UNIVERSIDAD CATOLICA ANDRES BELLO.
FACULTAD DE INGENIERIA.
ESCUELA DE INGENIERIA INFORMATICA
PUERTO ORDAZ – ESTADO BOLIVAR.
CATEDRA: INTERACCION HUMANO COMPUTADOR
SEMESTRE: V

SkorCam

ANGEL PEÑA
XAVIER BECKLES
GABRIELA ROZAS

PROF. JESUS LAREZ

CIUDAD GUAYANA, ENERO 2016

Introducción

La visión por computadora es un campo relativamente joven en el mundo de las ciencias informáticas. Al estar relacionado con otros campos como la inteligencia artificial, el procesamiento de imágenes, geometría, estadística, física y probabilidades, puede llegar a ser una ciencia muy útil al momento de resolver problemas del mundo real.

Este proyecto se basa en reconocer y tocar la melodía formada por un conjunto de signos musicales contenidos dentro de un pentagrama de 150cm x 70cm colocado en una superficie blanca, mediante una imagen obtenida desde una cámara digital y usando algunas de las técnicas que engloba la visión por computadora.

Para poder tocar una nota musical se necesitan 3 cosas: la altura de la cabeza de la nota y la altura de cada línea del pentagrama para indicar el tono y el tipo de nota para indicar el tiempo que sonara. Esos tres datos, son básicamente los tres desafíos que presenta el proyecto.

El tipo de nota, en particular, está dado por la forma de la figura musical. El hecho de tener que reconocer y clasificar que tipo de nota presenta cierta figura obliga a usar aprendizaje de máquina, una rama de la inteligencia artificial que permite a un computador clasificar datos nuevos a partir de conocimiento previo.

Pero al momento de leer varias notas también se debe tomar en cuenta la posición en el eje X de cada nota, para así poder tocarlas ordenadamente de izquierda a derecha, como indica la notación musical occidental.

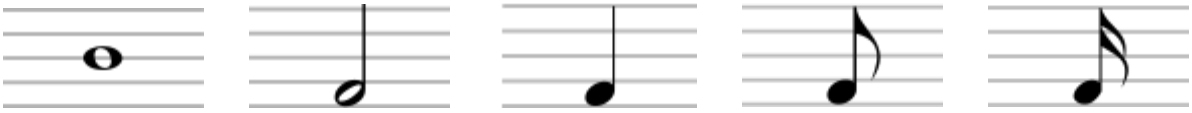
Marco teórico

Partitura musical

Una partitura es un documento que indica cómo debe interpretarse una composición musical usando un lenguaje propio llamado sistema de notación. La partitura consta de un pentagrama, formado por cinco líneas y cuatro espacios, sobre el cual se ubican los símbolos que representan los componentes musicales de la obra escrita en ella. La comprensión de las partituras requiere una forma especial de alfabetización, la capacidad de leer notación musical. La palabra partitura en griego es 'skor'.

Figura musical

Una figura musical indica al músico o compositor la duración de la nota que está representando en el pentagrama.



Redonda	**Blanca**	**negra**	**corchea**	**semi-corchea**
4 Tiempos	2 Tiempos	1 Tiempo	½ Tiempo	¼ Tiempo

Los valores de las figuras no son absolutos sino proporcionales en duración a las otras notas. La negra es la que define el tiempo, durando 'un tiempo' y es la referencia para asignarles la duración a las demás figuras musicales. Es decir, se le asigna una duración a la negra, la blanca durara 2 veces esa duración y la corchea la mitad de la duración asignada.

Pentagrama musical

El pentagrama es el lugar donde se escriben las notas y todos los demás signos musicales en el sistema de notación musical occidental. Está formado por cinco líneas y cuatro espacios o interlíneas. Las líneas son horizontales, rectas y equidistantes.

Las notas musicales se representan mediante figuras musicales que indican la duración del sonido y su ubicación en una línea o un espacio indica una determinada altura o tono. Así pues, la cabeza de nota puede ser colocada en una línea, es decir, con el centro de su cabeza de nota de intersección de una línea; o bien en un espacio, es decir, entre las líneas tocando las líneas superior e inferior. Las líneas y espacios se numeran de abajo hacia arriba, la línea más baja es la primera línea y la línea superior es la quinta línea.

El intervalo entre las posiciones adyacentes pentagrama es un paso en la escala diatónica. No obstante, la altura absoluta de cada línea está determinada por un símbolo de clave colocada al principio del

pentagrama. La clave identifica una línea en particular como una nota específica y todas las demás notas se determinan en relación a esa línea. Por ejemplo, la clave de sol indica que en la segunda línea se sitúa la nota

Visión por computadora

La visión por computadora es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un computador.

Los seres humanos usan sus ojos y cerebros para comprender el mundo que los rodea. Un computador no puede hacer eso, y es aquí donde entra la visión por computadora, la cual trata de producir el mismo efecto para que las computadoras puedan percibir y comprender una imagen o secuencia de imágenes y actuar según convenga en una determinada situación.

Al ser una disciplina científica, a la visión por computadora le concierne la teoría detrás de los sistemas artificiales que extraen información de imágenes.

Procesamiento digital de Imágenes

Extraer los datos necesarios de una imagen no es para nada fácil, los datos que se buscan pueden estar ocultos en dicha imagen. Antes de extraer información de una imagen, esta debe pasar por un proceso de pre procesamiento, con el fin de resaltar las características deseables de dicha imagen y ocultar o disminuir las características que son indeseables. Este proceso consta de un conjunto de técnicas o 'filtros' que son aplicadas en la imagen para obtener un resultado que sea más adecuado para una aplicación específica mejorando ciertas características de la misma que posibilite efectuar operaciones del procesado sobre ella.

