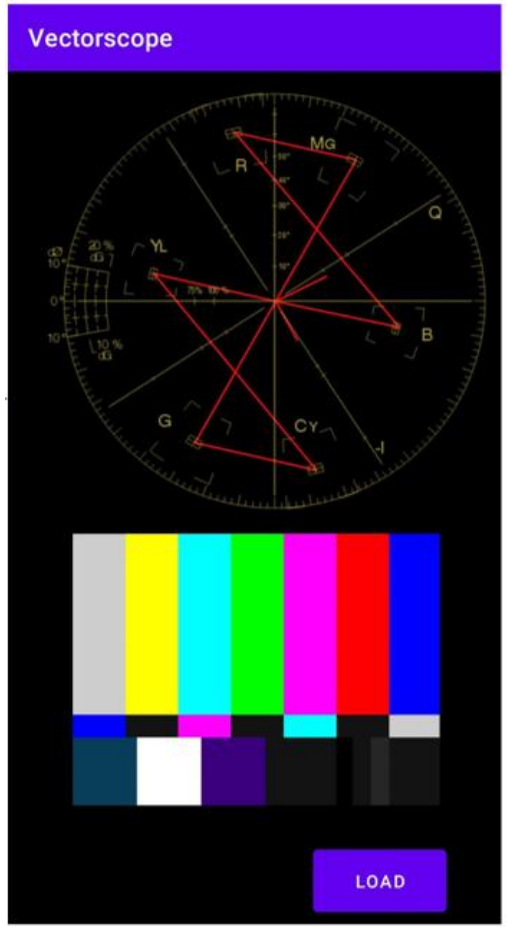

**Vectorscopio en *Matlab* con imagen de barras SMPTE.**

# Índice de contenidos

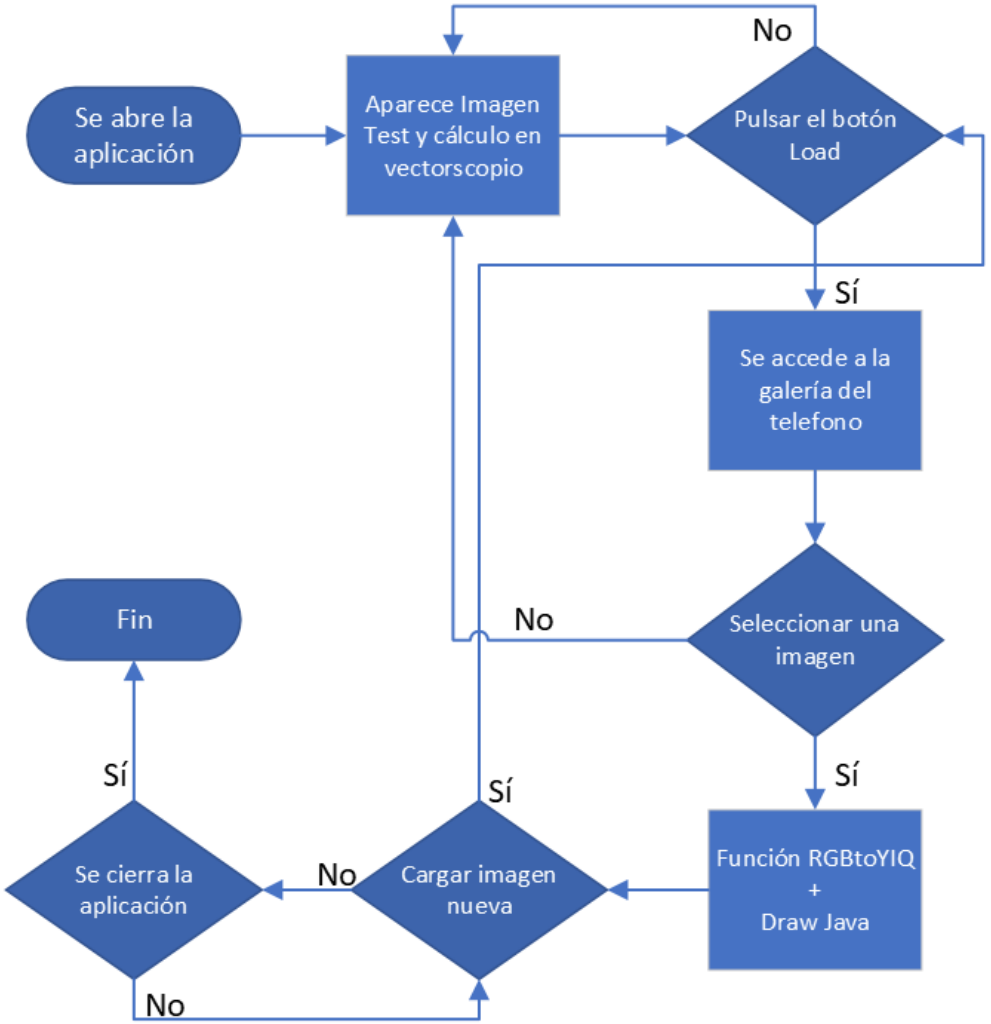

1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android




**Pantalla inicial aplicación *My Vectorscope*.**

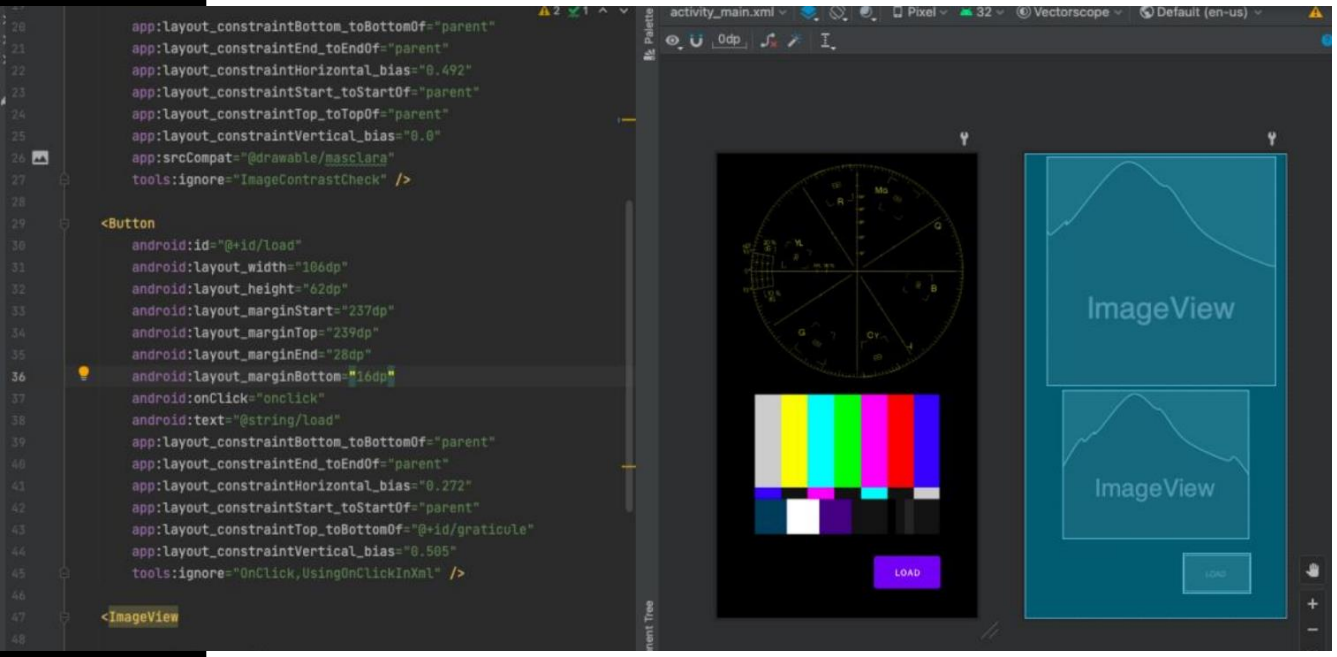



**Flujograma programa en *Android*.**

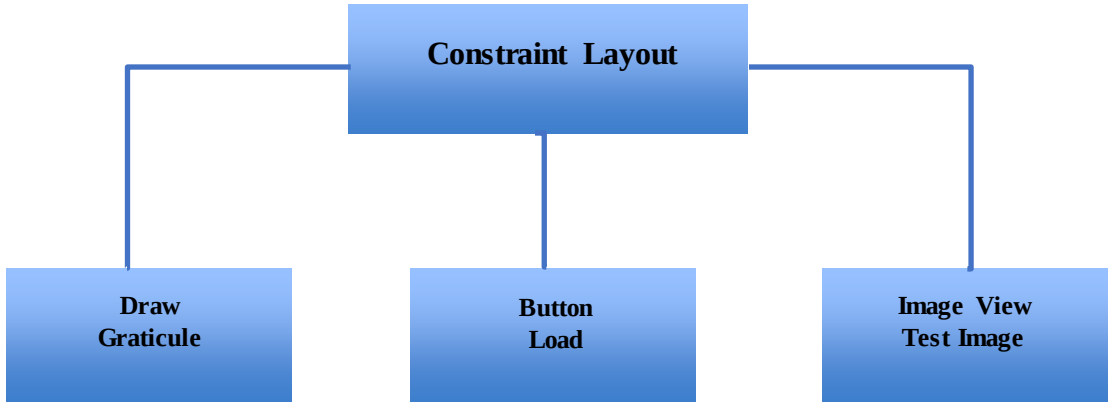
# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android

### *layout.xml*




*layout.xml en Android Studio*.




Equema de la estructura *layout.xml*.

# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android

***AndroidManifest.xml***

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.my_vectorscope">
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.Vectorscope">
        <activity
            android:name="com.example.my_vectorscope.MainActivity"
            android:screenOrientation="portrait"
            android:exported="true"
            tools:ignore="LockedOrientationActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

***AndroidManifest.xml*.**

# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android

### *MainActivity.java*

- Archivo principal
- Inicialización de variables globales (outHeight, outWidth…)

**public void onCreate**

Lógica de arranque básica de la aplicación que debe ocurrir una sola vez en toda la vida de la actividad.

Se recuperan los elementos necesarios del XML

*graticule = findViewById (R.id. graticule);*

**public void onClick**

Se ejecuta cada vez que se pulsa el botón Load.

