



# Visión Artificial

Ángel Hernández Mejías  
MIF  
[angelidpe@hotmail.com](mailto:angelidpe@hotmail.com)



# Índice

<b>Portada .....</b>	<b>Pág. 1</b>
<b>Índice .....</b>	<b>Pág. 2</b>
<b>1. ¿Quién puede entender este tutorial? .....</b>	<b>Pág. 3</b>
<b>2. ¿Qué es la Visión Artificial? .....</b>	<b>Pág. 3</b>
<b>3. ¿Qué necesitamos antes de empezar? .....</b>	<b>Pág. 3</b>
<b>4. ¿Qué es una imagen en realidad?.....</b>	<b>Pág. 4</b>
<b>5. ¿De qué va el programa que vamos a hacer? .....</b>	<b>Pág. 5</b>
<b>6. Tabla de controles a introducir en el formulario .....</b>	<b>Pág. 6</b>
<b>7. El Código .....</b>	<b>Pág. 7</b>
<b>8. Capturas de pantalla.....</b>	<b>Pág. 8</b>



# Tutorial básico de Visión Artificial con tu WebCam.

Dim VisionArtificial as NoImposible

## 1. ¿Quién puede entender este tutorial?

Espero que lo pueda entender cualquiera con unas nociones básicas de programación, no quiero hacer este tutorial un testamento, de modo que obviaré cuestiones relativas a la sintaxis de VB, porque doy por supuesto que se sabe algo básico de este lenguaje... si este es tu primer contacto con la programación... mejor empieza por otro tutorial, con este solo ibas a conseguir liarte.

## 2. ¿Qué es la visión artificial?

Visión artificial: Dícese de la tecnología que dota a... un cacharro de ojos... Estos ojos van a pasar la información a un PC que calcule lo que nosotros le pidamos. En este tutorial vais a ver como calcular un punto de un color específico. Seguramente haya muchas definiciones más serias de lo que es la Visión Artificial, pero me temo que son esas las que hacen a los principiantes como yo, que pensemos que es una tecnología imposible... así que de momento es todo lo que necesitamos saber de la tecnología.

## 3. ¿Qué necesitamos tener antes de empezar?

-Una WebCam... básico, no hace falta que sea nada especial, de hecho yo estoy usando mi cámara digital, una Minolta DX... nada del otro mundo.

-Un entorno de programación que soporte Visual Basic. Yo estoy usando Visual Studio 2005, así que puede haber algunas diferencias mínimas con otros entornos, en especial si no soportan VB.Net, pero no tendrán mucha importancia, ni serán imposibles de solucionar.

-Un lugar con colores nada parecidos al que pretendemos buscar. En mi caso es una esquina de mi cuarto, en la que predominan los tonos salmón y naranjas.

-Algo con un color que no pegue nada con el color que predomine en el sitio anterior... en mi caso un bote de desodorante de mi color favorito, azul eléctrico. Como os imagináis no pega nada con el color de fondo de mi habitación. La explicación más o menos técnica de esto es que necesitamos que la imagen tenga mucho contraste entre el fondo, y el color que buscamos.

-Necesitamos ante todo, muchas ganas de programar, muchas ganas de equivocarnos, y mucha imaginación.

Nada mas, yo creo que no es demasiado, ¿no?



#### 4. ¿Qué es una imagen en realidad?

Este es el concepto que nos va a abrir la mente, de modo que después de entenderlo algunos diréis... ¡Anda! Ahora lo entiendo todo, ya se por donde van los tiros.

Bien, al tema. Antes de nada vamos a empezar por las imágenes en blanco y negro: Una imagen en blanco no es ni mas ni menos que una matriz de números entre 0 y 255 que definen la claridad, siendo 0 = Negro, y 255 = Blanco; los valores intermedios serán tonos mas o menos oscuros de Gris.

Ahora imaginaos un cuadrado de 10 píxeles por 10 píxeles toooodos ellos negros; pues tendremos una matriz de 10 filas y 10 columnas en la que todos los valores son negros, es decir 0

Esta es la tabla:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Esta es la imagen:



¿Ridícula eh? Pues prometo que ahí están comprimidos los 100 valores que he puesto en la tabla haceos una idea de la cantidad de datos que habrá en muestras de 330 píxeles por 240... esto hace que las cosas sean lentas, claro.