Los principales objetivos que se persiguen con la aplicación de filtros son:

- Realizar cambios de colores a la imagen: cambiar el modelo de color de la imagen.
- Suavizar la imagen: reducir la cantidad de variaciones de intensidad entre píxeles vecinos.
- Eliminar ruido: eliminar aquellos píxeles cuyo nivel de intensidad es muy diferente al de sus vecinos.
- Realzar bordes: destacar los bordes que se localizan en una imagen.

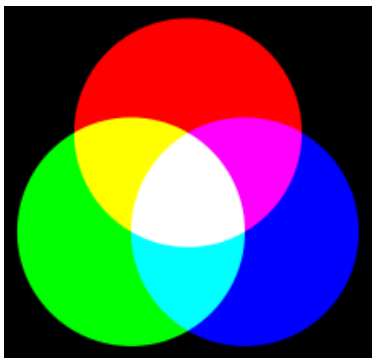
Modelos de color

Un modelo de colores es un modelo matemático abstracto que permite representar los colores en forma numérica, utilizando típicamente tres o cuatro valores numéricos que representan cierta información cromática. Hay muchos modelos de colores, pero los más comunes son los siguientes:

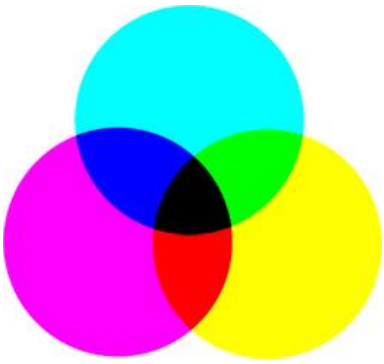
a) **RGB (Red, Green, Blue)** es un modelo de color aditivo, es decir, cuanto más rojo, verde y azul se agregue, más se parecerá el color al blanco. Cuando se mezcla la misma cantidad de rojo, verde y azul, siempre se obtiene un gris neutro. Para oscurecer un color, debe quitar la misma cantidad de los tres colores. El valor de cada "canal" (rojo, verde o azul) puede ir desde 0 (sin color) hasta 255 (color con la máxima saturación).

b) **CMYK (Cyan, Magenta, Yellow, Black)** se basa en los tres colores primarios: cian, magenta y amarillo. Éstos se denominan *colores sustractivos* porque cuanto más color se agrega, más se acerca al negro (contrario al RGB).

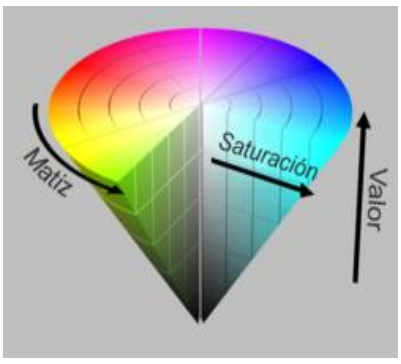
c) **HSV (Hue, Saturation, Value)** o **HSB (Hue, Saturation, Brightness)** Muchas personas consideran el modelo de color HSV más intuitivo porque define los colores en función del matiz, la luminosidad y la saturación (valor). Para especificar un color, puede seleccionar su matiz en un espectro de arco iris, seleccionar su saturación (la pureza del color) y establecer su luminosidad (de claro a oscuro). El rojo vivo es un color brillante muy saturado. Los tonos pastel, como el rosa claro, son menos saturados. El matiz se especifica en grados (de 0 a 360 grados) y la saturación y la claridad se especifican en porcentajes de 0 hasta 100 por ciento. Todo color HLS con cero saturaciones es un gris neutro.



(a)RGB



(b) CYMK



(c) HSV

El modelo HSV es el más usado al momento de reconocer ciertos colores en una imagen. Esto es debido a que a que los componentes R, G Y B del color de un objeto son afectados directamente por la cantidad de luz que llega al objeto. Por el contrario solo el componente V (y quizás el S) de un color en HSV son afectados por la luz, mientras que el componente H que indica el color primario del objeto -sin el brillo u opacidad que crea el color negro- probablemente varíe muy poco en varios escenarios de luminosidad.

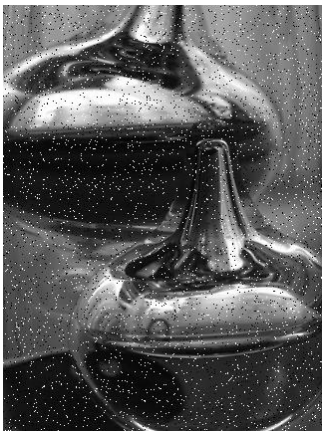
En pocas palabras, si se trata de detectar un color especifico definido en modelo RGB, cualquier parte con sombra o luz dentro del objeto tendrán características (valores del color) muy diferentes las otras partes del objeto, mientras que si se define ese color en HSV, la luz afectara mucho menos al valor de H, facilitando el reconocimiento.

Ruido

El ruido digital es la variación aleatoria (que no se corresponde con la realidad) del brillo o el color en las imágenes digitales producido por el dispositivo de entrada. Está universalmente aceptado que, así como, en la imagen analógica el grano era aceptable e incluso estético, en la fotografía digital el ruido es antiestético e indeseable.

