Automatització d'esquemes per a la Indústria 4.0 amb Visió per Computador

Menció de Computació

Hernán Capilla Urbano

1462773

Índice

0. Introducción

Este proyecto consiste en la creación de una herramienta que automatice la creación de la renovación de un sinóptico ya existe orientado a un sistema SCADA. Con el objetivo de simplificar y reducir la carga de trabajo del ingeniero/a que esté encargado de realizar dicha tarea. De forma que, conservando la mayoría de las propiedades del sinóptico original se pueda crear una nueva versión del ya existente sinóptico pasando por alto las limitaciones que implica el anterior sistema y formato sobre el que se creó el sinóptico original en su momento. Todo esto con el fin último de reducir la cantidad de horas que el ingeniero debe dedicarle a tareas que no aportan valor al sinóptico final renovado.

Con esto dicho pues, y antes de entrar en materia, existen ciertas terminologías que debemos explicar para poder comprender de qué forma actúa esta herramienta y qué impacto tiene sobre el estado del arte actual de la materia.

Comenzamos explicando qué es un sistema SCADA y qué es un sinóptico.

Un sistema SCADA (Supervisory Control and Data Acquisition) es un sistema completo empleado en lo que conocemos como Industria4.0 que se utiliza para supervisar y controlar procesos industriales de forma remota y que está compuesto por varias partes:

- Supervisión: software que permite la visualización y gestión de datos.
- Conexión con controladores: dispositivos que se conectan con sensores y actuadores situados en la planta industrial.
- Bases de datos: empleadas para el almacenamiento de un histórico de la información recopilada.
- Interfaces de usuario: pantallas en las que los operadores pueden observar los datos recopilados en tiempo real e interactuar con ellos.

Es sobre este último apartado sobre el que nos vamos a fijar, pues un sinóptico está englobado en esta descripción. Se le llama sinóptico a la pantalla que muestra el estado de los procesos de la planta, sus datos, y sus elementos en tiempo real [fig0.a]. De esta forma facilita la comprensión del funcionamiento del proceso industrial a los operarios que allí trabajan, les permite ver en tiempo real los datos recopilados del proceso y, además, disminuye el tiempo de respuesta ante cualquier situación anómala que pueda ocurrir.

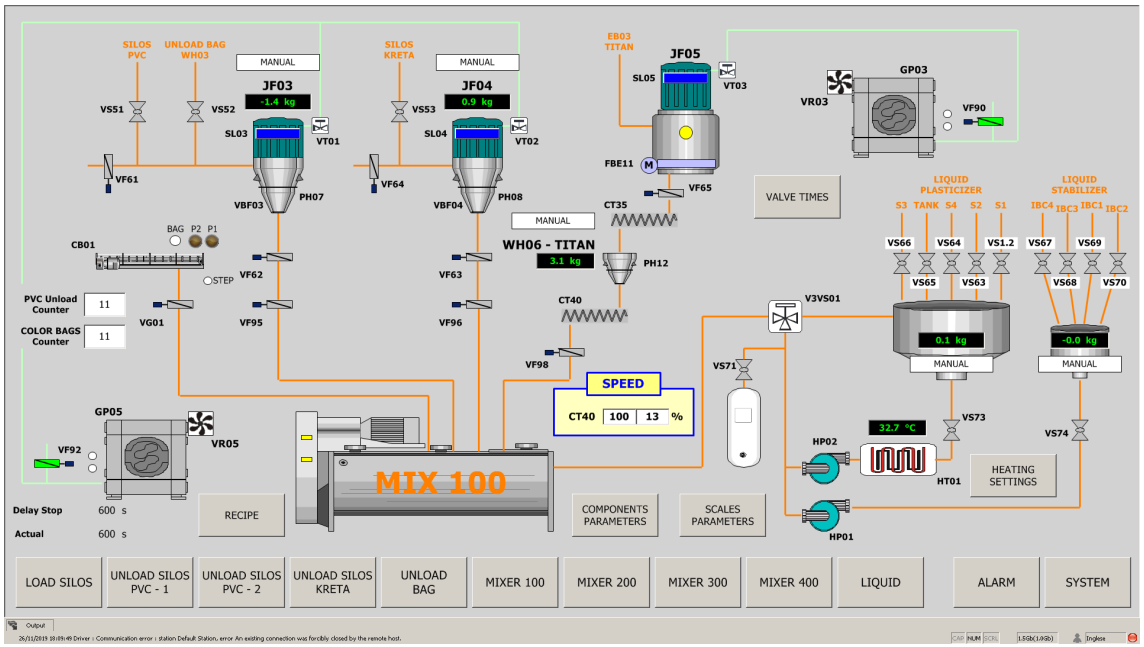

fig0.a – Ejemplo de sinóptico en un proceso industrial

Teniendo estos conceptos claros, podemos dar paso a ahondar en la pregunta de ¿cómo se crea un sinóptico? ¿Cómo pasamos de tener un PLC, una base de datos y los distintos elementos de la planta industrial a que todos formen parte de un mismo sinóptico mostrando sus datos en tiempo real?

Existen multitud de herramientas y programas que nos ayudan en la tarea de la creación de el sinóptico. Algunos ejemplos destacables de esto son TIAPortal, WinCC, PlantSCADA y SystemPlatform. Estos son solo algunos ejemplos, pero existen más programas y entornos tanto públicos como propios de la empresa para realizar esta tarea.

Lo destacable de estos es que ofrecen un entorno en el que manipular tanto las conexiones, como los elementos y sus características propias, las bases de datos e incluso editores de diagramas.

El papel del ingeniero es el de, junto con todo lo antes mencionado, combinarlo en la creación de un diagrama que actuará como sinóptico. Para este proceso el ingeniero puede abordar desde la recolección de señales que van desde PLC hasta el elemento en cuestión (bomba, motor, ventilador, etc), como el de configurar las características propias que tendrán esos elementos (variables, tipos de datos de las variables), qué variables usará dentro de la base de datos, y qué fórmulas emplearán esas variables para calcular sus valores, qué tipo de alarmas tendrá, cuándo se activarán estas alarmas; hasta el diseño y dibujo de todo este proceso industrial.

Para este primer trabajo de recolección, un ingeniero con experiencia puede llegar a dedicar decenas de horas para una sola pantalla. Hay que tener en cuenta que cada elemento tiene sus características y usos propios, y que una BombaA, puede no actuar de la misma forma que una BombaB a pesar de tener ambos el mismo elemento base. En el caso de un ingeniero sin mucha experiencia este trabajo puede llegar a superar más de cien horas en sus primeros proyectos.

Para el segundo caso, el de diseño de pantalla, nos encontramos con un trabajo también harto complejo. Si bien en el primer caso debíamos ser minuciosos al realizar el trabajo de recolección, en este segundo se ha de ser incluso más.

El principal motivo es que este será nuestra "presentación" de cara al cliente. Un cliente que, en caso de que la pantalla no muestre de forma correcta los datos anteriormente recopilados, o en caso de que haya algún error en cómo los muestra, el cliente perderá muchísimo tiempo y dinero; así como nosotros; y que puede llegar a generar problemas tanto de funcionamiento en la propia fábrica como que dé posibilidad a accidentes laborales. Es por eso por lo que este apartado es tan importante.

Hasta el momento hemos visto cómo se crean los sinópticos desde cero. Ahora que tenemos una idea aproximada de sobre qué estamos trabajando, vamos a explicar dónde actúa este proyecto y qué dificultades intenta solventar.