Bien, ahora ya sabemos lo que es una imagen en B/N. Pues si en vez de tener una tabla tenemos 3, tenemos tres valores por cada píxel.

¿Qué ocurre si cada una de las tablas no da valores en tonos de gris, sino en tonos de Rojo, Verde y Azul, y además las montamos una encima de otra? Pues que tenemos colores suficientes como para engañar a nuestros ojos y hacerles ver millones de colores. Para comprobar de lo que estoy hablando solo hay que coger una lupa y mirar la pantalla en una zona blanca, veréis claramente puntitos de estos colores. Esto pasa porque esa "celda" o píxel tiene los valores 255 de Rojo, 255 de Verde, 255 de Azul.

Ahora ya entendemos lo que son las imágenes en color, y cómo funcionan en cuanto a valores numéricos. ¿A que no era tan difícil? ¿Y a que ya os imagináis por donde van los tiros?



## 5. ¿De qué va el programa que vamos a hacer?

Antes de nada debemos saber que vamos a emplear una librería llamada WebCam.dll, que os podéis bajar de <http://www.yellowpark.net/angel>, miraos la sección Download. Esta librería nos da acceso a nuestra WebCam de un modo muy sencillo, porque nos proporciona un control que llamaremos WebCam1, con el que tenemos acceso o todas las propiedades de la WebCam.

El programa que vamos a diseñar va a seguir el siguiente proceso:

- Captura la imagen en directo de la webcam.
  
- Pasamos la imagen a un objeto bitmap (ImgOriginal) en el que compondremos la nueva imagen. Esto lo vamos a hacer porque tendremos más control sobre los píxeles, y podemos manejarla sin que se vea en la pantalla, lo que acelera bastante la velocidad de cálculo.
  
- Recorreremos las matrices de píxeles, preguntándole si el píxel actual es “azul” o no. No todo lo que parece azul lo es, y hay veces que no puedes limitarte a preguntarle ¿este píxel tiene un valor muy alto en la tabla de azules? Porque la respuesta con tonos blancos puede ser sí, sencillamente porque el blanco puro se compone con valores máximos de Rojo, Verde, y Azul, de modo que lo que haremos será preguntar ¿El azul de este píxel es más grande que el verde y el rojo considerando un grado de tolerancia? Si la respuesta es sí, ahora podemos decir que en ese píxel predomina el azul, y no falseamos la respuesta con blancos. Obviamente hay sistemas mucho más complejos para detectar color, pero este es muy sencillo, y comprensible, de modo que para empezar nos vale de sobra.
  
- Cada vez que el sistema anterior nos responda Sí, pintaremos ese píxel de color... amarillo, por ejemplo, y cuando sea que no, lo pintaremos de negro. Recordad que no estamos trabajando con una imagen visible, sino con el objeto ImgOriginal.
  
- En un par de variables guardaremos la posición x e y del último píxel pintado de amarillo, es decir, del último píxel que en la realidad es azul. Estos valores se Irán sustituyendo cada vez que se encuentre un píxel azul, así que en definitiva, lo que vamos a tener aquí es la posición del ultimo píxel considerado azul.
- Igual que hicimos antes, ahora pintaremos ese último píxel azul de color... rojo, y además los valores de las coordenadas x e y se los pasamos a un par de barras desplazables que nos marquen el punto.
  
- Pasamos la imagen de ImgOriginal a un control que nos proporcione esa capacidad de visualización, por ejemplo un PictureBox, con lo que ya estamos viendo la imagen que había en ImgOriginal, como un dibujo, en el que habrá puntos negros en los no azules, amarillos en los azules, y un punto mas grande rojo marcando el último píxel azul.
  
- Este proceso se repetirá cada 200ms, lo que nos va a proporcionar una imagen casi en tiempo real, del cálculo que hemos hecho en los pasos anteriores.

