Face detection via CoreImage in FileMaker

If you like , come to my booth at FileMaker DevCon and ask me about using CoreImage's detectors to find faces, text, rectangles and QRCodes on a picture. You can check whether faces are like in this picture on the right.

So the example database has a script to detect faces and than color them in yellow with some dots for eyes and mouth. Check out our example script for this:

```
Set Field [ Core Image Detection::Result ; MBS( "CoreImage.Detect"; Core Image
Detection::Image; Core Image Detection::Type; "Smile¶EyeBlink" ) ]
# Show areas
If [ MBS("IsError") = 0 ]
 Set Variable [ $json ; Value: MBS( "JSON.Parse"; Core Image Detection::Result ) ]
 Set Variable [ $count ; Value: MBS( "JSON.GetArraySize"; $json ) ]
 Set Variable [ $image ; Value: MBS( "GMImage.NewFromContainer"; Core Image
Detection::Image ) ]
 Set Variable [ $imageHeight ; Value: MBS( "GMImage.GetHeight"; $image) ]
 If [ $count > 0 ]
  # for loop
  Set Variable [ $index ; Value: 0 ]
  Loop
   Set Variable [ $item ; Value: MBS( "JSON.GetArrayItem"; $json; $index) ]
   # draw area
   Set Variable [ $x ; Value: MBS( "JSON.GetPathItem"; $item; "x" ; 1  ) ]
   Set Variable [ $y ; Value: MBS( "JSON.GetPathItem"; $item; "y" ; 1  ) ]
   Set Variable [ $w ; Value: MBS( "JSON.GetPathItem"; $item; "width" ; 1  ) ]
   Set Variable [ $h ; Value: MBS( "JSON.GetPathItem"; $item; "height" ; 1  ) ]
   Set Variable [ $y ; Value: $ImageHeight - $y -$h /* coordinates from CoreImage are
swapped vertically */ ]
   #
   Set Variable [ $r ; Value: MBS( "GMImage.SetFillColor"; $image; "#FFFF00" ) ]
   Set Variable [ $r ; Value: MBS( "GMImage.DrawRectangle"; $image; $x; $y; $x+$w; $y+
$h ) ]
   # left eye
   Set Variable [ $hasLeftEyePosition ; Value: MBS( "JSON.GetPathItem"; $item;
"hasLeftEyePosition" ; 1  ) ]
   If [ $hasLeftEyePosition = 1 ]
    Set Variable [ $x ; Value: MBS( "JSON.GetPathItem"; $item; "leftEyePositionX" ; 1  ) ]
    Set Variable [ $y ; Value: MBS( "JSON.GetPathItem"; $item; "leftEyePositionY" ; 1  ) ]
    Set Variable [ $y ; Value: $ImageHeight - $y /* coordinates from CoreImage are
swapped vertically */ ]
    #
```

```
    Set Variable [ $r ; Value: MBS( "GMImage.SetFillColor"; $image; "blue" ) ]
    Set Variable [ $r ; Value: MBS( "GMImage.DrawRectangle"; $image; $x-20; $y-20;
$x+20; $y+20 ) ]
   End If
   # right eye
   Set Variable [ $hasRightEyePosition ; Value: MBS( "JSON.GetPathItem"; $item;
"hasRightEyePosition" ; 1  ) ]
   If [ $hasRightEyePosition = 1 ]
    Set Variable [ $x ; Value: MBS( "JSON.GetPathItem"; $item; "rightEyePositionX" ; 1  ) ]
    Set Variable [ $y ; Value: MBS( "JSON.GetPathItem"; $item; "rightEyePositionY" ; 1  ) ]
    Set Variable [ $y ; Value: $ImageHeight - $y /* coordinates from CoreImage are
swapped vertically */ ]
    #
    Set Variable [ $r ; Value: MBS( "GMImage.SetFillColor"; $image; "blue" ) ]
    Set Variable [ $r ; Value: MBS( "GMImage.DrawRectangle"; $image; $x-20; $y-20;
$x+20; $y+20 ) ]
   End If
   # mouth position
   Set Variable [ $hasMouthPosition ; Value: MBS( "JSON.GetPathItem"; $item;
"hasMouthPosition" ; 1  ) ]
   If [ $hasMouthPosition = 1 ]
    Set Variable [ $x ; Value: MBS( "JSON.GetPathItem"; $item; "mouthPositionX" ; 1  ) ]
    Set Variable [ $y ; Value: MBS( "JSON.GetPathItem"; $item; "mouthPositionY" ; 1  ) ]
    Set Variable [ $y ; Value: $ImageHeight - $y /* coordinates from CoreImage are
swapped vertically */ ]
    #
    Set Variable [ $r ; Value: MBS( "GMImage.SetFillColor"; $image; "red" ) ]
    Set Variable [ $r ; Value: MBS( "GMImage.DrawRectangle"; $image; $x-50; $y-20;
$x+50; $y+20 ) ]
   End If
   # next
   Set Variable [ $index ; Value: $index + 1 ]
   Exit Loop If [ $index >= $count ]
  End Loop
 End If
 Set Variable [ $r ; Value: MBS( "JSON.Release"; $json ) ]
 Set Field [ Core Image Detection::Output ; MBS( "GMImage.WriteToPNGContainer";
$image; "test.png" ) ]
End If
```

As you see we just get back a JSON block with an array of items, where we loop through and draw each area in yellow and if available we draw eyes and mouth. The JSON looks like this with one face:

```
[
  {
    "rightEyeClosed" : false,
    "mouthPositionX" : 279.5625,
    "hasRightEyePosition" : true,
    "leftEyePositionY" : 1606.5,
    "hasLeftEyePosition" : true,
    "trackingFrameCount" : 0,
    "hasMouthPosition" : true,
    "type" : "Face",
    "x" : 66.9375,
    "mouthPositionY" : 1452.9375,
    "y" : 1350.5625,
    "trackingID" : 0,
    "hasFaceAngle" : true,
    "width" : 401.6250,
    "height" : 401.625,
    "leftEyePositionX" : 212.625,
    "hasSmile" : false,
    "leftEyeClosed" : false,
    "rightEyePositionY" : 1606.5,
    "hasTrackingID" : false,
    "hasTrackingFrameCount" : false,
    "faceAngle" : -7,
    "rightEyePositionX" : 366.1875
  }
]
```


This can be very useful for some databases as we can leverage Apple's libraries to find faces, detect QRCodes, find text and rectangle areas. For the text areas, you can pass those to OCR functions later.

PS: CIDetectorMBS class does the same in Xojo.