1. Problemática

Este proyecto está enfocado a una problemática que afecta tanto a ingenieros senior como júnior, pero, sobre todo a los ingenieros junior, que generalmente cuentan con menor experiencia y están menos preparados para este tipo de tareas de creación de sinópticos y manipulación de sus datos.

Es en estos primeros proyectos que se destinan a ingenieros poco maduros en este tipo de trabajos, que se ofrece la creación de sinópticos a partir de un modelo ya existente. Es decir, sinópticos que ya están listos y llevan en funcionamiento desde hace muchos años, quieren ser remodelados y actualizados a otras versiones u otros programas actuales.

Pongamos el siguiente ejemplo: *En mi fábrica de golosinas, llevo 25 años usando el mismo sinóptico para la línea de creación de caramelo. Este sinóptico se hizo con la tecnología más puntera de hace 25 años, y durante este tiempo mi cadena de montaje ha ido cambiando, antes tenía dos máquinas de mezcla de caramelo y ahora solo tengo una porque hace 10 años la cambié a una de tecnología puntera*.

Tengo todo este proceso industrial en una planta ubicada en Villarobledo, Albacete, y está dividida en varias secciones. Para cada sección tengo un sinóptico que muestra su respectiva línea de fabricación de cada golosina. Y cada sinóptico se creó en su momento en el entorno para SCADA's de WinCC, que era el programa más avanzado en su momento.

Pero ahora, en 2024, quiero remodelar todo el sistema de sinópticos, porque durante estos años ha habido variaciones en estas líneas. Y ahora quiero añadir más sensores que me permiten llevar un mejor control de mi producción. Es decir, los elementos ya existentes tendrán más características propias que los que tengo ahora mismo.

Así que voy a contratar a una empresa llamada DummyCorp S.L. que se encargará de renovar todo el sistema SCADA y sus respectivos sinópticos a unos nuevos y añadirle estas nuevas funcionalidades. Prescindiré de WinCC porque hoy en día ya no es el mejor programa que existe en el mercado y usaré un nuevo entorno propio de DummyCorp llamado DumDum.

Y este es el ejemplo. Esto es lo que está sucediendo hoy en día en el mercado de la Industria4.0. Empresas que hicieron sus SCADA hace años ahora los quieren remodelar y migrar de sistemas. Quieren que sus viejas pantallas se vean actualizadas y dispongan de mayor y mejor visibilidad, adaptado a las nuevas resoluciones de pantallas y con características nuevas para cada elemento.

Y es aquí donde entra este proyecto.

La dicotomía del asunto es que una parte importante de este trabajo de renovación de sinópticos es el tiempo y recursos empleados en únicamente el diseño y nuevo dibujo de la pantalla.

Las pantallas viejas no son reaprovechables, están en formatos concretos dentro de proyectos concretos en entornos concretos en sistemas operativos concretos que tienen muchos años. Así pues, su uso acaba siendo relegado al de patrón en el que basarse para poder realizar la pantalla nueva desde 0.

El problema que surge también de esto es que estas pantallas se encuentran en las propias fábricas, como es de esperar, y no están pensadas para ser manipuladas de forma cómoda. Así pues, a la hora de indagar el ingeniero en los diferentes elementos de esta pantalla para tomar notas y realizar su renovación, se tiene que emplear máquinas virtuales que emulan su sistema de uso original en la fábrica o escritorios remotos que conectan directamente con ella. Máquinas virtuales y escritorios remotos que convierten el simple hecho de abrir un archivo ya pesado de por sí en tamaño, en una tarea auténticamente farragosa. Los recursos en las máquinas virtuales o escritorios remotos, pese a ser los indicados, muchas veces no evitan el hecho de que el programa se cuelgue, o que vaya extremadamente

lento o el input lag propio de estar conectándote a un servidor con muchas capas de seguridad por encima y que se está usando a la vez en la producción de la fábrica.

Y esto es solo para empezar, porque una vez tengas abierto todo lo que necesites e identificados los elementos, tendrás que realizar el dibujo de la nueva pantalla desde cero. Tendrás que identificar cada elemento de su pantalla origen, crear uno parecido para la nueva pantalla, tomar sus dimensiones, ver cómo encaja en la nueva pantalla, asegurarte de las dimensiones de la nueva pantalla, ver que quepa todo, y que el proceso industrial es el mismo que el de la pantalla origen. Todo esto teniendo en cuenta que la máquina sobre la que estás trabajando es lenta e incómoda.

Y, como bien hemos indicado, solo para empezar a trabajar en la nueva pantalla. Puesto que, como dice el ejemplo, queremos migrar el sistema de uno antiguo a uno nuevo. Igual no queremos que los elementos, por ejemplo, las bombas, se vean como en el programa antiguo, y queremos que sean más grandes o que tengan otra forma. Obviamente querremos un resultado profesional, en el que esté todo alineado, todo con su respectiva nomenclatura y que se aproveche debidamente el espacio.

Todo esto es un trabajo que alguien tiene que hacer. Y como hemos comentado con anterioridad, tanto los perfiles júnior como a los senior tienen que enfrentarse a estas dificultades e incomodidades únicamente para empezar a trabajar.

Aquí es donde entra este trabajo. En solucionar y aligerar esta carga de trabajo que supone decenas de horas para el trabajador en estas partes iniciales de renovación de sinópticos en sistemas SCADA. Usando mediante:

- La automatización del proceso de identificación de elementos para cada pantalla de toda una planta.
- La sencilla implementación de nuevos elementos siguiendo el diagrama de la pantalla original.
- El fácil añadido de nuevas funcionalidades en forma de template modificable a estos elementos.
- La adaptación rápida tanto a rediseños como a cambios en las proporciones de la pantalla final resultante mediante uso de diagramas basados en vectores que permiten un reescaldo rápido a diferentes tamaños y resoluciones.
- La minimización de la necesidad de usar herramientas antiguas y engorrosas que entorpecen el proceso de creación.

2. Estado del arte

De todo lo mencionado anteriormente, ideamos el siguiente planteamiento acerca de puntos clave que debía de tener la herramienta que vamos a desarrollar en este proyecto. Estos puntos clave son los siguientes:

- El programa debe coger una imagen como input, y sin nada más que eso, ser capaz de realizar las siguientes tareas:
    - El programa ha de ser capaz de identificar los textos que corresponden a cada elemento. A diferencia de otras características de los elementos, los textos son visibles a simple vista desde el sinóptico. Y deben ser detectados y pertenecer a su elemento correspondiente.
    - El programa ha de identificar de forma autónoma los elementos clave del diagrama original. Ha de ser capaz de detectar las formas de los elementos y clasificarlos en el tipo de elementos que se corresponden. Ha de distinguir una bomba de una válvula, una válvula de un ventilador, un ventilador de un tanque de agua, etc.
    - El programa ha de ser capaz de modificar o añadir las funcionalidades que queremos a los elementos detectados. Un sistema basado en templates debe ser empleado para cada elemento diferente.
    - El programa ha de ser capaz de rastrear las conexiones (líneas) que unen los diferentes elementos del diagrama. Estas líneas indican el orden que sigue el proceso industrial y, pese a no ser un elemento como tal, es importante que mantengan la coherencia entre los elementos y unan únicamente a los que verdaderamente están conectados.
    - El programa debe ser sensible y estar preparado para un reescalado según las necesidades de la pantalla final. Se debe trabajar con un formato basado en vectores para que la fidelidad gráfica no se vea afectada. Se trabajará con SVG.

