1. CONTENIDOS DEL REPOSITORIO

- Detector: contiene el proyecto de Visual Studio con el código principal del sistema (a excepción del entrenamiento). Dejo todos los archivos de Visual Studio por si quisierais abrirlo directamente en el IDE.
- SVM_training: contiene el proyecto de Visual Studio correspondiente al entrenamiento de los SVM.
- Other files
    - Sample files: archivos de muestra, útiles si queréis compilar y probar el sistema en Visual Studio, para entender mejor el código
        - Training_samples: imágenes para entrenar, con su correspondiente ground truth y su archivo activations
        - Detection_samples: imágenes sobre las que testear un sistema ya entrenado. En la carpeta pre_trained_model existe un modelo para probar directamente la detección sin necesidad de entrenamiento previo
    - Deployment needed files: todas las dll necesarias para ejecutar el programa en una máquina sin Visual Studio y OpenCV. Lo dejo aquí para el futuro por si hacen falta.
    - runDetector: un simple batch script para ejecutar el programa fácilmente sin tener que escribir en línea de comandos. También lo dejo simplemente para el futuro o cuando haga falta.
- PropertySheetDebugx64_2412.props y PropertySheetReleasex64_2412.props: son los archivos de configuración de los proyectos de Visual Studio.

2. EL CÓDIGO: QUÉ ES LO IMPORTANTE Y POR DÓNDE EMPEZAR

2.1 Proyecto SVM_training

Como podréis comprobar, el proyecto consiste en un único archivo **SVM_training_main.cpp que incluye todas las funciones necesarias,** bastante simple. Además, encontraréis una carpeta llamada haar_source que contiene varios archivos que están incluidos en el main. Estos archivos han sido extraídos de la propias librerías de OpenCV para poder calcular features HAAR (os recuerdo que no existe un interfaz propio en OpenCV que te permita hacerlo directamente). Ni se os ocurra abrir y e intentar navegar por esos archivos porque perderéis el tiempo y probablemente entendáis poca cosa – simplemente fiaros de mí, todo lo que tenéis que saber es que realmente sirven para calcular features HAAR y ya está.

2.2 Proyecto Detector

El proyecto principal es algo más complejo y tiene varias clases/archivos que interactúan entre sí:

- haar_source: exactamente la misma carpeta que en el caso anterior. De nuevo, ignorarla completamente.
- frameProcessor.h: incluye una clase abstracta para facilitar cierto procesado. Podéis ignorarla por completo (incluso puedo que desaparezca en el futuro).
- videoProcessor.h y videoProcessor.cpp: implementan la clase VideoProcessor, que es la que se encarga de manejar flujos de vídeo (abrir, coger cada frame, visualizar, detener el flujo, etc.). No está escrita por mí (aunque tiene algunas modificaciones por mi parte) y realmente no merece la pena que buceéis en ella. Lo único importante que debéis saber es que al ejecutar su método configure() se llama al método preprocess() de la clase Detector, y que al ejecutar su método run() se entra en un bucle infinito donde se coge cada uno de los frames de la secuencia de entrada y se llama al método process() de la clase Detector.
- main.cpp: es el punto de entrada al programa. En su método main se configuran las principales variables de las clases Detector y VideoProcessor, y se llama a la función que procesa frame a frame la secuencia de entrada (processor.run()).
- Detector.h y Detector.cpp: definen la clase principal del sistema. Como veréis hay bastantes variables y funciones, así que algunos comentarios:
    - El sistema tiene 3 modos principales de funcionamiento: GENERATION – dado un ground truth y definido un grid de puntos, realiza un procesado para generar los archivos necesarios para entrenar los SVMs (a través de SVM_training); DETECTION – dados unos SVMs ya entrenados, procede a la detección en tiempo real; TEST – es un modo suplementario al de detección,

donde se almacenan datos para realizar comparativas y se calculan ciertas métricas al finalizar una secuencia de entrada finita.