Existen distintos tipos de ruido, pero el más común en las imágenes digitales es el ruido de “Sal y Pimienta”, en el cual los píxeles de la imagen son muy diferentes en color o intensidad a los píxeles circundantes. El hecho que define este tipo de ruido es que el pixel ruidoso en cuestión no tiene relación alguna con los píxeles circundantes. Generalmente, este tipo de ruido, afectará a una pequeña cantidad de píxeles de la imagen. Al ver la imagen, encontraremos puntos blancos sobre puntos negros o puntos negros sobre puntos blancos, de ahí el término sal y pimienta.

Imagen sin ruido



Sal y pimienta

Operaciones morfológicas en procesamiento de imágenes

Las operaciones morfológicas son operaciones simples que generalmente se realizan a imágenes binarias, utilizadas con la finalidad de resaltar características deseables de ella (o las figuras que contiene) o de ocultar características indeseadas, como el ruido en la imagen.

Las operaciones básicas son erosión y dilatación, de las cuales se derivan otras más como la apertura, cierre, gradiente, etc.

Erosión

La idea básica de la erosión es desechar los límites o bordes de la figura que contiene los pixeles en 1 (blanco). Lo que ocurre es que todos los pixeles que estén cerca de los límites de la figura serán descartados dependiendo del tamaño del elemento estructural que se use para definir la operación. Por lo que el grosor de la región blanca disminuye en la imagen. Esta operación es muy útil al momento de despegar dos objetos levemente conectados o remover pequeños conjuntos de pixeles de ruido blanco, pues al ser su área muy pequeña, si se eliminar sus bordes, simplemente desaparece la figura indeseada.

Imagen original

Erosión

Dilatación

Es lo opuesto a la erosión. La región blanca de la imagen incrementa dependiendo del tamaño del elemento estructural que se le aplique. Es útil para unir partes separadas de un objeto.

Imagen original

Dilatación

Cuando se quiere eliminar el ruido blanco de una imagen binaria, se usa una operación compuesta por erosión y dilatación llamada **apertura.** Esta operación es una erosión seguida de una dilatación. Esto se debe a que la erosión remueve todo el ruido blanco, pero además de eso, también encoje a nuestro

objeto. Así que se aplica una dilatación. Ya que el sonido ha sido eliminado totalmente por la erosión, no va a volver con la dilatación, pero el área de nuestro objeto se incrementara, dejándolo a su tamaño original.

Imagen original erosión dilatación

Reconocimiento digital de objetos

El Reconocimiento digital de objetos es la tarea para encontrar e identificar objetos en una imagen o secuencia de video usando computadoras. Los humanos reconocemos una multitud de objetos en imágenes con poco esfuerzo, a pesar del hecho que la imagen del objeto puede variar un poco en diferentes puntos de vista, en diferentes tamaños o escala e incluso cuando están trasladados o rotados.

No obstante, es un desafío para una computadora lograr esto. La computadora trabaja con datos e información numérica, y aunque nuestro cerebro es muy bueno reconociendo un objeto sin importar, por ejemplo, su tamaño, para un computador un mismo objeto con diferentes dimensiones es simplemente dos objetos distintos.

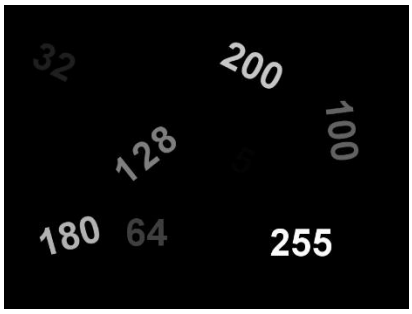
Existen muchísimas herramientas y técnicas que ayudan al computador a sobreponer sus limitaciones y si son usadas correctamente se puede aumentar muchísimo la precisión del reconocimiento, sin importar factores externos que normalmente influirían negativamente.

Thresholding

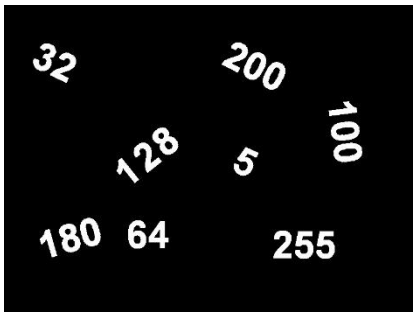
Thresholding es una operación cuya finalidad es segmentar gráficos, es decir separar los objetos de una imagen que nos interesen del resto.

Al ser un método de segmentación, se trata de asignar cada píxel a un cierto grupo, llamado comúnmente "Threshold", y asignarle uno de dos valores (1 o 0) para obtener una imagen binaria con todos los pixeles que están dentro del threshold asignados con un valor, y todos los demás con el valor contrario. La imagen que se debe segmentar, está compuesta por valores numéricos (uno o más valores

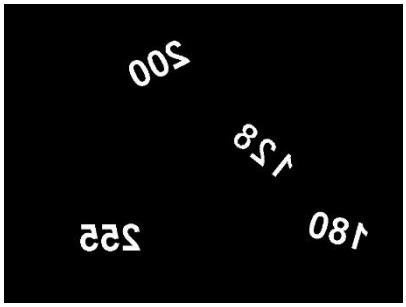
de color para cada píxel). La pertenencia de un píxel a un cierto segmento se decide mediante la comparación de su nivel de gris (u otro valor unidimensional) con un cierto valor umbral o threshold.