Con esta recopilación de los requisitos, buscamos el estado del arte de este proyecto. Proyectos realizados con anterioridad que cumplan esta función que queremos realizar o sean parecidos.

Lo que encontramos fue lo siguiente, y es que no hemos encontrado ni conocido de la existencia de proyectos empleados para la Industria4.0 ni la elaboración automática de sinópticos industriales.

Asumimos que es debido a la propia naturaleza del mundo de la Industria4.0. Es un mundo en auge y los motores propios o herramientas de empresas privadas deben

mantenerse en secreto y no ser accesibles al público. Además de esto, los sinópticos contienen información crucial de un sistema industrial complejo. La publicación o el trabajo sobre estos podría suponer pistas o información confidencial de procesos industriales que no interesa que se sepan a gran escala.

A pesar de esto, en la investigación inicial, encontramos varios proyectos usados para diferentes propósitos que nos darían una idea de hacia dónde encaminarnos para realizar la herramienta que cumpla con las ideas anteriormente mencionadas.

LIVE: Towards Layer-wise Image Vectorization

Este proyecto de la Universidad del Noroeste aplica a la problemática de la conversión de imágenes planas a SVG. Pasar de una imagen directamente y pasarla a un formato basado en vectores.

Esto lo consigue mediante la división por capas de la imagen original. Y poco a poco y capa mediante ir creando el SVG recursivamente entorno a esto, de forma que a cada iteración se vaya pareciendo más y más a la imagen original.

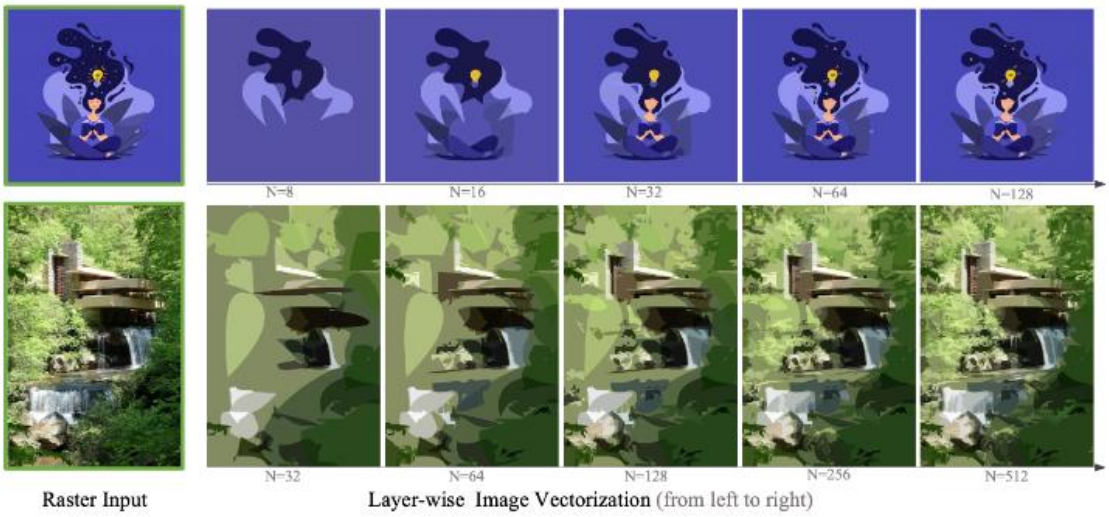

fig2.a – Proceso de vectorizado

Este proyecto encajaba con la idea con la partíamos al principio, y es que era la de procesar la imagen input del sinóptico y convertirla enteramente a SVG. El problema con esto era que los diferentes elementos del sinóptico perdían sus características, aquello que los hacía útiles de cara a la Industria4.0. Además de que el enfoque de realizarlo capa por capa no nos parecía interesante desde el punto de vista práctico.

Un sinóptico puede tener decenas de elementos, de líneas de conexiones entre elementos y también contiene textos. De nada nos sirve vectorizarlo todo si perdíamos las propiedades y no clasificábamos cada elemento por su clase.

Además, los sinópticos son prácticamente siempre imágenes en 2D, no hay necesidad de realizar una división por capas. Sino que es más interesante el detectar directamente qué tipos de elementos clave sostiene la imagen.

Raster-to-Vector: Revisiting Floorplan Transformation

Este Proyecto de la Universidad de Washington, trata la problemática de convertir una imagen a un plano vectorizado y después a uno 3D en una casuística arquitectónica [fig2.b].

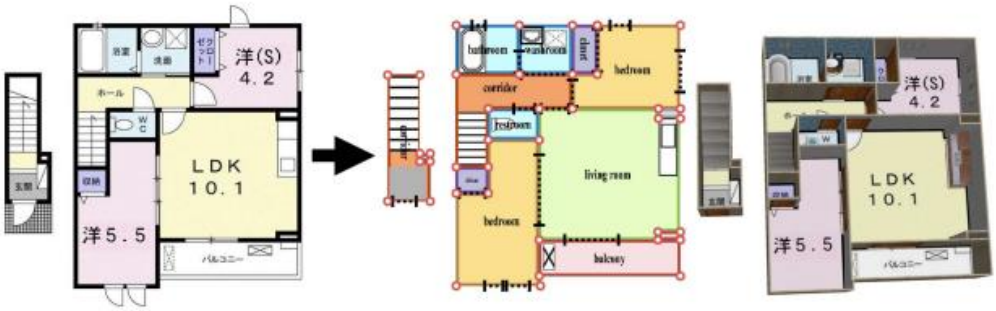

fig2.b – De imagen a plano vectorizado y modelo 3D

En él se explica que emplean un modelo para detectar los puntos de unión de los muros para identificar cada habitación, coger sus dimensiones y después pasarlo a un formato basado en vectores [fig2.c].

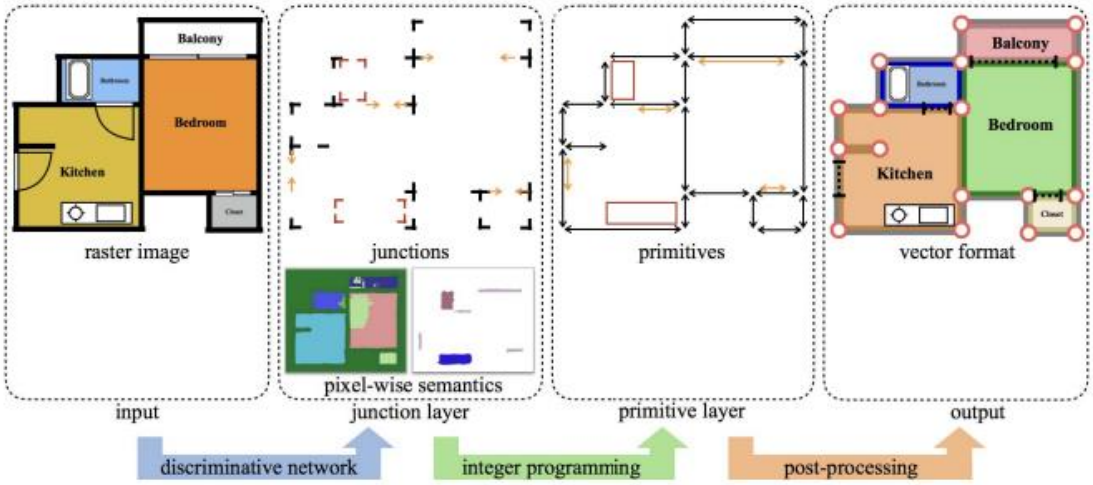

fig2.c – Proceso de detección de puntos clave

Para realizar este proceso de detección, crean un conjunto de formas para cada uno de los puntos clave que se pueden llegar a encontrar [fig2.d].