Se asocia con el button view en *layout.xml.*

*Android: onClick=""onclick"".*


**Ventana al pulsar *Load.***

# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android

**public void onActivityResult**

Evalúa si se ha seleccionado una imagen.

Redimensiona.

Se crea el bitmap relativo a la imagen y se coloca en el image view para que el usuario pueda verla.

Se llama a RGBtoYIQ().

**public RGBtoYIQ**

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

**Cálculo para pasar de RGB a YIQ [18].**

```
//Matriz directamente permutada
float[][][] matrizYIQ = new float[outWidth][outHeight][3];
for (int x = 0; x < matrizYIQ.length; x++) { // matriz.length = 180, columnas
    for (int y = 0; y < matrizYIQ[x].length; y++) { //matrix.length[x] = 135,filas
        for (int z = 0; z < matrizYIQ[x][y].length; z++) { //matriz.length[x][y] = 3 dimensiones

            int pixel = imagenFinal.getPixel(x, y);
            r = (pixel >> 16) & 0xff;
            g = (pixel >> 8) & 0xff;
            b = pixel & 0xff;
            //Para pasar de uint 8 a double hay qye dividir entre 255
            Y = (float) (Math.round(((0.299 * r + 0.587 * g + 0.114 * b) / 255.0) * 1000.0) / 10000.0);
            (matrizYIQ[x][y][0]) = Y;

            //Log.i("!!!!!!", "Y" + Y);
            I = (float) (Math.round(((r * 0.595879 + g * -0.274133 + b * -0.321746) / 255.0) * 10000.0) / 10000.0);
            (matrizYIQ[x][y][1]) = I;
            Q = (float) (Math.round(((r * 0.211205 + g * -0.523083 + b * 0.311878) / 255.0) * 10000.0) / 10000.0);
            (matrizYIQ[x][y][2]) = Q;
        }
    }
}
```