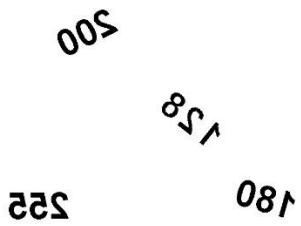
(a)Imagen original



(b) Thresh = 0



(C)Thresh = 127



(d) Threshold inverso

Teniendo una imagen de entrada (a) en escala de grises y con sus números teniendo un valor de gris igual al número, al aplicarle la operación threshold con thresh=0, cualquier pixel con un valor mayor que 0 (o negro absoluto) será asignado al conjunto de threshold, al cual se le asignara el valor de 1 (blanco), dando como resultado la imagen binaria (b). Si ahora, se cambia el valor thresh a 127, al conjunto de los pixeles que se convertirán en uno solo entraran los pixeles con valor mayor a 127, dando como resultado la imagen (b), en la cual los números menores a 127 quedaron en 0 y los mayores quedaron en 1. La imagen (d) es la operación inversa de threshold con un valor de thresh=127.

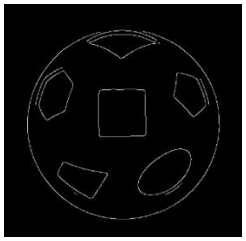
Detección de ejes y bordes

La detección de ejes y bordes son herramientas fundamentales en el procesamiento de imágenes y en visión por computadora, particularmente en las áreas de detección y extracción de características. La detección de ejes tiene como objetivo la identificación de puntos en una imagen digital en la que el brillo de la imagen cambia drásticamente o, más formalmente, tiene discontinuidades. Uno de los algoritmos

de detección de ejes más usados es el de Canny, el cual fue desarrollado por John Canny en 1986 y es capaz de detectar una amplia cantidad de tipos de ejes en imágenes.

Básicamente, un borde en una imagen puede ser definido como una secuencia de ejes, es decir conjunto de pixeles sucesivos cuyos pixeles vecinos tienen un brillo drásticamente diferente al suyo. Estas variaciones fuertes de la intensidad corresponden a las fronteras de los objetos visualizados.

Imagen original



Detección de ejes de Canny

Es importante notar que aunque el algoritmo de Canny funciona bien con imágenes en escala de grises, es recomendable aplicar primero la operación de threshold a la imagen original, para así crear una imagen binaria. Debido a que en una imagen binaria, los cambios de luminosidad son extremadamente bruscos, la detección de bordes es mucho mejor que en una imagen en escala de gris.

Transformada de Hough

La Transformada de Hough es una técnica para la detección de figuras simples en imágenes digitales. Con ella es posible encontrar todo tipo de figuras que puedan ser expresadas matemáticamente, tales como rectas, circunferencias o elipses. Como primer paso, es recomendable usar un detector de bordes para obtener los puntos de la imagen que pertenecen a la frontera de la figura deseada.

El objetivo de la transformada de Hough es realizar agrupaciones de los puntos que pertenecen a los bordes de posibles figuras a través de un procedimiento de votación sobre un conjunto de figuras parametrizadas.

Hough para detectar líneas

Hough para detectar círculos

Momentos de una imagen

Un momento de una imagen es una propiedad que describe la figura en una imagen. Existen muchos tipos de momentos como el área, la orientación, grosor o centro de masa. Estas propiedades son útiles al momento de identificar diferentes figuras en una imagen ya que dos objetos iguales en área –por ejemplo- serán iguales si se usa ese momento para comparar.

Como es de esperarse, comparar figuras por sus momentos no necesariamente traerá los mejores resultados, por ejemplo, hay muchos objetos que tienen la misma área, mismo centro de masa y aun así son diferentes. Es posible calcular momentos que son invariantes respecto a la traslación o la escala. Los más comunes son los momentos Hu, un grupo de 7 momentos que, usados en conjunto, describen una figura independientemente de su rotación, tamaño y traslación.

Un descriptor de una figura es simplemente un conjunto de características que describen a esa figura, con el fin de compararla con otras.

Aprendizaje de maquina

El aprendizaje de máquinas es una rama de la inteligencia artificial cuyo objetivo es desarrollar técnicas que permitan a las computadoras *aprender*. De forma más concreta, se trata de crear programas capaces de generalizar comportamientos a partir de una información no estructurada suministrada en forma de ejemplos. Es, por lo tanto, un proceso de inducción del conocimiento. Existen muchos algoritmos de aprendizaje de máquina, los cuales se clasifican en:

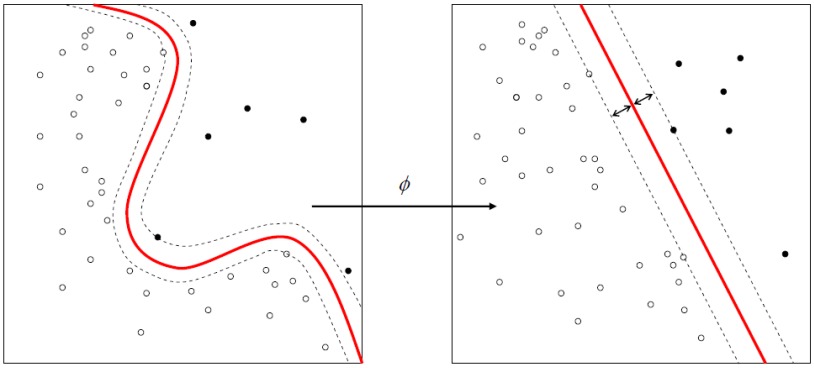
- **Aprendizaje supervisado** La base del conocimiento del sistema está formada por un grupo de ejemplos etiquetados que alimentan a la máquina antes de introducirle entradas reales.
- **Aprendizaje no supervisado** Todo el proceso de aprendizaje se lleva a cabo sobre un conjunto de ejemplos formado tan sólo por entradas no etiquetadas al sistema. El sistema se debe encargar de reconocer lo patrones y etiquetar las entradas.
- **Aprendizaje semisupervisado** una mezcla de los dos algoritmos anteriores. Se tiene en cuenta los datos marcados y los no marcados.
- **Aprendizaje por refuerzo** Su información de entrada es la retroalimentación que obtiene del mundo exterior como respuesta a sus acciones, aprendiendo a base de ensayo-error.
- **Aprendizaje multi-tarea** Métodos de aprendizaje que usan conocimiento previamente aprendido por el sistema de cara a enfrentarse a problemas parecidos a los ya vistos.

SVM (máquina de soporte de vectores)

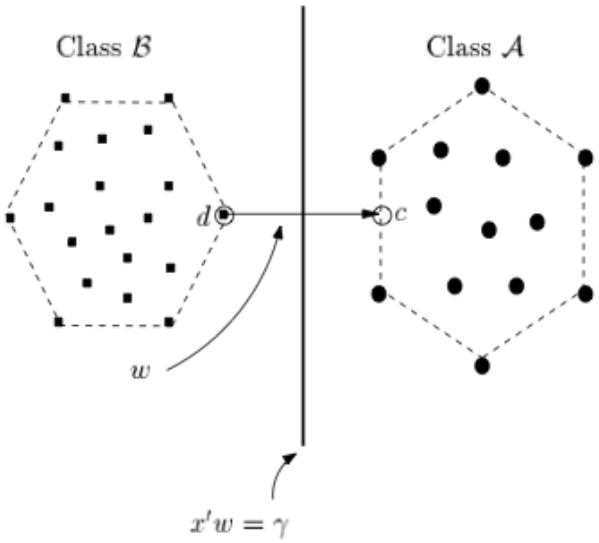
Las SVM son una serie de métodos de aprendizaje supervisado usados para clasificación y regresión. Los algoritmos de SVM usan un conjunto de ejemplos de entrenamiento clasificado en dos o más categorías para construir un modelo que prediga si un nuevo ejemplo pertenece a una u otra de dichas categorías.

Un hiperespacio es una forma de espacio geométrico que puede poseer más de cuatro dimensiones y un hiperplano es una extensión del concepto de plano. Es decir es una región que divide a un hiperespacio de una dimensión mayor en dos mitades separadas. En un espacio unidimensional (como una recta), un hiperplano es un punto: divide una línea en dos líneas. En un espacio tridimensional, un hiperplano es un plano corriente: divide el espacio en dos mitades. Este concepto también puede ser aplicado a espacios de cuatro dimensiones y más.

Si se tienen dos clases o grupos de vectores contenidos dentro de un hiperespacio, la svm divide dicho hiperespacio mediante un hiperplano para separar los vectores y así poder clasificar nuevos vectores.



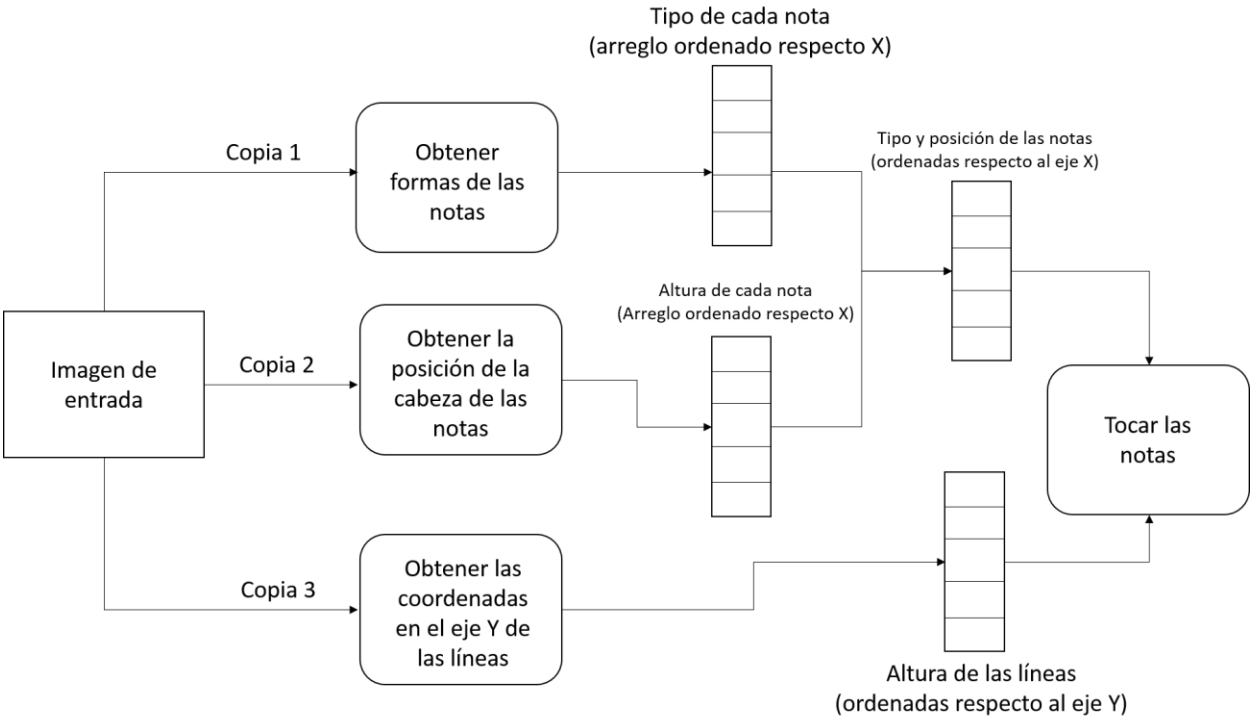
A diferencia de otros clasificadores, una SVM trata de encontrar la mejor separación del hiperespacio. Esto lo logra buscando los puntos más cercanos entre sí de clases diferentes, a los cuales se les llama los 'vectores de soporte'. Una vez encuentra los puntos con la menor distancia entre sí, la SVM traza una línea que los conecta (w) y luego traza una línea que intersecta a w por la mitad y es perpendicular a ella.