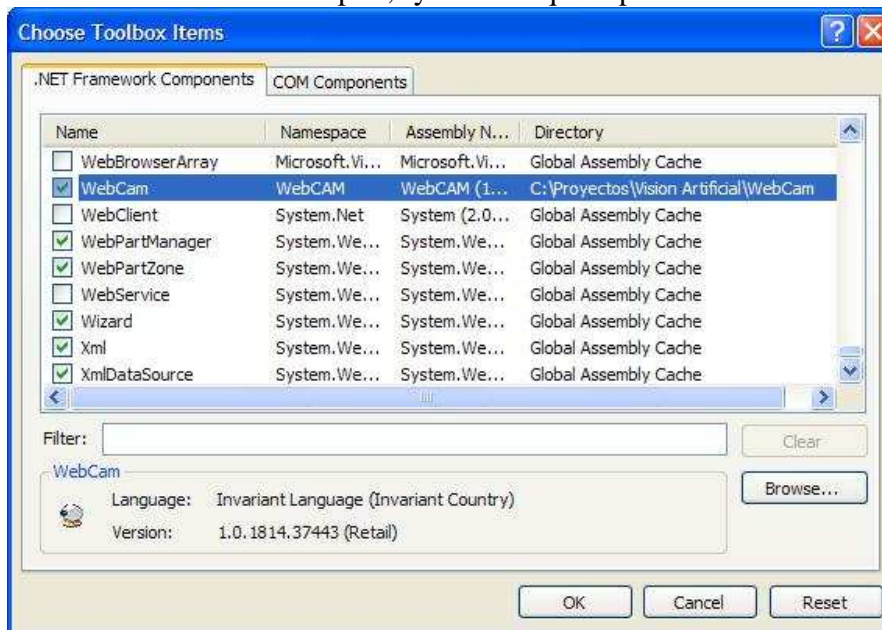


## 6. Tabla de controles a introducir en el formulario:

Orden	Control	Nombre	Valores configurados
1	Label	LabDirecto	Text = "WebCam directo"
2	Label	LabResultado	Text = "Resultado"
3	WebCam	WebCam1	Ninguno *
4	TrackBar	TrackBarY	Maximum = 250 Orientation = Vertical RightToLeft = No Size = 45; 250 TickFrequency = 10
5	PictureBox	PicResultado	Size = 330; 250
6	Button	ButMuestra	Text = "Muestra"
7	Button	ButBuscaAzul	Text = "Busca azul"
8	Button	ButDetener	Text = "Detener"
9	Button	ButConfiguracion	Text = "Configuración"
10	TrackBar	TrackBarX	Maximum = 330 Orientation = Horizontal RightToLeft = No Size = 350; 45 TickFrequency = 10
11	Timer	TimMuestreo	Interval = 200

\*Para insertar el control WebCam tenéis que incluirlo como componente en Visual Studio, ya que viene de la librería WebCam.Dll. Los pasos a seguir son estos:

1. Copiáis la librería WebCam.dll en el directorio donde guardéis el proyecto.
2. Menú Tools en la pantalla de Visual Studio (Recordad que trabajo con 2005)
3. Choose Toolbox ítems...
4. Según se abre veréis un botón que pone Browse, pues le dais y buscáis la librería WebCam.dll.
5. Le dais a aceptar, y veréis que aparece WebCam entre los controles, así:



Ahora ya esta la webcam entre los controles normales



## 7. El Código:

Recordad que este código es para que sea comprensible, de modo que no es el más efectivo, pero lo que quiero es que sepamos de lo que va, a partir de aquí es nuestra imaginación quien debe mandar. Perdonad que modifique el margen, pero creo que queda algo mas claro. No hagáis caso a los saltos de línea de los `Private Sub` son demasiado largos, así que van en una sola línea. Generadlos con un doble clic sobre el control.

```
Public Class Form1 'Esto lo genera solo

    'Subrutina del botón de toma de muestra
    Private Sub ButMuestra_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ButMuestra.Click
        WebCam1.Start() 'Activa la WebCam
        PicResultado.Image = WebCam1.Imagen 'Pasa una muestra al para ver el resultado
    End Sub

    'Subrutina del botón Detener
    Private Sub ButDetener_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ButDetener.Click
        WebCam1.Stop() 'Simplemente detiene la camara y deja los últimos fotogramas
        TimMuestreo.Enabled = False 'También detendremos el temporizador
    End Sub

    'Subrutina del botón de configuración
    Private Sub ButConfiguracion_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ButConfiguracion.Click
        WebCam1.Configuracion() 'Abre una pantalla de configuración de la webcam
    End Sub

    'Subrutina del botón de Búsqueda de Azul
    Private Sub ButBuscaAzul_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ButBuscaAzul.Click
        WebCam1.Start() 'Activa la WebCam
        TimMuestreo.Enabled = True 'Activa el timer que nos generará el muestreo
    End Sub

    Private Sub TimMuestreo_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TimMuestreo.Tick
        'Pasamos el fotograma actual al PictureBox para ver lo que hay
        PicResultado.Image = WebCam1.Imagen

        'Declaramos un par de objetos Bitmap para trabajar sin que se vea.
        'Declaremos dos para facilitar la comprensión del código
        Dim ImgOriginal As Bitmap = PicResultado.Image
        Dim ImgResultante As Bitmap = PicResultado.Image

        'Declaramos una variable para cada color. Son de tipo entero porque
        'Serán valores entre 0 y 255
        Dim Rojo, Verde, Azul As Integer

        'Necesitaremos dos variables para guardar las coordenadas del último punto azul
        Dim UltimoAzulX, UltimoAzulY As Integer

        'La tolerancia nos permitirá no ser tan estrictos con la selección de color
        Dim Tolerancia As Integer

        'Las variables x e y servirán para guardar coordenadas actuales
        Dim x, y As Integer
```



```
'Pondremos una tolerancia de 5 unidades de color.
'Cada uno debe ajustarla según sus necesidades por el contraste de la imagen
Tolerancia = 5

'Ahora recorreremos la imagen del objeto ImgOriginal como si fuese una
'matriz, de modo que iremos por las horizontales en orden.