**Bucle para pasar de RGB a YIQ.**

Universidad Rey Juan Carlos

# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android

**public RGBtoYIQ**

Convertir matriz 3D EN 2D *matriz2D [x ∗ y][z]*

Recorrer matriz *3D*, y almacenar valores de Y en columna 0, valores de I en columna 1 y valores de Q en columna 2.

```
150         //Inicializamos la u, nueva fila de la nueva matriz en cero
151         int u = 0;
152         // Recorremos las dimensiones
153         for (int k = 0; k < dimensiones; k++) {
154             for (int j = 0; j < outHeight; j++) {
155                 for (int i = 0; i < outWidth; i++) {
156                     if (k == 0) {
157                         //Rellenar primera columna con La información relativa a Luminancia
158                         matriz2D[u][0] = matrizYIQ[i][j][k];
159                     } else if (k == 1) {
160                         //Rellenar segunda columna con La información relativa a in-phase
161                         matriz2D[u][1] = matrizYIQ[i][j][k];
162                     } else {
163                         //Rellenar tercera columna con La información relativa a quadratura
164                         matriz2D[u][2] = matrizYIQ[i][j][k];
165                     }
166                     //Una vez relleno dato, saltamos a la siguiente fila
167                     u++;
168                 }
169             }
170             // Para la siguiente dimensión, empezamos en la fila 0, reiniciamos la u
171             u = 0;
172         }
```

**Bucle para crear matriz 2D.**

Universidad Rey Juan Carlos

# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android

**public RGBtoYIQ**

Separar valores de cuadratura y fase:

- Recorrer matriz 2D y almacenar en dos listas independientes.
- Convertir Array List en Array.

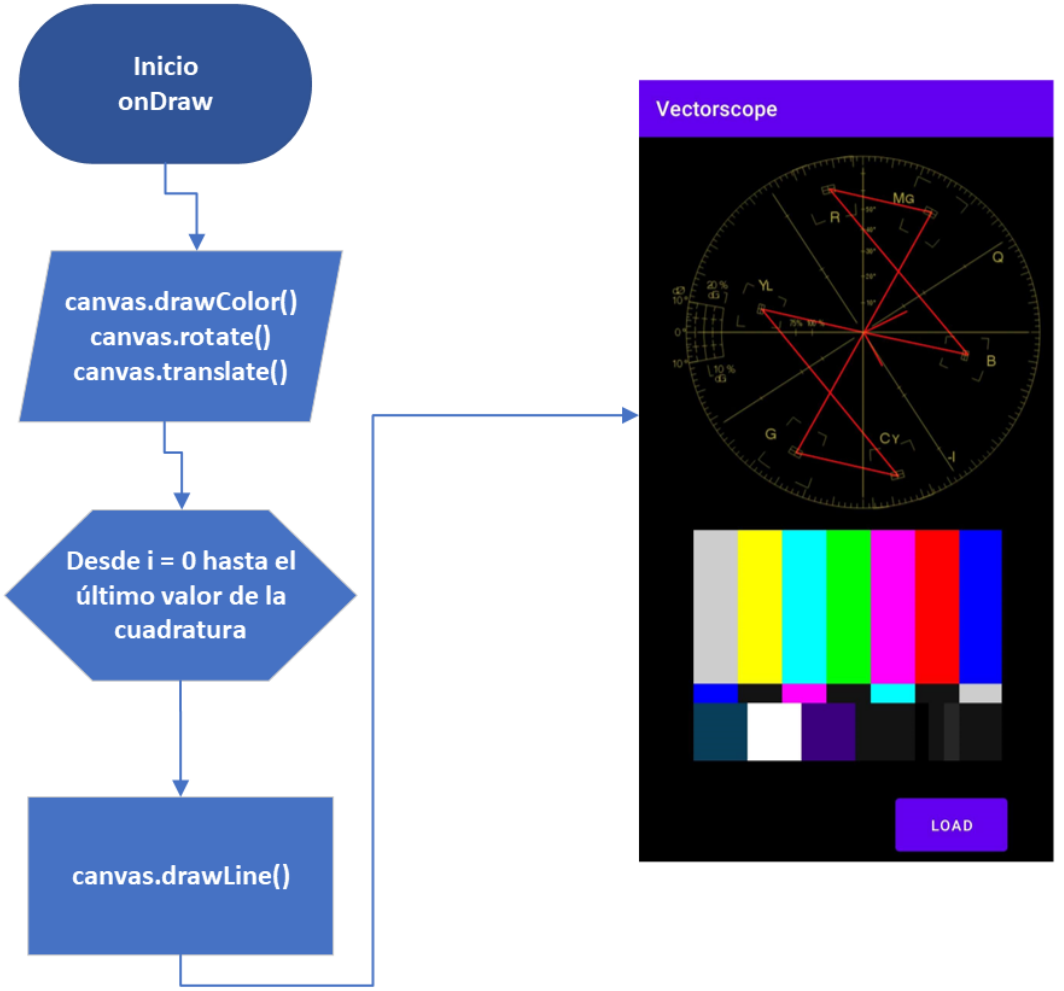
```
167         // Lista ordenada con Cuadratura y fase
168         for (int i = 0; i < matriz2D.length; i++) {
169             for (int j = 0; j < matriz2D[0].length; j++) {
170                 if (j == 1) {
171                     Cuadratura.add(i, (matriz2D[i][1]));
172                 } else if (j == 2) {
173                     Fase.add(i, matriz2D[i][2]);
174                 }
175             }
176         }
177
178         cuadratura_final = new float[Cuadratura.size()];
179         for (int i = 0; i < Cuadratura.size(); i++) {
180             cuadratura_final[i] = Cuadratura.get(i);
181
182         }
183         fase_final = new float[Fase.size()];
184         for (int i = 0; i < Fase.size(); i++) {
185             fase_final[i] = Fase.get(i);
186         }
```

# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android

### *Draw.java*

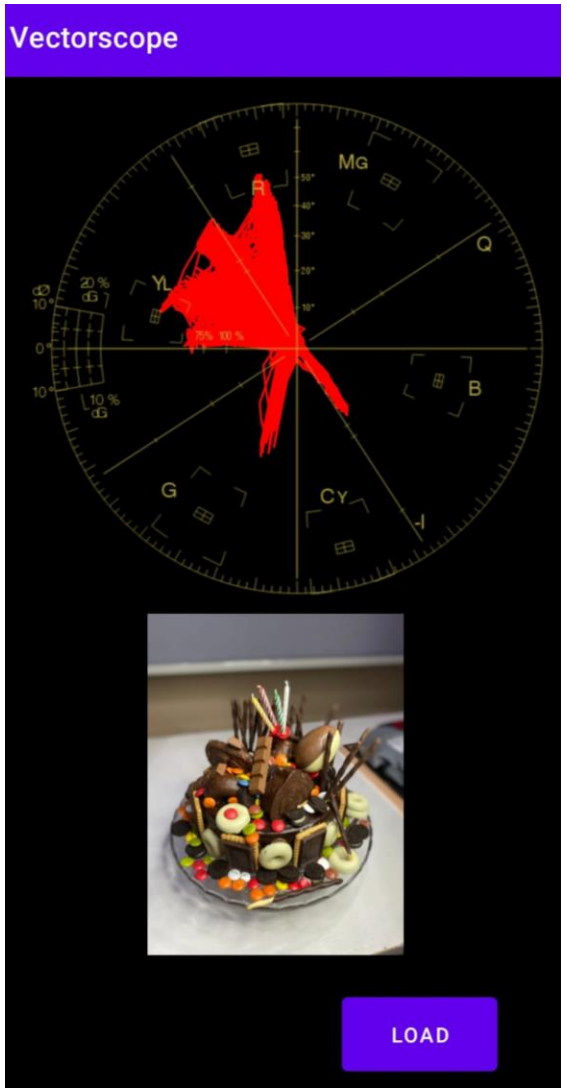
- Extiende directamente del *imageview* correspondiente a la gráfica del vectorscopio.
- ***setupDrawing*,** se indican los parámetros con los que se va a pintar.(grosor, color…)
- *onDraw*, define que el *canvas* sea transparente. Define la rotación y translación
- Se recorre desde i = 0 hasta valor máximo
    - Para cada valor se dibuja línea desde valor de cuadratura al de fase.



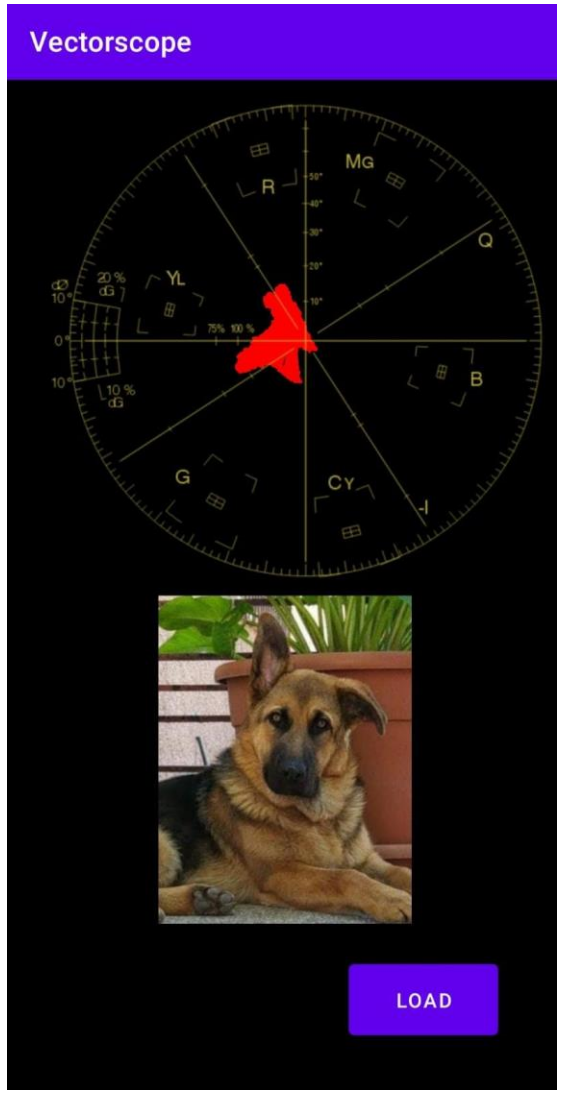

**Flujograma de *Draw.java*.**

# 4. Desarrollo del trabajo realizado

## ii. Implementación en Android








**Ejemplos de visualización de imágenes en la aplicación My Vectorscope.**

# Índice de contenidos

1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

# 5. Resultados

## Imagen de Barras

- Cada color en el centro de su cuadrado, tonos puros.
- Grises, Blancos y Negros, en el centro.
- Líneas I y Q, vídeo analógico.