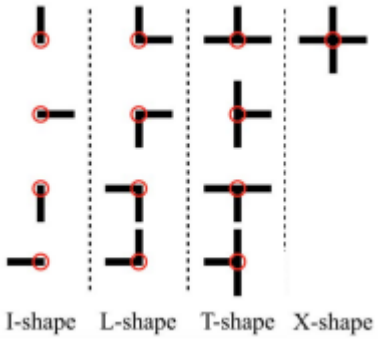

fig2.d – Conjunto de formas clave

Constatan que empleando estas formas clave y viendo la diferencia entre las coordenadas entre ellas, pueden definir dónde están los muros que se conectan entre estos puntos clave.

La forma que tienen para definir cuándo ha acabado el proceso es una vez todos los puntos clave queden conectados entre sí. Pues esa será la dimensión de la casa y todo lo que quede fuera será exterior. Habiendo terminado así el proceso.

Puntos a destacar de este proyecto:

- Uso de una imagen representando un plano como entrada.
- Detección de puntos clave y formas.
- Desintegración por partes del proceso.
- Uso de un formato basado en vectores.

Este último proyecto fue extremadamente interesante de leer y es el que más pistas e indicaciones me dio acerca de cómo avanzar y de cómo abordar nuestra problemática. Se puede observar que en sí la problemática que sostiene el proyecto es muy parecida a la nuestra.

Cabe indicar que no han sido los únicos proyectos investigados, en total cuento más de diez proyectos que tratan la vectorización de imágenes de diversas formas. Es por este motivo que hemos seleccionado estos dos últimos como mejor representación de toda la investigación para empezar el proyecto.

Ha sido con este último pero, con lo aprendido de los otros, y con reuniones con profesionales del sector de la Ingeniería Informática con amplios conocimientos de Visión por Computador que ideamos el plan de trabajo para abordar los diferentes requisitos que debía sostener y solucionar nuestro proyecto.

3. Plan de trabajo

Una vez recopilados los requisitos y funcionalidades que esperamos del proyecto. E investigados también varios proyectos relacionados con la materia de lo que queremos realizar. Ideamos un plan de trabajo.

Hemos optado por dividir el proyecto en varias fases. De esta forma, priorizamos la modularización del proyecto en pos de la siguiente afirmación <<Divide y vencerás>>. Filosofía que hemos aprendido en el grado y que nos será muy útil para lo que queremos abordar.

La modularización del proyecto sostiene la idea de ir tratando poco a poco los diferentes problemas con los que nos encontramos generalmente en un sinóptico e irlos solucionando en pos de un correcto tratamiento de los mismos. Asimismo, tenemos en mente la idea futura de poder realizar diferentes mejoras en secciones ya anteriormente abordadas con el objetivo de sacarle aún más partido al sistema.

El plan de trabajo es el siguiente:

1. **Detección de textos**: Emplear un OCR (Optical Character Recognition) para realizar un primer barrido por todo el sinóptico. Obtener los diferentes textos que en él se encuentran, sus coordenadas y su tamaño. La idea es en base a esa área donde se ha encontrado el texto, sustituirlo por el color predominante de la zona, que asumimos será fondo, y pintar toda la zona de ese color, esta forma extraemos el texto de la imagen original. La idea detrás de esto es evitar que más adelante el texto pueda suponer una interferencia o actúe como ruido en los siguientes pasos. El texto extraído será colocado desde este paso, manteniendo su tamaño y coordenadas en un SVG que actuará como sinóptico final.
2. **Detección de elementos**: Emplear YOLO para realizar una primera detección de cada elemento distinto dentro de cada sinóptico, obteniendo también sus coordenadas. De esta forma obtenemos la clase del elemento, que será introducida por el ingeniero, y nos servirá para su posterior detección dentro de todos los sinópticos de una misma planta. La detección cogerá ese elemento y lo comparará mediante pattern recognition con ese mismo elemento en varias orientaciones. Así se podrá abordar todas las posibilidades de este elemento dentro de todos los sinópticos. Al obtener la clase del elemento, nos permite ingeniar un sistema de sustitución basado en templates ya vectorizados y con características modificables según las conveniencias del usuario. Con esto esperamos que, una vez detectado cada elemento distinto, ya se contará con su clase y en un futuro podrá ser fácilmente reemplazable en caso de requerirlo. Cabe recalcar que los elementos detectados y clasificados, serán añadidos usando su template al

sinóptico final antes mencionado en el punto 1. *[Esta idea está aún en desarrollo y su explicación será más precisa en el Informe de Progreso II]*

3. **Detección de conexiones**: Emplear un sistema de reconocimiento basado en píxeles desde cada elemento para trazar los caminos de las conexiones que conectan cada elemento con el siguiente. Cada elemento está conectado a otro (o no) siguiendo un camino que equivale a las líneas del proceso industrial. Detectaremos esto y los relacionaremos entre sí empleando algoritmos como Manhattan, por ejemplo, para unirlos buscando la distancia mínima entre ambos y evitando a su vez el cruce de caminos que no están pensados para cruzarse. *[Esta idea está aún en desarrollo y su explicación será más precisa en el Informe de Progreso II]*

Este es el proceso que esperamos seguir para la realización de este proyecto. Cabe destacar que el punto 2 y 3 se abordará de mejor forma en el siguiente Informe de Progreso. Estas son las ideas que tenemos actualmente, pero con el avance del proyecto pueden verse alteradas en pos de una mejor forma de abordar la situación.

4. Progreso

4.1. Punto 1 – Detección de textos

En este apartado describiré el proceso que seguí en la búsqueda y aplicación de un OCR para realizar el Punto 1 descrito en el plan de trabajo. Todo el proyecto está siendo realizado en Python.

Primeramente, hice una búsqueda preliminar de OCR's de código libre. Las características que tenía claras desde el principio de la búsqueda era que no se deberían enfrentar a mucho ruido, y que no era necesario que fuesen extremadamente potentes. El objetivo era realizar un barrido rápido de un diagrama en 2D en el que en el peor de los casos la imagen se vería un poco borroso, pero no lo suficiente como para que la detección debiese enfrentarse a problemas que requiriesen de dilataciones o contracciones o mucho preprocesamiento de la imagen. Además de esto, no se vería enfrentada a mucho ruido. Esto es porque por norma general en los sinópticos se intenta evitar colocar textos que intercedan con otros elementos del diagrama. Esto se hace por seguridad, para que no puedan surgir confusiones entre los operarios que visualizan estas pantallas en una pasada rápida y que pueda llevar a una malinterpretación en el nombre del elemento. Esto es un dato que, si bien no está como tal escrito en ningún sitio, es algo que he ido viendo y se me ha recalcado en este último año que he estado trabajando en el sector de la Industria4.0.

La búsqueda concluyó en dos principales librerías: EasyOCR, Pytesseract.

También se recomendaba que en caso de imágenes en las que hubiese una gran cantidad de textos, una OCR extensamente entrenada sería una gran aliada a tener en cuenta. Esto será importante más adelante.

En un principio, para lo que se iba a usar y dadas las condiciones anteriormente expresadas con las dos librerías encontradas sería suficiente.

Al inicio trabajamos con un pequeño diagrama hecho por nosotros mismos [fig4.1.a] en Draw.io (web para hacer diagramas en SVG) para hacer pequeñas pruebas del OCR y decidir sobre cual decantarnos.