'Para no perder mucho tiempo en el análisis de cada pixel, haremos
'que los salte de 10 en 10 tanto en vertical como en horizontal
For y = 0 To ImgOriginal.Height - 1 Step 10
    For x = 0 To ImgOriginal.Width - 1 Step 10
        'Guardamos los valores de los colores del pixel (x, y)
        Rojo = ImgOriginal.GetPixel(x, y).R
        Verde = ImgOriginal.GetPixel(x, y).G
        Azul = ImgOriginal.GetPixel(x, y).B

        'Ahora preguntaremos si en el pixel actual predomina el
        'azul sobre el rojo y el verde, teniendo en cuenta la tolerancia
        If Azul >= Rojo - Tolerancia Then 'Si hay más azul que rojo sigue
            If Azul >= Verde - Tolerancia Then 'Si hay mas azul que verde sigue

                'si llega hasta aquí es porque el pixel es suficientemente azul
                'así que lo pintamos con un punto de 2x2 en amarillo.
                'Ésta es una manera de pintar cuadrados, pero luego vemos otra.
                ImgResultante.SetPixel(x, y, Color.Yellow)
                ImgResultante.SetPixel(x + 1, y, Color.Yellow)
                ImgResultante.SetPixel(x + 1, y + 1, Color.Yellow)
                ImgResultante.SetPixel(x, y + 1, Color.Yellow)
                'Guardamos la posición x y de este pixel azul, si es el último
                'se quedará aquí, porque no habrá más azules
                UltimoAzulX = x
                UltimoAzulY = y
            End If

        Else
            'Si entra aquí es porque el pixel no era suficientemente azul
            'así que pondremos un punto de 1 pixel en negro.
            ImgResultante.SetPixel(x, y, Color.Black)
        End If

    Next 'Fin del análisis del pixel x, y
Next 'Fin de la línea horizontal pasa a la siguiente (y+1)

'Pintaremos un cuadrado de 4 x 4 en las coordenadas del último pixel azul
For y = UltimoAzulY To UltimoAzulY + 4
    For x = UltimoAzulX To UltimoAzulX + 4
        'Estos if sirven para no salirnos de la tabla (de la imagen), porque
        'si lo hacemos tendremos un error
        If y < ImgResultante.Height Then
            If x < ImgResultante.Width Then ImgResultante.SetPixel(x, y, Color.Red)
        End If
    Next 'ya está pintado el pixel con coordenadas UltimoAzulX y UltimoAzulY, a por otro
Next 'ya están los cuatro horizontales, a por la siguiente fila

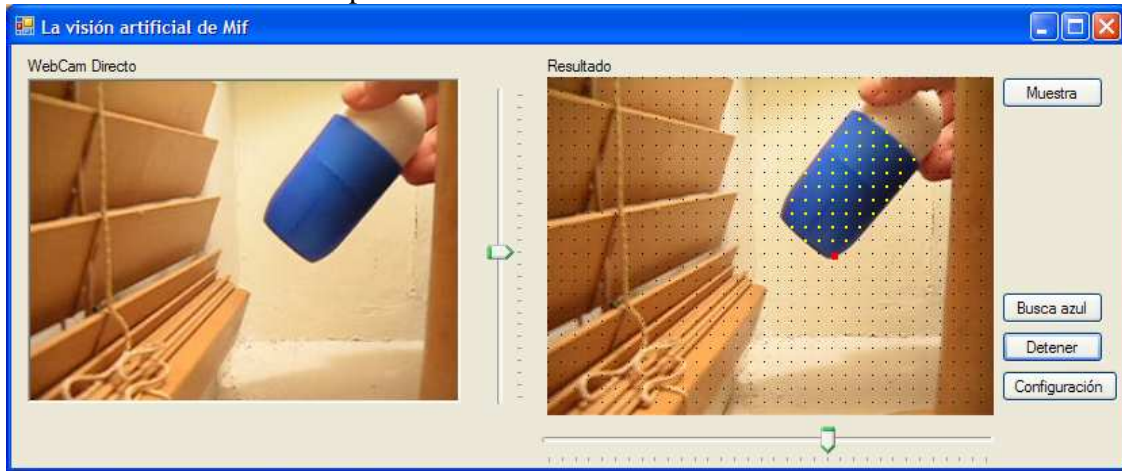
'Pasamos el objeto ImgResultante al PictureBox, para ver el análisis en la pantalla
PicResultado.Image = ImgResultante

'Ponemos los valores de las coordenadas último pixel azul en las barras
TrackBarX.Value = UltimoAzulX
TrackBarY.Value = 250 - UltimoAzulY

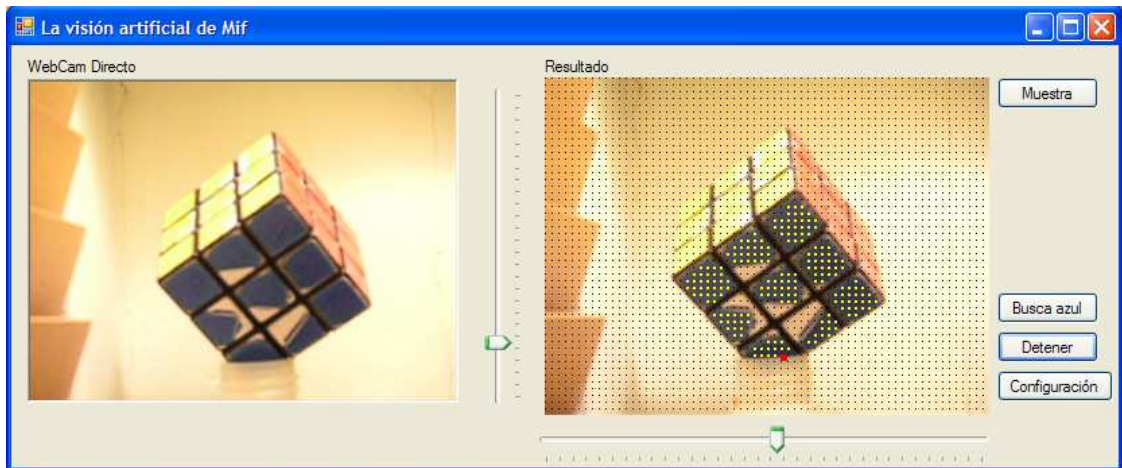
End Sub 'Fin del análisis, en menos de 200ms vendrá otro
End Class 'Fin del programa.
```



## 8. Capturas de pantalla Con mi desodorante de pruebas.



## Con mi cubo de rubik.



## Y otra vez con mi cubo de rubik.