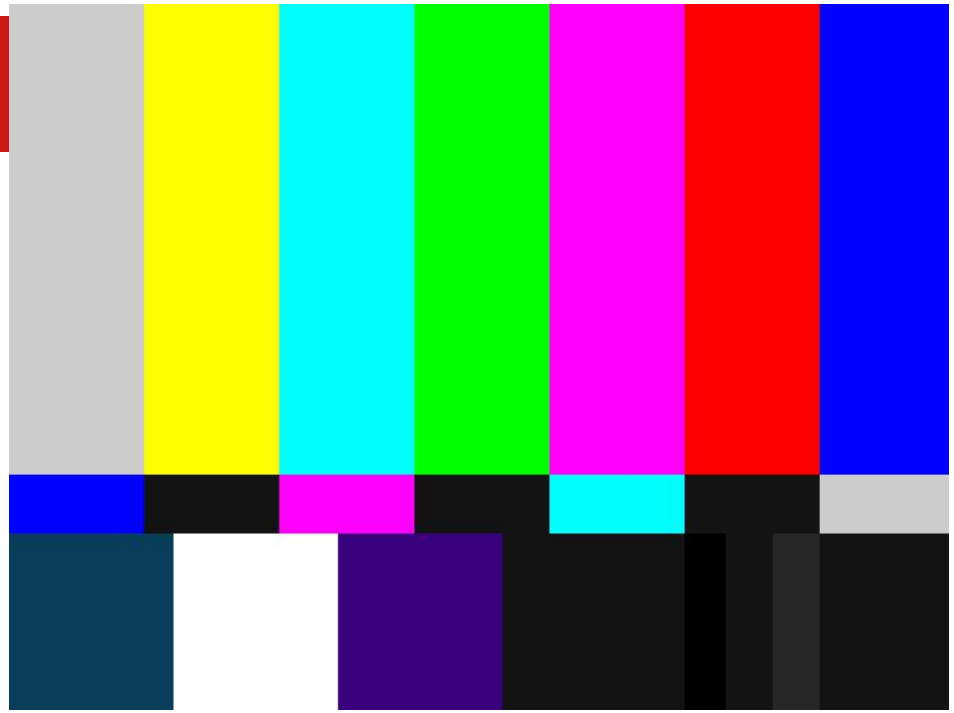
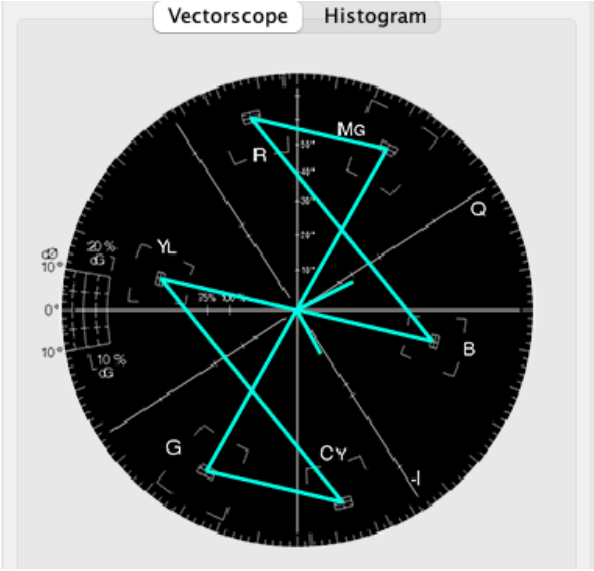
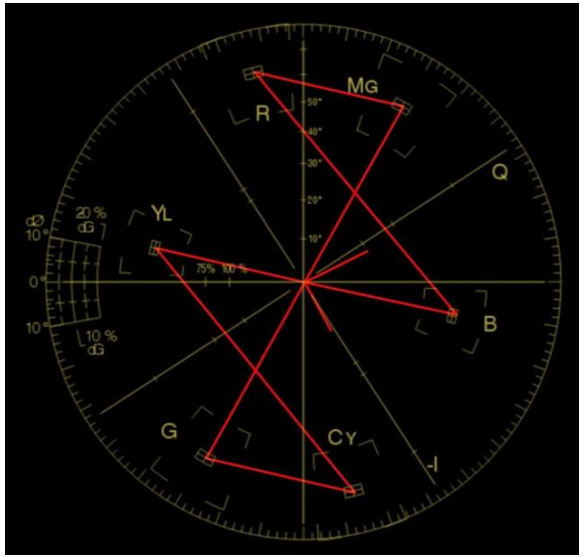


**Imagen de Barras [23].**




**Matlab**




**My Vectorscope**

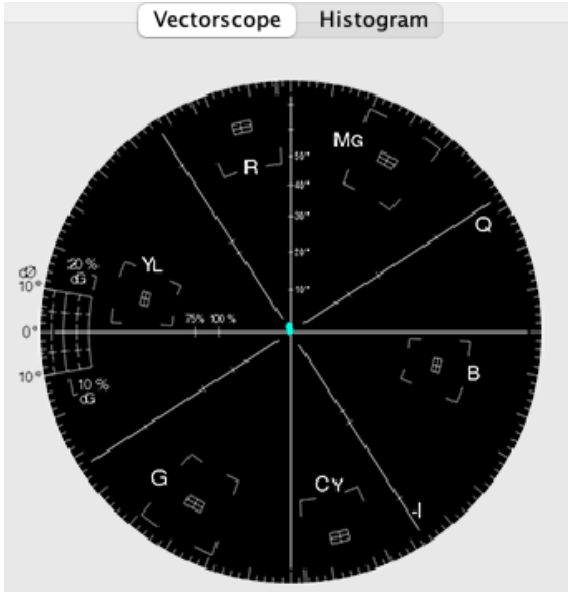

**After Effects**

# 5. Resultados

## Imagen escala de grises

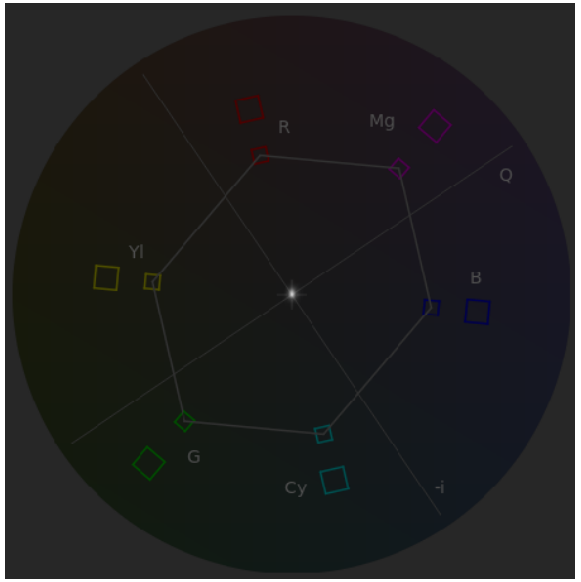
- Gris, blanco y negro no tienen información de color.
- Se distingue un punto en el centro de la rueda del vectorscopio.