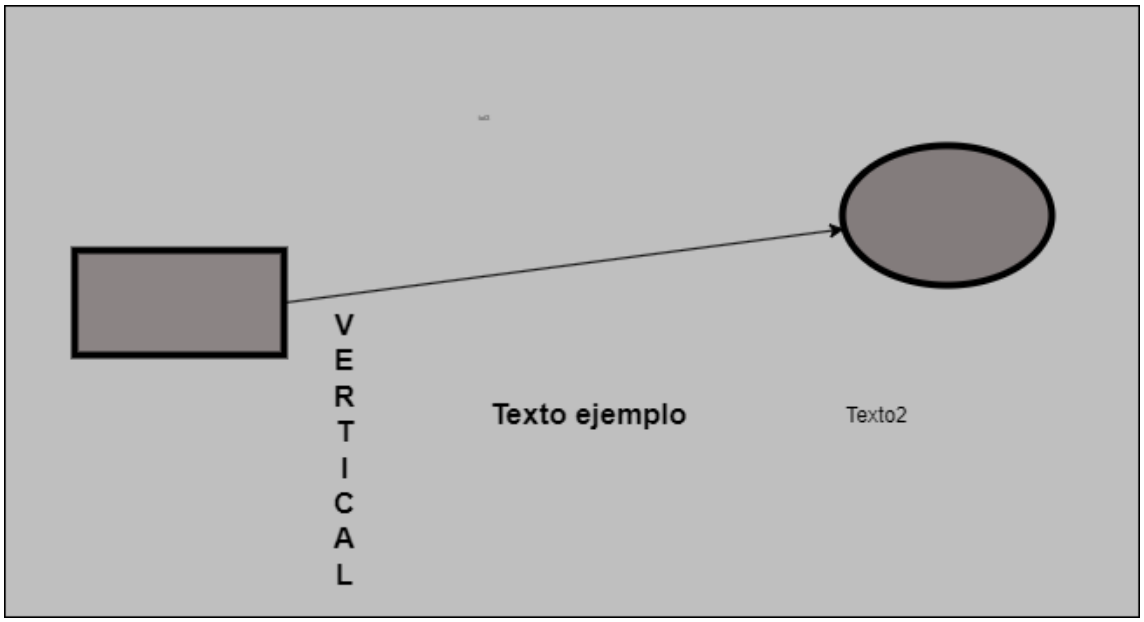

fig4.1.a – Diagrama de ejemplo

Este diagrama, si bien no era gran cosa, nos servía para probar un poco de cada cosa:

- Un texto muy pequeño (está situado sobre la flecha y parece un borrón)
- Un texto en vertical
- Un texto bien definido a un tamaño normal
- Un texto bien definido un poco más pequeño que contiene un número

Sobre esto hay que indicar varias cosas. Y es que no estábamos habituados a trabajar con OCR. Era la primera vez que trabajábamos con esta tecnología fuera de clase y no sabíamos exactamente qué esperar de ello, cómo nos vendrían devueltos los datos, o cual era su capacidad de detección real, sus fortalezas y flaquezas.

Creamos un pequeño script de test para comprobar qué librería debíamos escoger. En la siguiente imagen [fig4.1.b] podemos observar cómo actuó EasyOCR con el diagrama de ejemplo mencionado:

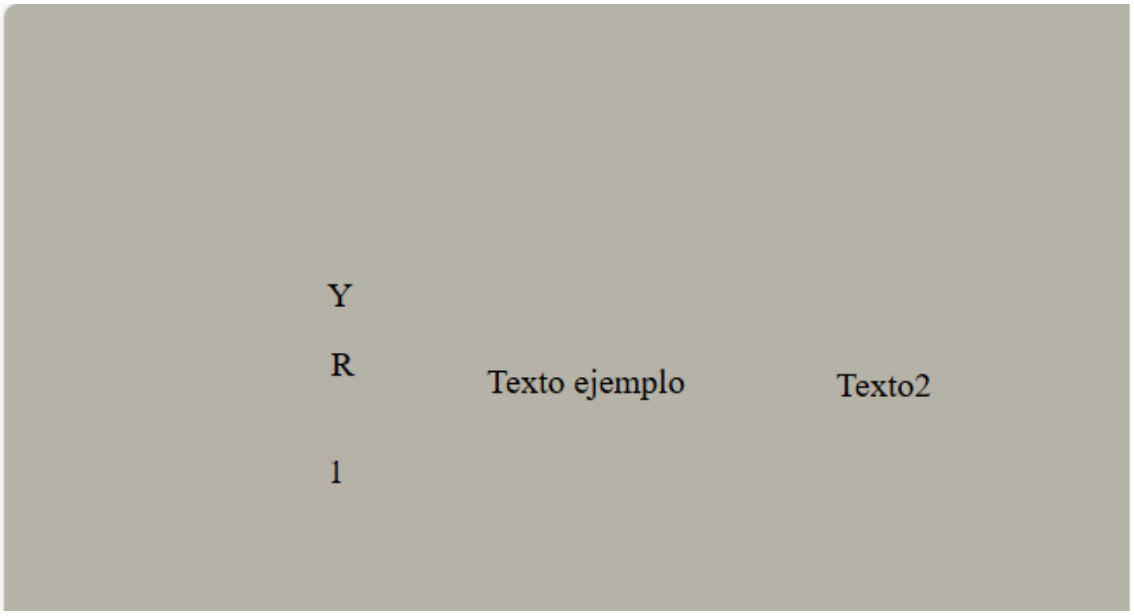

fig4.1.b – Resultado de EasyOCR

Se puede observar cómo:

- El texto muy pequeño ha desaparecido, no ha sido detectado.
- El texto vertical ha sido detectado erróneamente.
- El texto definido ha sido detectado perfecto.
- El texto con número ha sido detectado perfecto.

Ahora observemos el resultado de Pytesseract ante la misma imagen:

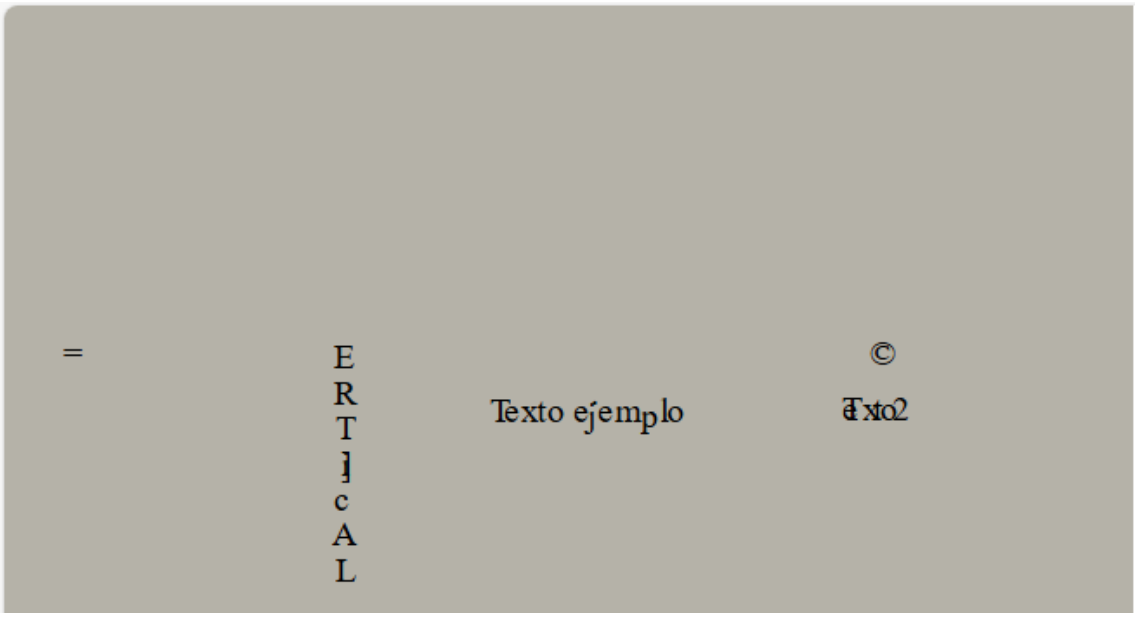

fig4.1.c – Resultado de Pytesseract

Se puede observar cómo:

- El texto muy pequeño ha desaparecido, no ha sido detectado.
- El texto vertical ha sido detectado casi perfecto, pero la V no ha sido detectada, la I ha tenido alguna clase de error y la C es minúscula.
- El texto bien definido ha sido detectado correctamente pero lo ha detectado de forma individual cada caracter.
- El texto con número ha pasado por lo mismo que el anterior.
- Existen pequeños caracteres sueltos por el lienzo, fallos de detección y detectados como caracteres inexistentes originalmente.

Se realizaron muchas pruebas cambiando diferentes parámetros de ambos modelos. Pero a la conclusión que se llegó es que EasyOCR era mejor para la detección de textos horizontales en casi todos los casos. Mientras que Pytesseract era mejor para la detección de los textos verticales, pero para los horizontales era un despropósito.

Lo bueno de ambos OCR y que parece que es algo estandarizado en diversos OCR que también fueron considerados (como Keras), solo que no tan exhaustivamente, es que te devolvían los datos con: el texto detectado, los cuatro puntos de coordenadas en los que se ha encontrado y una puntuación de confianza del resultado.

Esto es muy interesante puesto que, con el texto y los puntos de coordenadas, podemos emplear la librería Svgwrite para colocar directamente en un SVG con las dimensiones originales de la imagen el texto detectado.

La confianza es otro dato muy importante pero en este estado del proceso la estamos desestimando.

Además de esto, EasyOCR incluye un parámetro para indicar el idioma del texto que se quiere detectar. Esto es muy útil también.

Así pues, se avanzó en el proyecto habiendo escogido EasyOCR como modelo OCR con el que actuar. Pero esta vez debía probarse en un sinóptico de verdad y ver cómo respondía.

La imagen escogida para la prueba fue la siguiente [fig4.1.c] y los motivos fueron que tiene multitud de textos, todos de un tamaño parecido, y para los que tienen tamaños dispares no tienen diferencias muy exageradas. También hay textos que contienen números. Además de esto, la imagen es en color, con lo cual existen textos de diferentes colores y textos pintados sobre colores distintos.

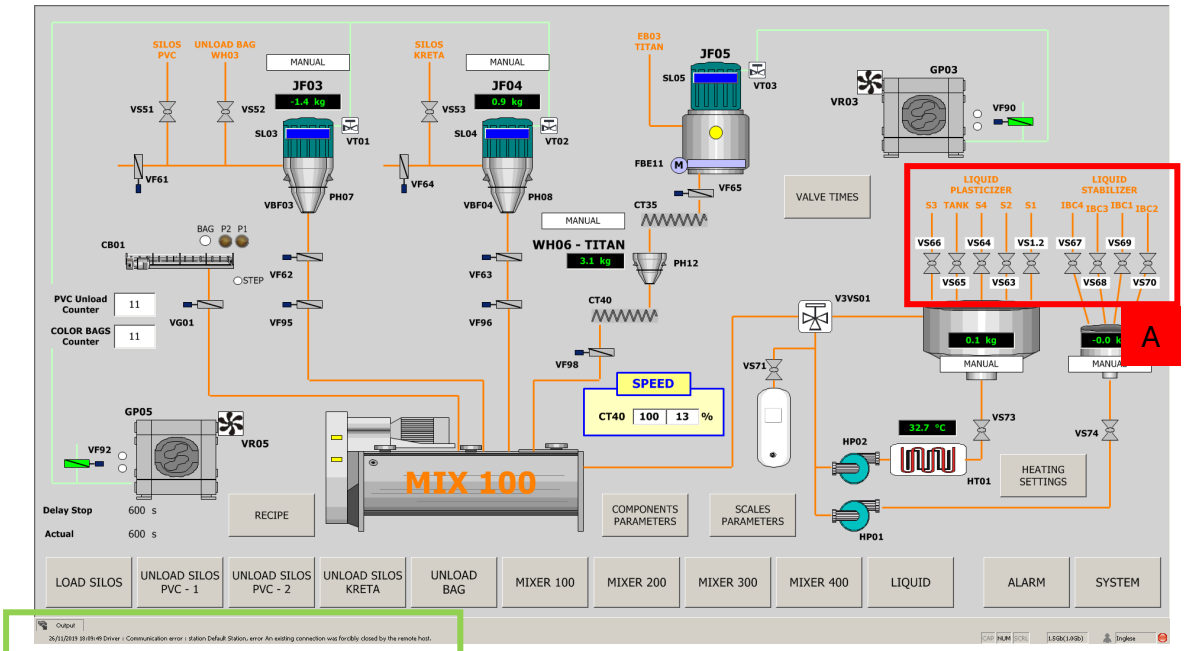

fig4.1.d – Imagen input

Los resultados fueron los siguientes [fig4.1.e]:

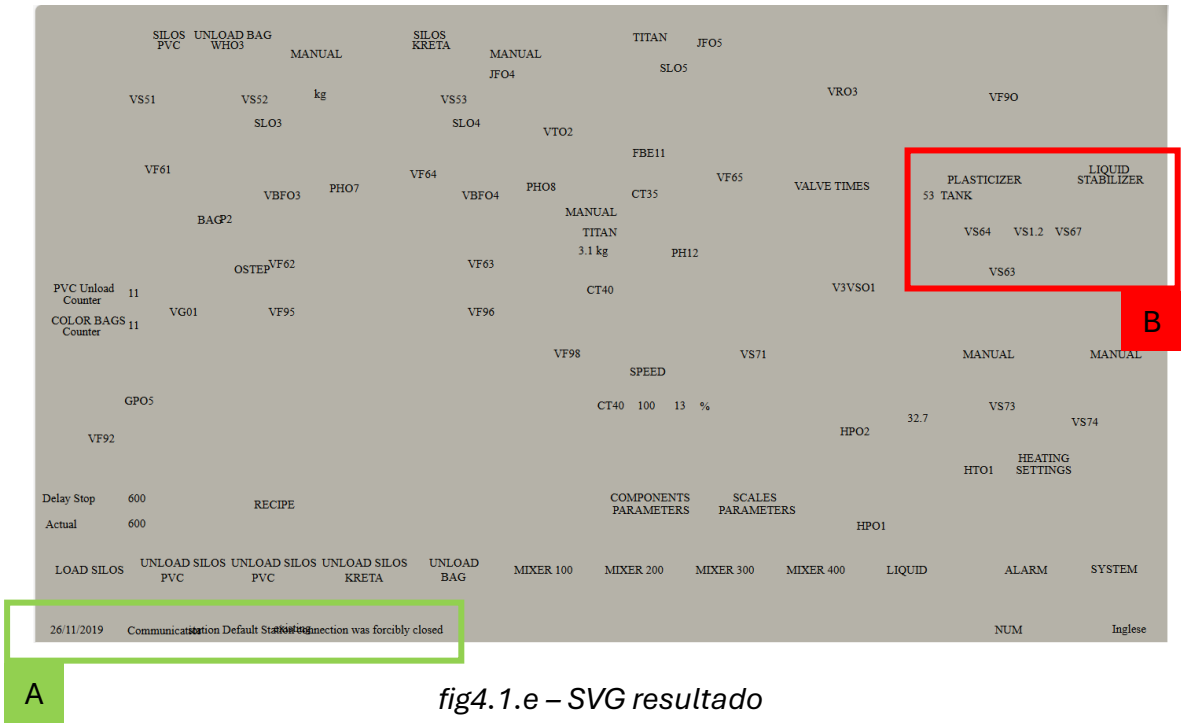

fig4.1.e – SVG resultado

Si bien a simple vista puede parecer un buen resultado, y lo es, existen partes que faltan como en la zona B marcada, y otras que quedan solapadas como por ejemplo la zona A.