El hecho de que la SVM funcione con planos y vectores de N dimensiones, permite a la SVM clasificar imágenes y figuras. Si obtenemos los descriptores de las figuras de entrenamiento, por ejemplo los 7 momentos Hu, obtendremos vectores en R7 <a,b,c,d,e,f,g>. Si la SVM puede clasificar vectores de N dimensiones, fácilmente podemos usar una para construir un hiperespacio de 8 dimensiones que contiene los vectores en R7, y dividirlo por un hiperplano óptimo, separando a los vectores según su clase.

Diseño de la solución

El programa se puede dividir en tres problemas principales: la detección de líneas, de círculos y la clasificación de figuras musicales. Así que la solución planteada es un programa modular, el cual trabaja cada uno de estos problemas por separado, obteniendo así la información necesaria para resolver cada uno.



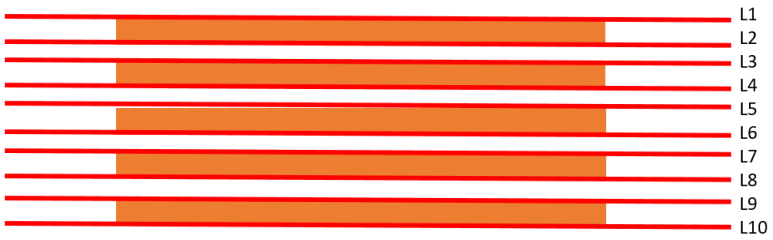
Básicamente se obtiene la imagen desde la cámara de la computadora y se crean 3 imágenes que serán una copia de esa entrada, las cuales serán procesadas por separado.

La primera copia se usa para clasificar los tipos de figuras musicales de color azul que hay en la imagen y de ella se obtiene una lista de figuras detectadas, la cual contiene el tipo de cada una (redonda, blanca, negra o corchea) y con su centro de masa. Esta lista esta ordenada respecto a la posición en el eje X de los centros de masa de cada figura.

La segunda copia se usa para detectar los círculos de color azul en la imagen. Cada figura musical tiene una ‘cabeza’ que se coloca en una línea o espacio del pentagrama para indicar su tono. La altura que realmente interesa al momento de tocar una nota, no es la altura de su centro, sino la altura de su cabeza. Esta cabeza es circular y del mismo color que la nota, es decir, azul. De este proceso de detección se obtiene una lista con los centros de los círculos, ordenados respecto a su posición en el eje X.

En estas dos listas está contenida toda la información necesaria acerca de las notas, es decir, su y la altura de su cabeza. Al estar ordenadas ambas listas de izquierda a derecha, con una simple operación aditiva es posible obtener una lista ordenada con objetos de tipo ‘nota’ el cual contiene su duración y su altura.

La tercera imagen es procesada para obtener la altura todas las líneas naranjas en la imagen. Ya que hay cuatro espacios en el pentagrama, en la imagen habrá 10 líneas de referencia:



Estas líneas obtenidas serán las que se comparen con la altura de una nota para asignarles un tono. Por ejemplo, si la altura de cierta nota es mayor a la de L8 y menor a la de L9, a la nota se le asigna un ‘sol’, pues está justo en la segunda línea del pentagrama. La salida de este procesamiento es una lista con 10 alturas, ordenadas verticalmente, las cuales representan las 10 líneas.

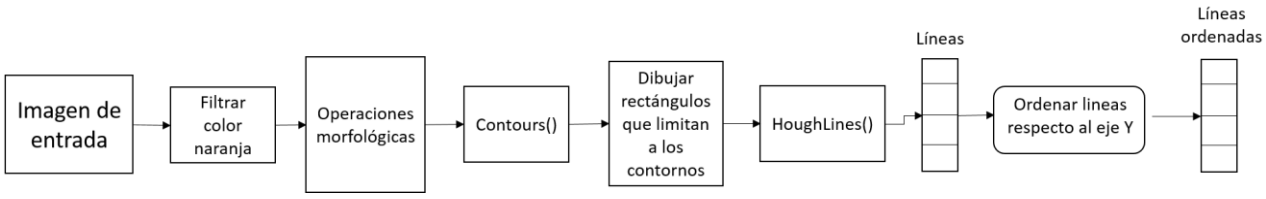
Después de tener la lista con las alturas de las líneas y la lista con las notas, estas se tocan secuencialmente. Es decir, por cada nota en la lista, se tocara un tono que depende de la altura de la nota con respecto a las líneas en la segunda lista y por un tiempo que depende del tipo de figura.

Implementación de la solución

El desarrollo de la aplicación se logró usando la librería OpenCV (open computer visión), la cual contiene herramientas de procesamiento de imágenes y aprendizaje de máquina, la librería SFML (Simple and Fast Multimedia Library), para la reproducción de las notas y Microsoft Visual Studio 2013 como entorno de desarrollo.

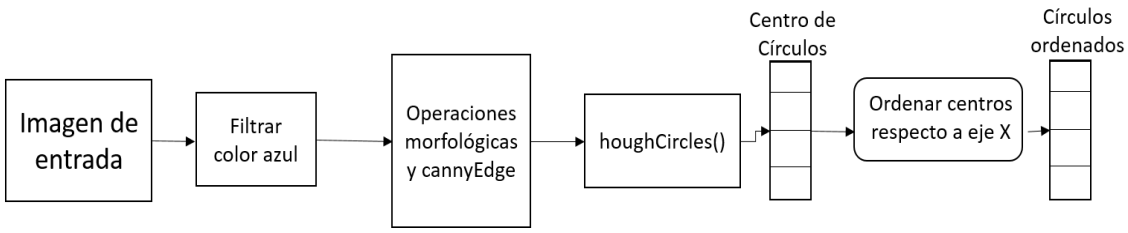
Para la parte de la detección de líneas naranjas en la imagen de entrada, proveniente de la cámara web, el proceso consiste cambiar el color de la imagen de entrada a HSV, lo cual facilita la filtración de los colores. Luego se aplica threshold a la imagen para obtener una imagen binaria, con todo lo naranja con valor de 1, a la cual se le aplican operaciones morfológicas (erosión y dilatación) para reducir el ruido en la imagen.

Luego se consiguen todos los contornos de la imagen, lo cual dará un contorno de cada línea y se creara otra imagen en la cual se dibujaran los rectángulos que encierran dichos contornos. Esto con la finalidad de obtener 5 rectángulos paralelos y uniformes, lo cual mejora la precisión del algoritmo de Hough para detectar las líneas. El algoritmo de Hough devuelve un arreglo con dos puntos que se usan para construir la línea detectada. Luego, este arreglo se ordena para obtener 10 líneas ordenadas verticalmente.



La detección de los círculos azules es bastante parecida a la de las líneas. La imagen de entrada se convierte a HSV, se filtra por el color azul y se obtiene la imagen binaria en la cual se buscaran los círculos.

Luego a esta imagen se le procesa aplicando operaciones morfológicas y filtros gaussianos, para luego aplicar Canny edge detection, lo cual facilita muchísimo al algoritmo de Hough la detección de los círculos en la imagen. Los centros de los círculos detectados son guardados en un arreglo y son ordenados respecto al eje X.



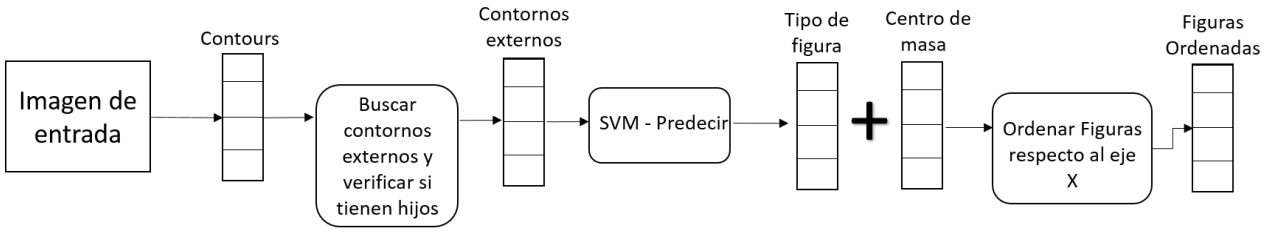
Para detectar y reconocer las figuras de la imagen de entrada se usa una SVM, La cual fue entrenada con aproximadamente 400 imágenes por cada figura a detectar –redonda, blanca, negra y corchea- . El entrenamiento de la SVM no se realiza con la imagen como tal, sino que de la imagen se calculan todos los contornos de las figuras de color azul, obteniendo un arreglo con un contorno por cada figura. Finalmente, a cada contorno se le calculan los momentos Hu, los cuales se usaran como descriptores para entrenar a la SVM.

Al momento de recibir la imagen de entrada, de ella se obtienen los contornos de las figuras de color azul. Luego se determina que contornos son externos, es decir, que no están incluidos dentro de otro contorno más grande. Debido a que ningún contorno interno puede representar una nota musical, estos son descartados. Los que si son contornos externos se pasan a la función predecir() de la SVM la cual clasifica la figura en una de tres: , , o circular.