**Imagen en escala de grises [24].**




**Matlab**


**My Vectorscope**




**After Effects**

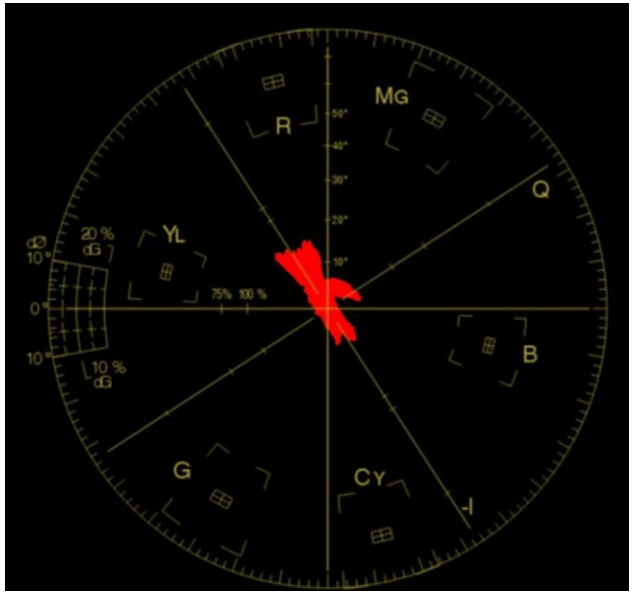
# 5. Resultados

## Imagen de persona

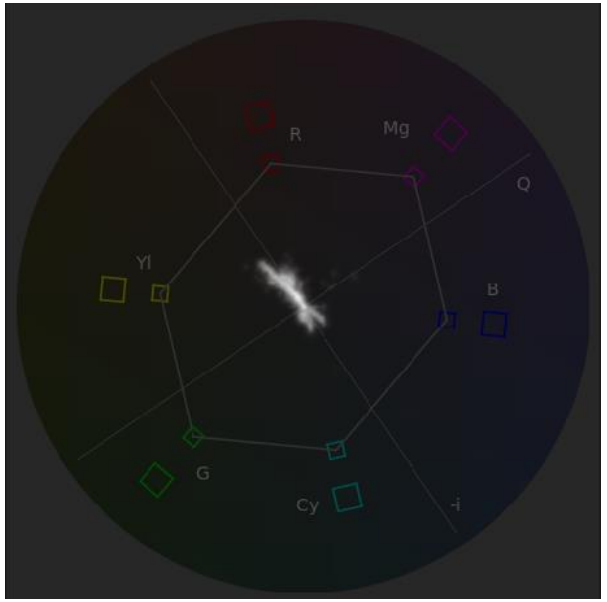
- Línea de tono de piel entre Yl y R.
- Distribución entre Cian y Azul, tonos del jersey.
- Entre el magenta y el azul, tonos marrones (barba, pelo, fondo...).

Imagen de hombre con barba [25].


Matlab




My Vectorscope




After Effects

# Índice de contenidos

1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

# 6. Costes

**Costes materiales:**

- Ordenador → 800 euros.
- *Smartphone Android → 200 euros.*
- Licencia Matlab →800 euros.

**Tiempo estimado:** 360 horas.

**Costes humanos:**

- Ingeniero con conocimientos de programación en Matlab y Java → 30 euros/hora (360 x 30 = 10.800 euros).

| Tipo de Coste | Coste total [€] |
|---|---|
| ***Mano de obra*** | 10800 € |
| ***Material*** | 1000 € |
| ***Licencias*** | 800 € |
| TOTAL | 12600 € |

**Tabla resumen de Costes.**

# Índice de contenidos

1. Introducción
2. Objetivos y Fases de desarrollo
3. Definiciones y Conceptos
4. Desarrollo del trabajo
    i. Matlab
    ii. Android
5. Resultados
6. Costes
7. Conclusiones

# 7. Conclusiones

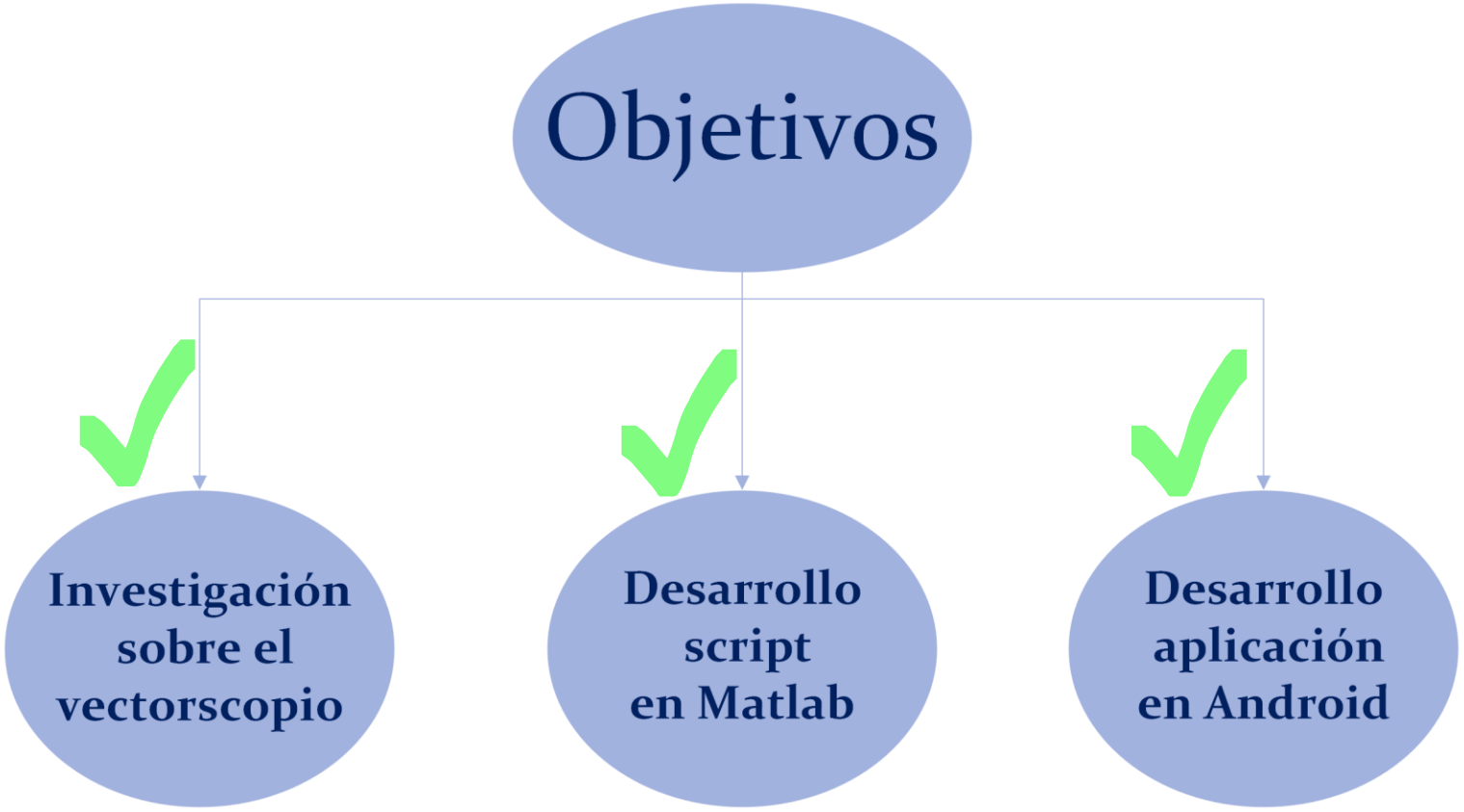



**Diagrama de Objetivos.**

# 7. Conclusiones → Competencias empleadas

## Competencias empleadas

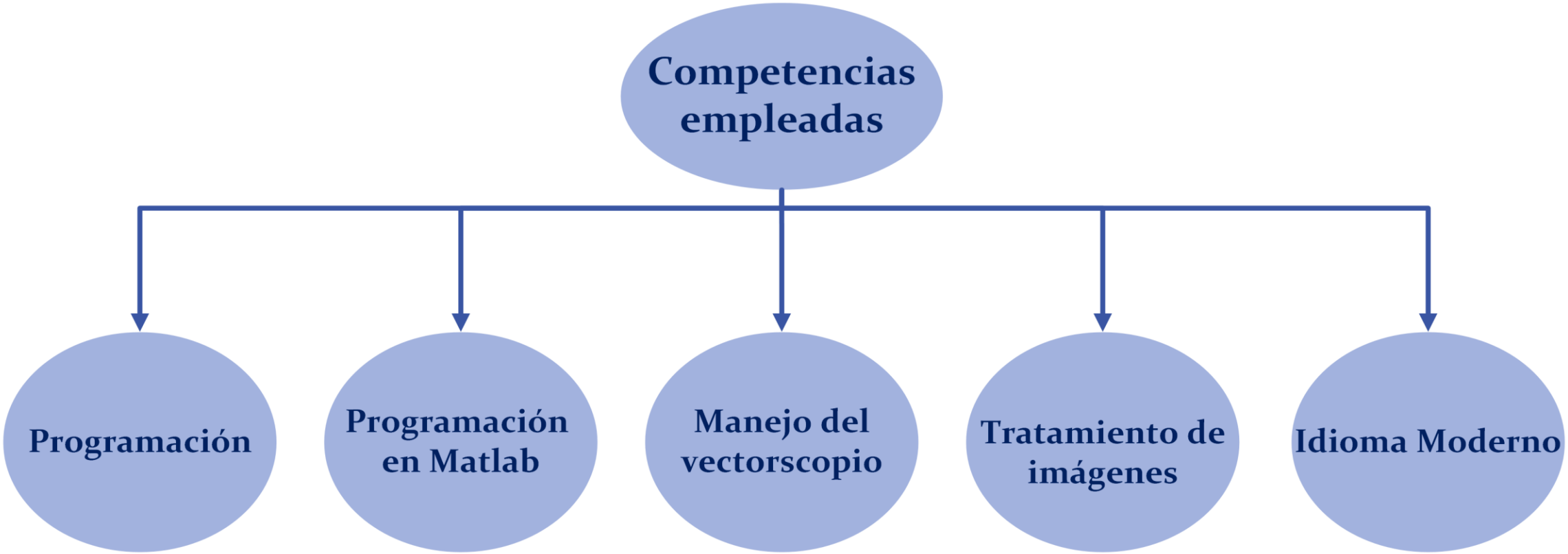



**Diagrama de competencias empleadas.**

# 7. Conclusiones → Competencias adquiridas

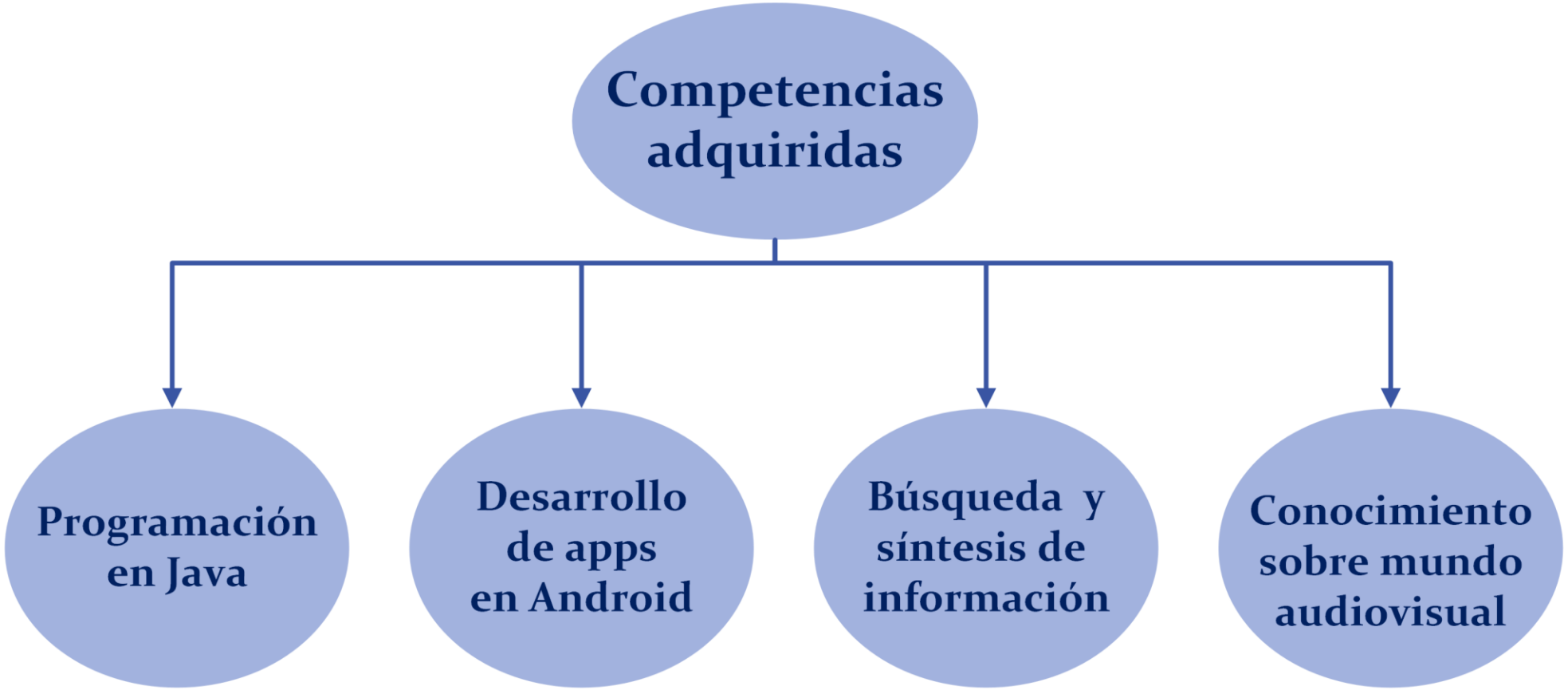



**Diagrama de competencias adquiridas.**

Universidad Rey Juan Carlos

# 7. Conclusiones → Trabajos Futuros

- Mejorar la adaptación de la aplicación en diferentes tamaños de pantalla.
- Poder estudiar vídeos frame a frame.
- Añadir como fuente de entrada de datos la posibilidad de tomar imágenes/vídeos en tiempo real.
- Añadir pantallas extra con otros elementos de cálculo de imagen, como monitor de onda, histograma, separación de canales etc.
- Integrar la aplicación en una aplicación de edición de vídeos.

# Bibliografía


[1] Pixeles, «Ionos Website», 2023 [En línea]. Available: https://www.ionos.es/digitalguide/paginas-web/diseno-web/que-es-un-pixel/

[2] Uniqtech. (2019). Distribución RGB en un píxel. [Figura]. Recuperado de:

https://miro.medium.com/v2/resize:fit:640/format:webp/1*KzSu3ytpLj8Dl0v2qVC1zw.png

[3] Fernández, F. (2023). Separación de valores de luma y Croma de un píxel. [Figura]. Recuperado de:

https://quecamararareflex.com/wpcontent/uploads/2016/11/submuestreo_croma_chroma_subsampling.png

[4] Mavis. (2018). Rueda cromática. [Figura]. Recuperado de: https://support.shootmavis.com/hc/en-us/articles/212451065-Vectorscope-

[5] Borys Golik. (2010). «Software Interface for Video Image Quality Analysis» Recuperado de:

https://www.imageengineering.de/content/library/diploma_thesis/borys_golik_software_interface.pdf

[6] Detalle del vectorscopio https://tilanotv.es/wp-content/uploads/2021/02/Figura-12.16- 983x1024.png

[7] Tetronik. (2013). The graticule of a vectorscope superimposed with a color wheel.[Figura].

Recuperado de: https://download.tek.com/document/25W_29166_0_Interactive.pdf

[8] Mavis. (2017). Drawing Colour. [Figura]. Recuperado de: https://support.shootmavis.com/hc/en-us/article_attachments/206962009/vectorscope.jpg

[9] Steve Hullfish y Jamie Fowler, « Color Correction for video», 2008, Focal Press

[10] Steve Hullfish, «The Art and Technique of Digital Color Correction», 2008, Focal Press [21] Fabara, S. (2018).

# Bibliografía


[11] Carta de colores de tonos de piel en Davinci Resolve. [Figura]. Recuperado de: https://i.blogs.es/be182d/pieles002/1366_2000.webp

[12] Balance de blancos cámara fotográfica. [Figura]. Recuperado de: https://coloristfactory.com/2022/09/16/professionals-secret-guide-to-fixing-white-balance-in-under-a-minute-filmlook/