EasyOCR estaba bien y cumplía las expectativas, pero necesitábamos más, por más pruebas que hicimos, por más preprocesamiento que instásemos realizar en la imagen, la detección no mejoraba. Intentamos binarizar la imagen según diferentes thresholds con el fin de resaltar más los textos y minimizar el impacto de las formas [fig4.1.f], así como pasar la imagen a escala de grises para observar los resultados [fig4.1.g].

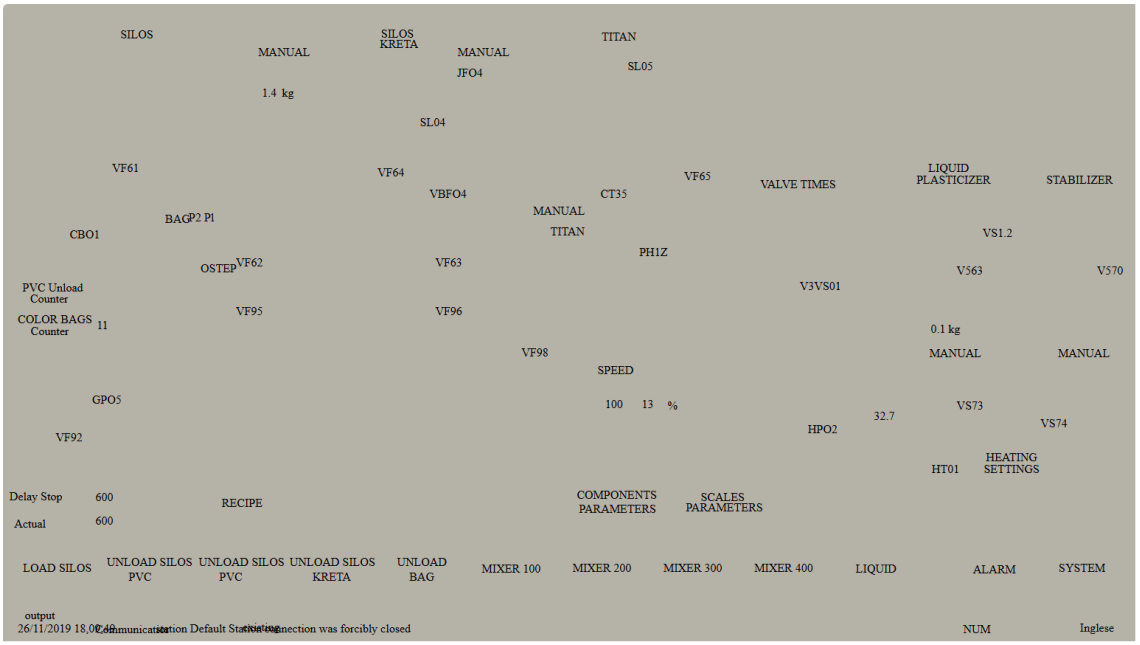

fig4.1.f – SVG resultado de la imagen binarizada

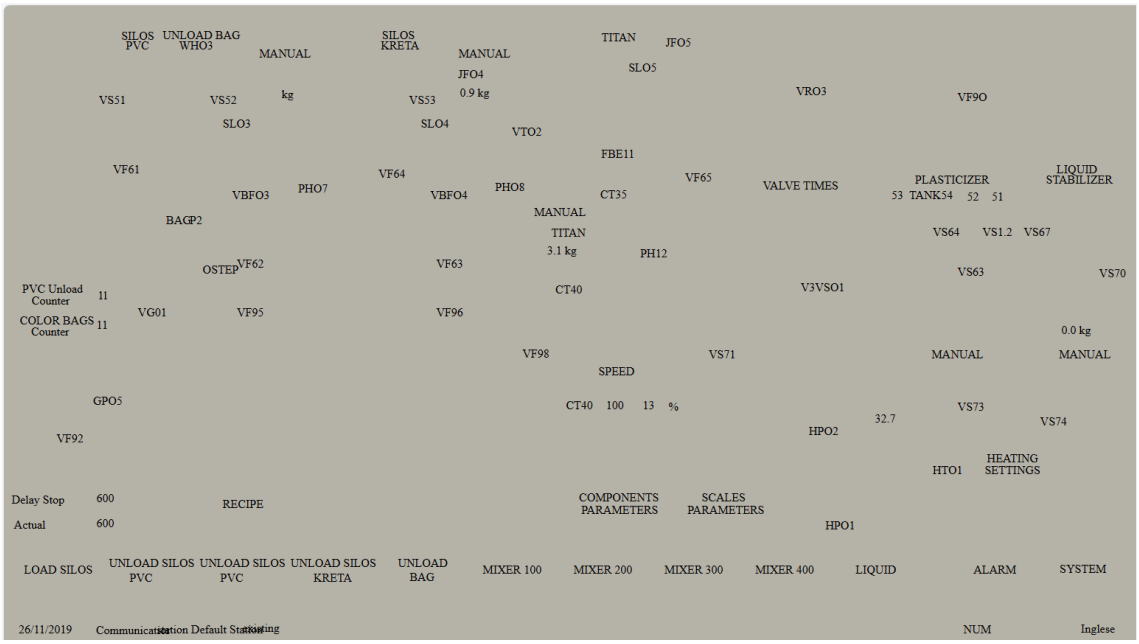

fig4.1.g – SVG resultado de la imagen en escala de grises

Como se puede observar, sí es cierto que existe cierta mejora respecto al original. Pero únicamente en el caso de la imagen en escala de grises. En el caso de la imagen binarizada el resultado es incluso peor al original pues se pierden gran cantidad de textos.

La conclusión a la que llegamos es que necesitábamos más, un OCR pre entrenado exclusivamente en la detección de textos en diagramas 2d.

Después de mucha investigación hallamos un OCR especializado en la lectura de archivos de estilo PDF y en libros escaneados. El nombre de este OCR es Surya-OCR.

Así que realizamos pruebas y analizamos los resultados [fig4.1.h].

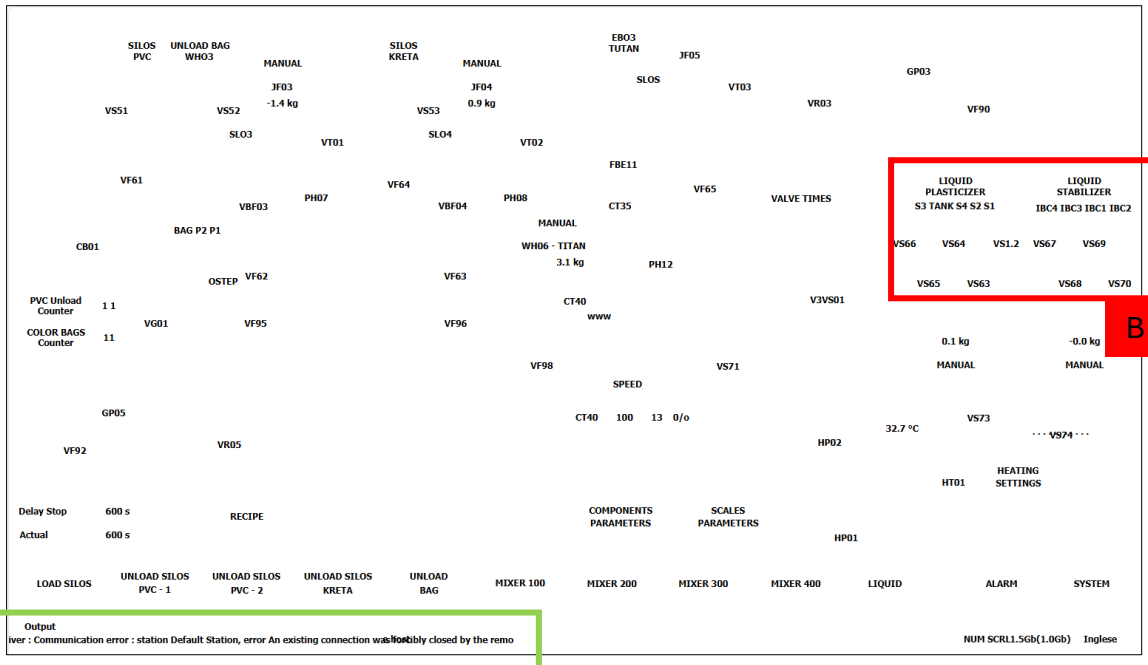

fig4.1.h – SVG resultado con Surya-OCR

Se puede observar que esas zonas en las que anteriormente no se detectaba nada, ahora sí lo hacían. Surya-OCR se adaptaba muchísimo mejor al estilo de diagrama que empleábamos y mejoraba por mucho la detección de los textos en la Zona B.

Se pueden ver aún algunos pequeños errores en la Zona A y en alguna otra. Pero la mejora era tan grande que tomamos la decisión de quedarnos con este último modelo.

A partir de aquí, hicimos un análisis de las confianzas de cada texto, marcando con un color azul la Bounding Box de aquellas que tuviesen una confianza superior al 0.8 y en amarillo aquellas que superasen el 0.72, que es donde detectamos después de varias pruebas que había ruido. Todo lo que no llegase a 0.72 fue

considerado rojo y, por lo tanto, texto no fiable. En la siguiente imagen se puede ver el resultado de este proceso [fig4.1.i].

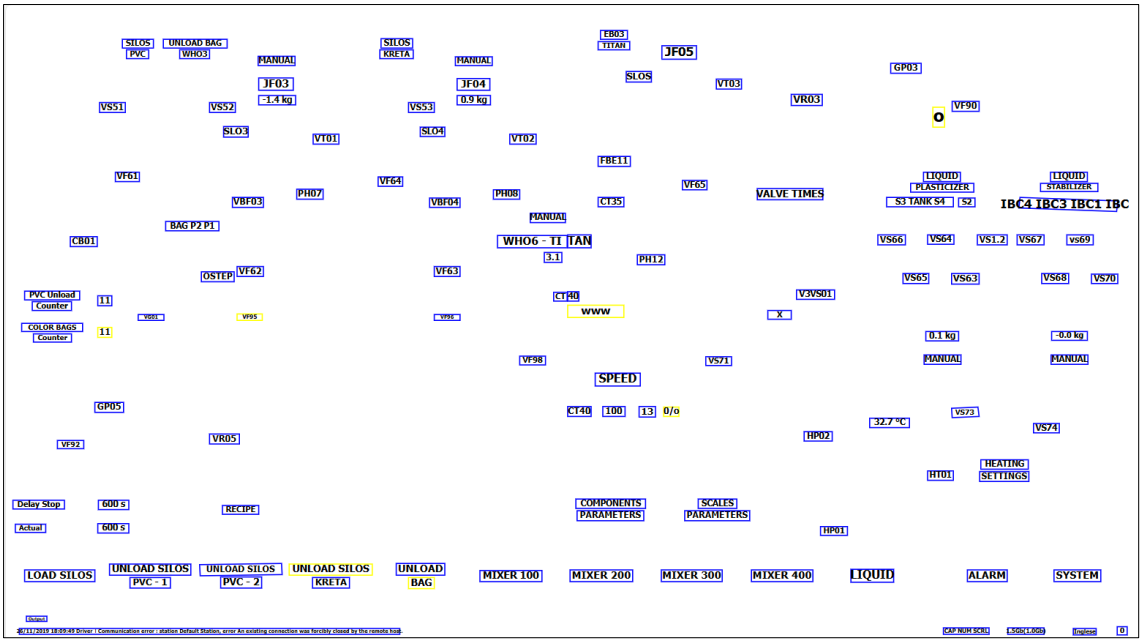

fig4.1.i – SVG resultado con Surya-OCR, Bounding Boxes resaltadas

A partir de aquí. Hemos observado también que a pesar de que el OCR sea el mismo, mejora su detección al ejecutarse en trozos de esta misma imagen más pequeños, a pesar de que el contenido sea el mismo.

Esto es una mejora que sigue en proceso y que será explicada de forma detallada en el siguiente informe de progreso.

-- Hasta aquí, de momento, el progreso. –

Actualmente estoy trabajando en esta mejora de detección del texto, pero con el tiempo que tengo, probablemente decida seguir con la planificación y no trabajar en ella por el momento. A espera de las fiestas de navidad o algún puente en el que tenga más tiempo para trabajar en aquellos puntos no prioritarios.

5. Planificación

La planificación se ha visto altamente alterada debido al cambio de modelo de trabajo y debido también a una mayor comprensión y acotamiento en los requisitos del proyecto.

Así pues, sin ánimo de ser tan exacta en cuanto a trabajo semanal se refiere respecto al informe inicial, la nueva planificación es la siguiente:

Octubre:

- Creación del Informe Inicial.
- Investigación y creación del modelo de trabajo.
- Búsqueda de OCR, implantación y creación de herramienta de exportación a SVG.

Noviembre:

- Semana 1:
    - Creación e implantación de mejoras de Surya-OCR.
    - Creación del Informe de Progreso I.
- Investigación acerca de YOLO.
- Implantación de YOLO en el proyecto.
- Implantación de posibles mejoras de YOLO.

Diciembre:

- Investigación para aplicar métodos de reconocimiento de caminos.
- Semana 2:
    - Aplicación de los métodos encontrados de reconocimiento de caminos.
    - Creación del Informe de Progreso II.
- Mejoras en el sistema de reconocimiento de caminos.

Enero:

- [En caso de dar tiempo] Aplicar todas las mejoras que se han ido apuntando durante el proceso de creación del proyecto.
- Encapsularlo todo en un mismo programa.
- Creación del dosier final.

Febrero:

- Semana 1:
    - Creación de la presentación del TFG.

6. Bibliografía

La bibliografía será añadida de la forma más profesional posible en el Informe de Progreso II.