Principales problemas encontrados - soluciones

1) Adicción de layers

Al clonar los repositorios con la estructura lista para ser añadidas como layers, surge un error: “layers is not compatible with core layer…”.

Solución: este error se soluciona debido a que el repo se clona y encuentra en el branch más reciente compatible posiblemente con la versión más reciente de Yocto: se debe hacer el cambio al brnahc correspondiende de la versión en uso, en este caso sería Scarthgap.

2) Integración de dlstreamer

No se pudo compilar dlstreamer desde el código fuente en Yocto debido a errores de configuración y dependencias.

Solución: Se modificó la aplicación para funcionar solo con GStreamer estándar y OpenCV, evitando los elementos específicos de dlstreamer. Este punto queda por mejorar en futuros proyectos.

3) Rendimiento en entorno virtualizado

Reproducción extremadamente lente del video en QEMU.

Solución: Se optimizó el pipeline de GStreamer reduciendo la resolución, limitando los FPS y usando un sink más ligero, se pasó de usar ‘autovideosink’ a ximagesink’, el cual es más ligero.

4) Archivos no encontrados

La receta no encontraba correctamente el archivo de video fuente.

Solución: cuando se hace una receta y no debe compilar código solo copiar/descargar debemos tanto danrle la dirección está donde encontrar dicho recurso, ya sea en nuestro sistema de archivos de la mauqina host o la url al repositorio, por ejemplo, esto en el apartido de SCR_URI. Dentro del apartado do_install debemos específicar donde será copiado o descargado ese recurso, esta dirección es dentro de nuestra imagen.

Conclusiones y recomendaciones

OpenVINO y Gstreamer se integraron satisfactoriamente en el ecosistema Yocto, permitiendo hacer el pipeline de un video y teniendo a disposición elementos de OpenVINO y Gstreamer.

Se realizó una aplicación en python que hizo uso de Gstreamer.

Las capas personalizadas permiten adaptar aplicaciones específicas al entorno embebido, así como se realizó con la aplicación ‘run_pose.py’.

La virtualización, con herramientas como qemu, facilita el desarrollo y prueba de imágenes antes de desplegarlas en hardware real u otras plataformas virtualizadas, como las VM.

La integración de aplicaciones de CV en imágenes Yocto es viable, pero requiere especial atención a las dependencias.

Se probó la aplicación de forma exitoso en un contenedor lógico, en este caso,

Docker, para el cual se tiene ya a dispocisión una imagen base, basada en intel-dlstreamer.

Se recomienda considerar para fututos proyectos que para componentes complejos como OpenVINO, es preferible utilizar capas oficiales mantenidas en lugar de intentar compilar desde el código fuente, al ahorrar así mucho trabajo y errores a nivel de dependencias.

Considerar el uso de contenedores Docker para componentes con dependencias complejas, como un paso previo para poder observar la totalidad de dependencias a nivel de software.

Se recomienda ‘cocinar’ las recetas de forma aislada previamente a ‘cocinar’ toda la imagen, de esta forma poder aislar erroes y facilitar su correción, especialmente con layers personalizadas.

Se recomienda optimizar las aplicaciones de procesamiento de video para entornos con recursos limitados, como qemu.