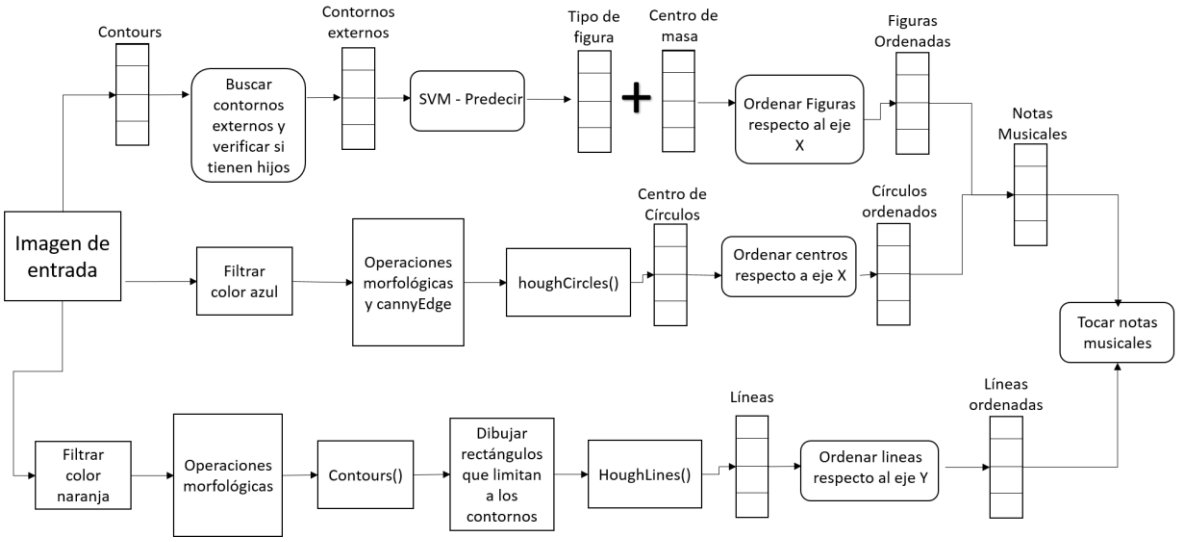
Luego se verifica una simple condición:

- Si tiene hijos, y el contorno es circular es una redonda.
- Si tiene contornos hijo y su contorno externo tiene forma , la nota es una blanca.
- Si no tiene contornos hijo y su contorno externo tiene forma , la nota es una negra.
- Teniendo hijos o no, si su contorno externo tiene forma de  , es una corchea.

Luego de ser clasificadas, las figuras se van metiendo en un arreglo, el cual se ordena de izquierda a derecha según los centros de masa de las figuras.



Teniendo los arreglos con las figuras y con los círculos, se hace una operación aditiva sobre ellos para crear un arreglo ordenado horizontalmente que contiene la altura y el tipo de las notas. Luego con esta información y el arreglo de las líneas, se tocan secuencialmente las notas musicales.



Detalle de las funciones

Void GuardarPuntosDetectados(std::vector<Vec2f> lines)

Utiliza el vector lines (que es el resultado de houghlines ()) para construir una matriz que contiene la información de todas las líneas detectadas. Esta información es P1x, P1y, P2x, P2y y una bandera que se usa para verificar si esa línea se intersecta con otra.

bool intersection(Point2f o1, Point2f p1, Point2f o2, Point2f p2)

Verifica si dos líneas se intersectan o no. retorna verdadero si se intersectan y falso si no lo hacen. Las dos líneas están representadas por dos puntos cada una L1 = (o1, p1) y L2 = (o1, p2).

Void DepurarLineas()

Determina cuales de las líneas de la matriz que contiene las líneas detectadas se intersectan entre sí. A aquellas que lo hagan, se les coloca en 1 la bandera de la posición 4 del vector para denotar su intersección con una línea anterior. Luego se pasa una (y solo una) de las muchas líneas que se intersectan entre sí a otro vector. Las líneas contenidas en ese vector son las líneas que de verdad se van a dibujar y las cuales se usaran para tocar las notas

Void DibujarLineas(Mat &matriz)

Dibuja todas las líneas contenidas en el vector Puntos_LineasD en la matriz pasada como parámetro

Void GuardarCirculosDetectados(std::vector<Vec3f> circles)

Utiliza el vector circles (que es el resultado de houghCircles ()) para construir una matriz que contiene la información de todas los círculos detectados.

Void DibujarCirculos(std::vector<Vec3f> circles)

Dibuja todos los círculos detectados por la función HoughCircles().

Void shellsortLineas(int size)

Ordena las líneas de forma ascendente respecto a su posición en Y

Void shellsortCircles(int size)

Ordena los circulos respecto a la pocision X de su centro

Void shellsortFiguras(Figura f[], int size)

Ordena las figuras respecto a la posición X de su centro de masa

Void MorfologiaLinea(Mat &thresh)
Void MorfologiaCirculo(Mat &thresh)
Void MorfologiaFigura(Mat &thresh)

Aplican operaciones morfológicas a la matriz pasada como parámetro. Estas operaciones dependen de si se quiere detectar líneas, círculos o figuras.

Void TocarNotas(NotaMusical notas[] , int CantNotas)

Se le pasa un arreglo de notas musicales, el cual tocara secuencialmente dependiendo de su altura y su tipo. Acá se usa la librería SFML para ayudar a reproducir los archivos mp3 que contienen las notas.

Conclusión

La teoría musical y la computación son ciencias que no están comúnmente relacionadas. Ambas están llenas de complejidades y tecnicidades. El hecho de unir estas dos ramas del conocimiento humano presenta retos que pueden ser superados usando técnicas de visión por computador.

Un proyecto como este puede ser la base para un modelo de aprendizaje de la notación musical en personas jóvenes, presentando una forma interactiva de introducir a la persona al mundo de la teoría musical. Además brinda una mirada al mundo de la visión por computadora, la cual puede ser la solución a muchos problemas del mundo real.