[13] Pantalla verde y vectorscopio. [Figura]. Recuperado de: https://images.squarespace-cdn.com/content/v1/5046b167e4b0b2bcc3a91ee3/1425615428505-VGIFU21PWXW615XC36DD/image-asset.jpeg?format=2500w

[14] MathWorks, «MATLAB Website», 2022. [En línea]. Available: https://es.mathworks.com/help/matlab/.

[15] Rafael C. Gonzalez, Richard E. Woods y Steven L. Eddins, «Digital Image Processing Using MATLAB», 2003, Pearson Prentice Hall.

[16] Jerome DiMarzio, «Beginning Android Programming with Android Studio», 2016, Wrox

[17] Organización de XML en Android, «Academia Android Website», 2022. [En Línea]. Available: https://academiaandroid.com/tratamiento-de-xml-en-android-introduccion/

[18] Imgur. (2022). Cálculo RGB to YIQ. [Figura]. Recuperado de: https://i.stack.imgur.com/R22jh.png

[19] Jef Friesen y Peter Späth, «Learn Java for Android development», 2020, Apress.

[20] Programming with Android Java «Developer Android Website», 2023. [En Línea]. Available: https://developer.android.com/docs

[21] Iván Guerrero Vaquerizo, «Sistemas de producción audiovisual», 2017, Paraninfo

# Bibliografía

[22] iStock. (2022). Imagen de fresas. [Figura]. Recuperado de: https://www.finedininglovers.com/es/sites/g/files/xknfdk1706/files/styles/article_1200_800_fall back/public/2022-04/fresas%C2%A9iStock.jpg?itok=iBcd_HLd

[23] Pixabay. (2020). Carta de barras de color SMPTE. [Figura]. Recuperado de: https://cdn.pixabay.com/photo/2020/11/30/18/14/smpte-color-bars-5791787_1280.png

[24] Hudson, M. (2022). Imagen de paisaje en escala de grises. [Figura]. Recuperado de: https://michael-hudson.com/wp-content/uploads/2022/08/landscape-gaaf71654b_640.jpg

[25] Aprendum. (2020). Imagen de hombre con barba. [Figura]. Recuperado de: https://d2qc4bb64nav1a.cloudfront.net/cdn/13/images/curso-online-de-como-leer-el-rostro-enlas-personas:-los-microgestos_l_primaria_1_1561117635.jpg

[26] Jorge Carrasco González, «Cine y televisión digital. Manual técnico», 2010, Publicacions i Edicions de la Universitat de Barcelona. [27] Sueldo Ingeniero en España «Talent Website», 2023. [En Línea] Available: https://es.talent.com/salary?job=ingeniero

Universidad Rey Juan Carlos

# Publicación

Todo el proyecto se puede consultar y descargar desde GitHub

- **Aplicación Matlab**

  https://github.com/Elenadr/Matlab_Vectorscope

- **Código fuente Android**

  https://github.com/Elenadr/MyVectorscope

  **Apk**

  https://github.com/Elenadr/MyVectorscope/blob/master/my_vectorscope.apk

- **Memoria**

  https://github.com/Elenadr/Memoria_TFG

# ¿Preguntas?

# ¡Gracias!