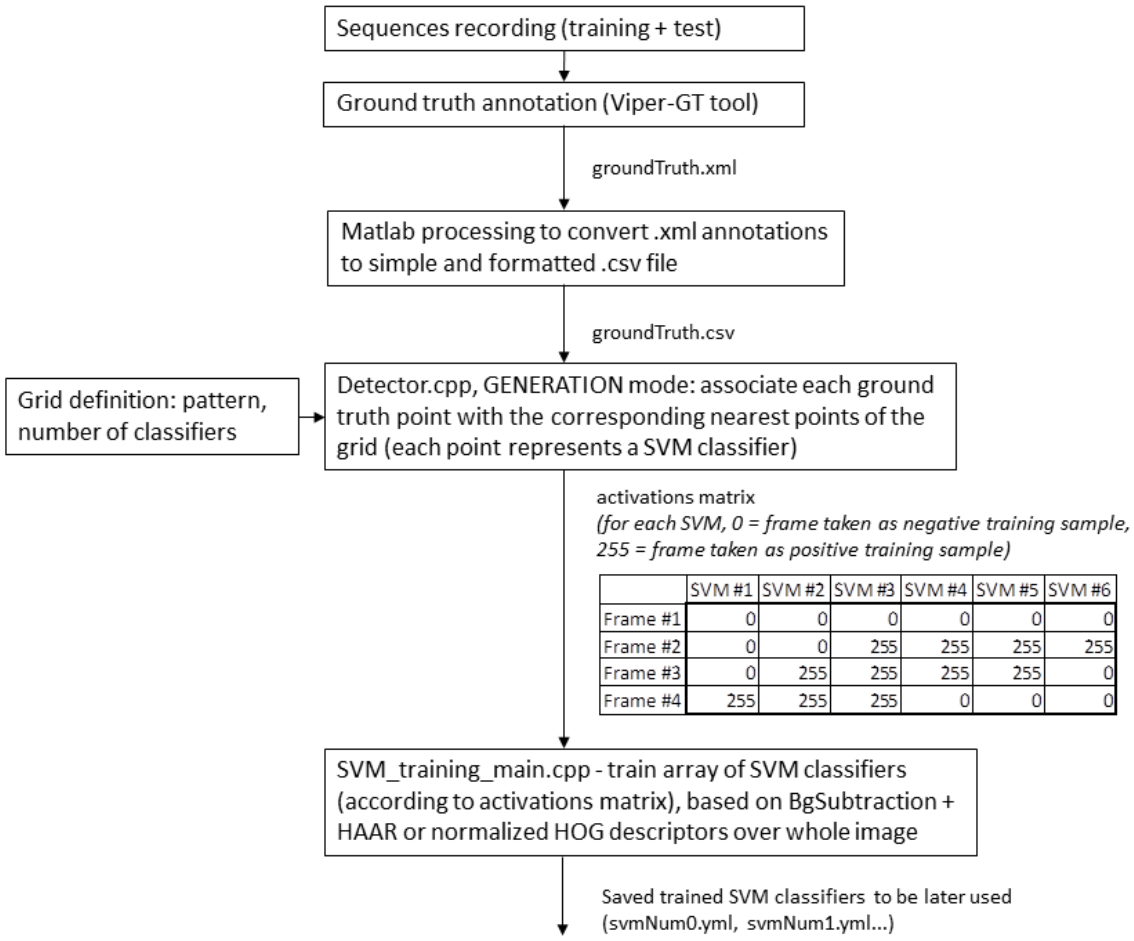
- Las funciones principales de la clase son preprocess() y process(). Cada una de ellas llama a la correspondiente subrutina según el modo del sistema (e.g. preprocessGeneration() or processDetection()). Las primeras subrutinas se encargan de inicializar el sistema y todos sus parámetros antes de comenzar el procesado cuadro a cuadro. Las segundas contienen el procesado mínimo necesario de cada cuadro (las detecciones). Todas estas funciones están en la parte baja del archivo Detector.cpp, y dado que son el punto de entrada al resto de funciones de la clase, os aconsejo empezar por ellas e ir poco a poco buceando en funciones más profundas y específicas.
- Las funciones específicas del modo TEST, las que calculan errores y métricas, aún necesitan cierta revisión y son posiblemente las que más cambios sufrirán en el futuro. Por tanto, os aconsejo dejarlas para el final.
- Existen algunas funciones que todavía están a medio desarrollar o que realmente no se utilizan pero que están ahí por comodidad o referencia. Si os encontráis con alguna, no hagáis mucho caso o preguntadme directamente.

Por tanto, después de lo dicho anteriormente, mi consejo es que en primer lugar miréis main.cpp por encima y que después vayáis directamente a Detector.cpp y sus funciones preprocess() y process().

3. SYSTEM WORKFLOW

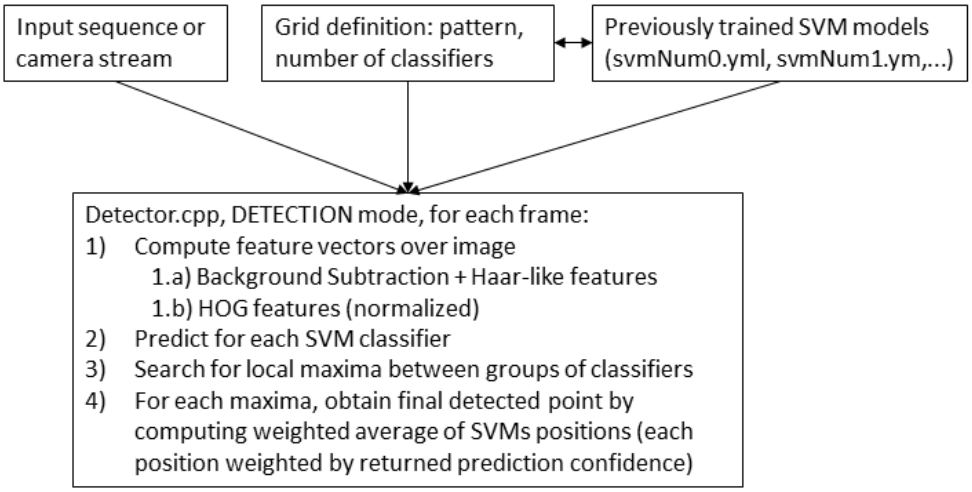
Como es cierto eso de que una imagen vale más que mil palabras, os dejo unos pequeños esquemas que muestran de forma general el workflow del sistema completo, desde que se graban las secuencias hasta que se obtienen los resultados. Espero que os sirvan para entender más fácilmente qué pasos hay que seguir para que todo funcione, y las variables más importantes involucradas.

3.1 Generation + training



	SVM #1	SVM #2	SVM #3	SVM #4	SVM #5	SVM #6
Frame #1	0	0	0	0	0	0
Frame #2	0	0	255	255	255	255
Frame #3	0	255	255	255	255	0
Frame #4	255	255	255	0	0	0

3.2 Detection



3.3 Test

Es básicamente igual al modo de detección, solo que se almacenan algunas variables extra y es necesario proporcionar archivos con información sobre ground truth y activations para hacer comparativas: