Automatización de esquemas para la Industria 4.0 con Visión por Computador

Hernán Capilla Urbano

1 de febrero de 2025

Resumen– Este trabajo presenta una herramienta basada en Python para la automatización de la renovación de sinópticos en la Industria 4.0. A partir de una imagen inicial, el sistema utiliza técnicas de visión por computador, como OCR y Template Matching, para identificar textos, elementos industriales y conexiones entre los distintos elementos del sinóptico. El resultado es un diagrama escalable en formato SVG, con estructuras de datos que permiten la modificación y actualización eficiente. La herramienta reduce significativamente el tiempo de trabajo, manteniendo la fidelidad del diseño original, y facilitando la tarea de diseño del mismo.

Palabras clave– Visión por Computador, Automatización Industrial, Industria 4.0, Reconocimiento Óptico de Caracteres, Reconocimiento de caminos, SCADA, Generación de SVG, Renovación de diagramas sinópticos, Emparejamiento de plantillas.

Abstract– This work presents a Python-based tool for automating the renewal of synoptics in Industry 4.0. Starting from an initial image, the system employs computer vision techniques, such as OCR and Template Matching, to identify texts, industrial elements, and connections between the various components of the synoptic diagram. The result is a scalable SVG-format diagram with data structures that enable efficient modification and updating. The tool significantly reduces work time while maintaining the fidelity of the original design and assisting the user in streamlining the design process.

Keywords– Computer Vision, Industrial Automation, Industry 4.0, Optical Character Recognition, Path Recognition, SCADA, SVG Generation, Synoptic Diagram Renovation, Template Matching.

1 Introducción

La idea para este proyecto parte de una problemática que afecta tanto a ingenieros juniors como ingenieros seniors y que el propio autor se ha topado en su carrera profesional. Esta problemática es la de la renovación de sinópticos [1] en el ámbito industrial.

La renovación de un sinóptico puede consistir en la actualización a una versión posterior del programa original en la que fue diseñado o una migración a un nuevo entorno conservando todas las funcionalidades de las que disponía el programa original.

Y es que actualmente está habiendo una explosión de lo que llamamos industria 4.0 [2]. El mercado se está diversificando y, en consecuencia, vemos como donde antes únicamente había un par de programas para gestionar y monitorizar toda la información de las que nos provee un sinóptico; hoy en día hay decenas y decenas de programas, cada uno perteneciente a empresas diferentes, y con funcionalidades muy diversas entre sí. Es por este motivo que la renovación de los sinópticos se ha convertido en una tarea muy común en el mercado actual. Muchas empresas invierten en actualizar sus antiguos sinópticos y quieren conservar las antiguas funcionalidades que estos tenían.

Algunos de los principales problemas de esta tarea es que originalmente la mayoría de los programas que se usaban hace años suelen ser muy pesados. También suelen estar instalados en sistemas operativos antiguos que se siguen usando en las fábricas a día de hoy y que se mantienen sin actualizar. Ésto nos lleva a que, para obtener información del entorno en el que fueron diseñados, es necesario emular en máquinas virtuales las condiciones de esos programas.

Esto conlleva varias incomodidades y ralentí derivado del entorno desfasado sobre el que se ha de trabajar. Ejemplos de esto son el input lag propio de una máquina virtual o

• E-mail de contacto: 98.capilla.h@gmail.com
• Mención realizada: Computación
• Trabajo tutorizado por: Elitza Nikolaeva Maneva (Departamento de Ciencias de la Computación)
• Curso 2024/25

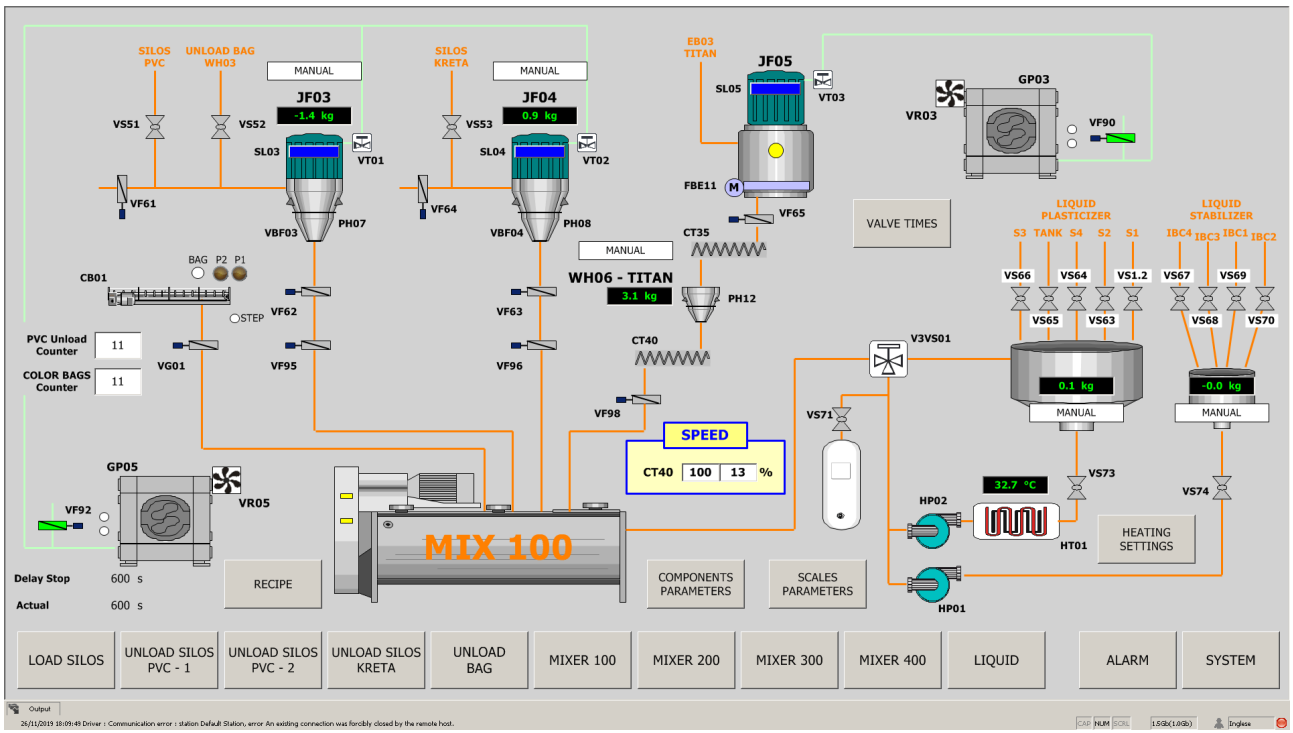

Fig. 1: Ejemplo de sinóptico en un proceso industrial.

el tener que levantar la imagen de esta máquina virtual, así como configurarla; además del hecho de que normalmente se tiene que transportar en formato físico por lo pesado del propio archivo, con lo cual implica una necesidad de transporte físico de este contenido.

Otro problema son las licencias que se usaron en aquel entonces, pues son caducas y están pensadas para un uso anual. La renovación de una pantalla no suele durar tanto tiempo como para que salga a cuenta adquirir una licencia anual o mensual para una versión específica de un entorno que hoy en día está desfasado.

2 Objetivos

El principal objetivo consiste en emplear los conocimientos adquiridos en visión de computador, análisis de algoritmos, heurísticas y lógica; para conseguir aligerar toda esta pesada carga y ayudar al usuario encargado de hacerla a aligerar el proceso de renovación de sinópticos.

3 Estado del Arte

Entre todos los artículos y trabajos revisados para la realización de este proyecto, cabe destacar dos en particular, ya que representan las aproximaciones más cercanas a la idea conceptualizada para esta propuesta, aunque presentan ciertas diferencias [3, 4].

3.1. LIVE: Towards Layer-wise Image Vectorization

[5] LIVE se enfoca en partir de imágenes “raster” para ir creando mediante iteraciones y capa por capa una imagen semejante en formato SVG.

Este proyecto encajaba de manera óptima con el objetivo de convertir una imagen de tipo “raster” a formato SVG. No obstante, fue en este contexto ha sido identificada una diferencia clave entre dicho proyecto y el planteamiento personal, la cual resulta de especial relevancia para el desarrollo de la propuesta propia.

El objetivo no se limita a obtener simplemente una imagen en formato SVG, sino que radica en extraer los datos subyacentes, etiquetar los elementos y definir sus posiciones dentro del diagrama y, de esta manera, dotar al proyecto de un carácter modular, permitiendo su modificación y adaptación según las necesidades del usuario. Figura 2

3.2. Raster-to-Vector: Revisiting Floorplan

[6] Raster-to-Vector se enfoca en crear planos tridimensionales a partir de planos bidimensionales. Esto lo realiza a través de buscar los puntos clave de unión entre las diferentes estancias de una casa. A partir de estos y del texto que detecta en el plano diferencia habitaciones, le asigna a cada una un uso doméstico. Figura 3, Figura 4

Este proyecto resulta fundamental en la definición del enfoque y la toma de decisiones para el desarrollo de la propuesta propia, ya que incorpora diversas técnicas previamente consideradas, como el uso de OCR (Reconocimiento Óptico de Caracteres), la detección de puntos clave y el enfoque en la identificación de estancias. No se limita únicamente a la transformación en 3D, sino que otorga especial importancia a la relación entre los puntos clave y a la heurística aplicada para su detección.

4 Metodología

En primer lugar, se realiza una detección de textos. Primeramente, todos los textos del sinóptico son detectados y

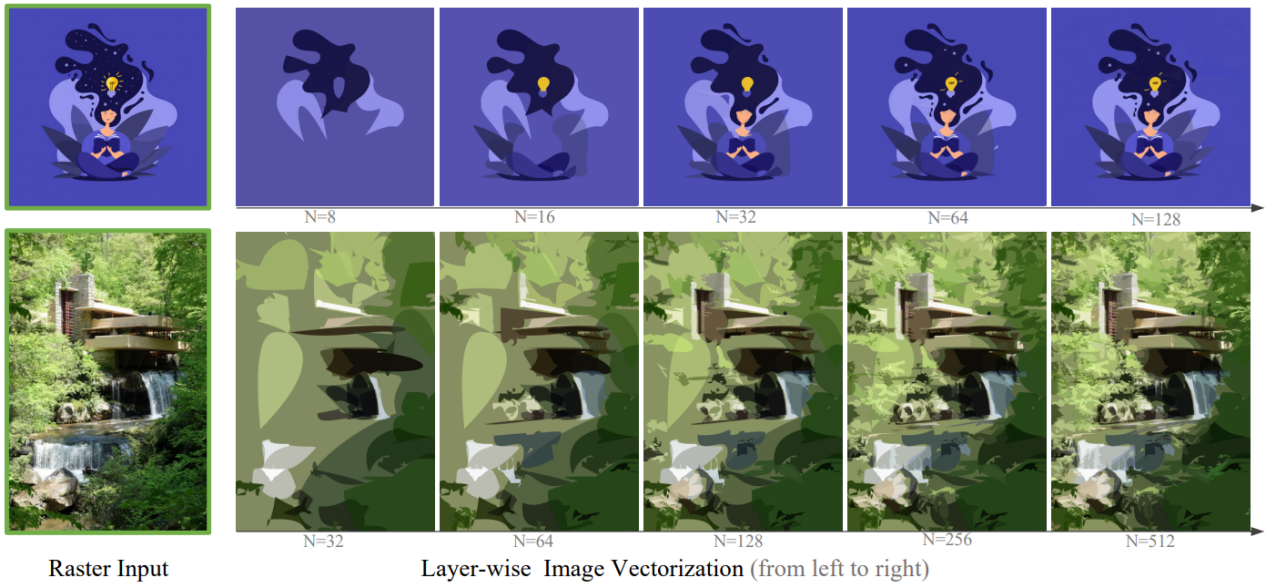

Fig. 2: Proceso de vectorizado por capas en LIVE.

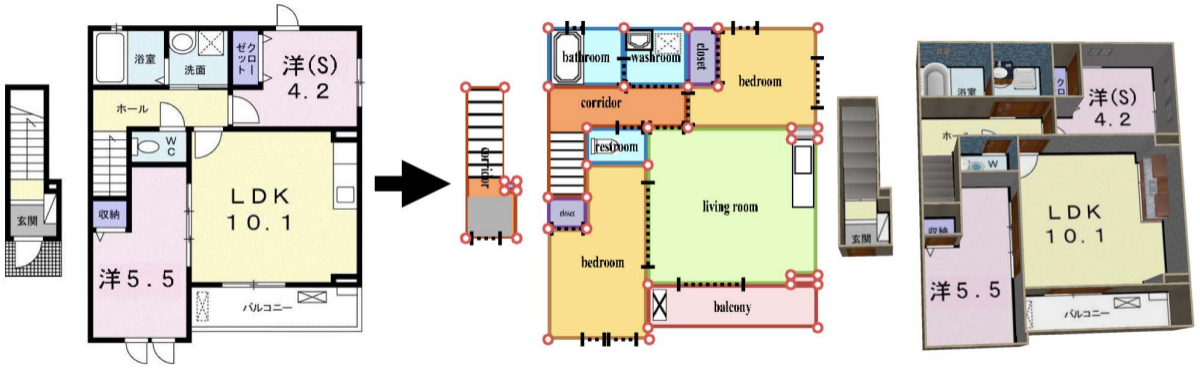

Fig. 3: De imagen a plano vectorizado y modelo 3D.

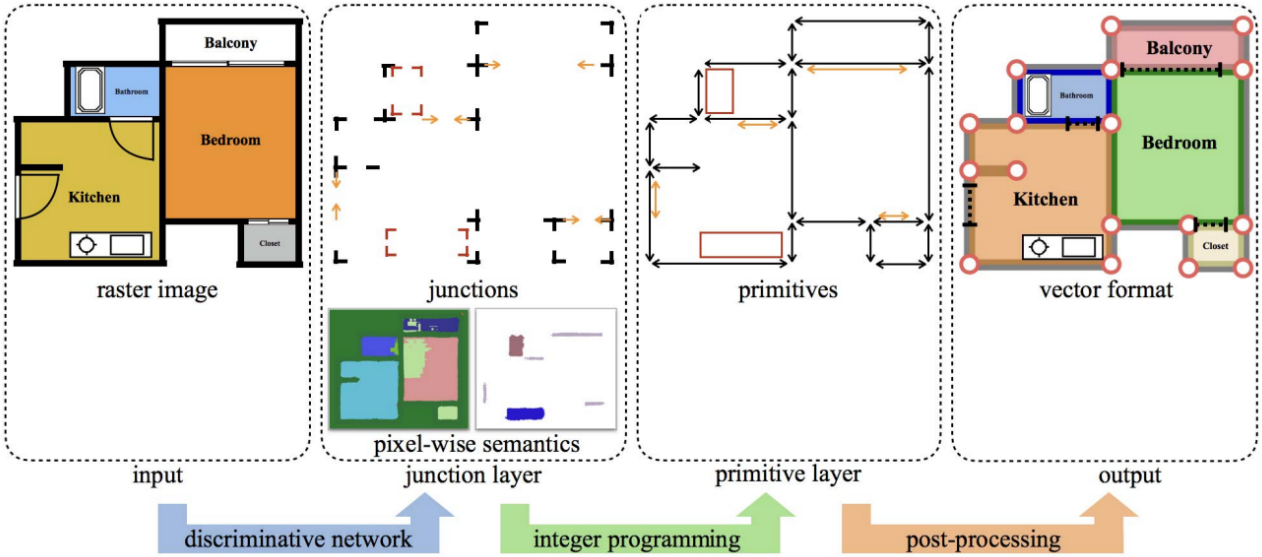

Fig. 4: Proceso de detección de puntos clave. Formación de habitaciones en base sus textos y terminales.

posteriormente eliminados de la imagen original, siguiendo la filosofía de que no interfiriesen en el siguiente punto del proceso. La gran mayoría de estos textos son descriptivos de elementos pertenecientes a la cadena de producción o incluidos en botones o tablas, con lo cual no suelen sobreponerse a objetos que forman parte del diagrama. De esta forma será más sencilla su comprensión y su visualización a la hora de verlos y fijarse en ellos.

En segundo lugar, realizamos una detección de maquinaria. Maquinaria son todos aquellos elementos de origen industrial que realizan un trabajo dentro de la línea de producción. Ejemplo de esto pueden ser bombas, válvulas y motores, es decir, todo aquello que de alguna forma conecta o ayuda a procesar diferentes sustancias o fluidos dentro del proceso industrial descrito en la pantalla. Una vez realizada la detección de los elementos, son extraídos tal y como se ha hecho en el punto anterior para dejar paso al último punto.

En tercer lugar, sería realizada una detección de caminos entre elementos. Habiendo extraído previamente tanto textos como maquinaria, únicamente quedaría en el diagrama los diferentes caminos que unen las diferentes máquinas, mostrando el flujo que sigue el proceso industrial. Se detectarían estos caminos y se crearían uniones entre los varios elementos del diagrama.

A modo de resumen, cabe resaltar que la idea detrás de todos estos pasos es la siguiente: evitar toda esa farragosa adquisición de elementos originales en su entorno natural y replicarse en un nuevo y renovado entorno. Se pretende partir de la base de una sola imagen de entrada.

La hipótesis es que si es posible desgranar progresivamente y, utilizando cada uno de los puntos descritos previamente, convertir cada una de las partes detectadas en una imagen en un elemento modificable que pueda ajustarse, actualizarse y moldearse según las necesidades del usuario. Por este motivo, se ha decidido plantear como output de este proyecto la transformación del sinóptico original a un formato escalable y vectorial, como es SVG. Este formato, además de ser compatible con todos los entornos de diseño de pantallas de origen industrial, permite conservar las estructuras de datos extraídas en cada etapa del proyecto.

4.1. Detección de textos

Para este primer punto, se decidió investigar en diversos sistemas de OCR (Optical Character Recognition) [7] con el objetivo de extraer los textos. Los tres investigados han sido EasyOCR, Pytesseract y SuryaOCR. Estas opciones consideradas se limitaron a herramientas de código libre y de potencia moderada, dado que el proceso se planteó como relativamente ligero, sin la necesidad de enfrentarse a un nivel significativo de ruido en la imagen, lo cual elimina la exigencia de una capacidad de procesamiento elevada.

Debido a su simplicidad y ligereza, se optó por utilizar EasyOCR [8], pero los resultados obtenidos no cumplieron con las expectativas. Esta herramienta presentaba dificultades para detectar textos en orientación vertical, así como textos de tamaño reducido. Aunque los textos en vertical no son frecuentes, los de pequeño tamaño sí lo son, lo que representaba un inconveniente significativo.

Alternativamente, se probó con Pytesseract [9] [10], cuyos resultados fueron más satisfactorios, aunque con una limitación notable: fragmentaba las cadenas de texto, identificándolas como caracteres individuales. Esto afectaba directamente al objetivo de preservar la estructura completa de las palabras y, por ende, sus coordenadas dentro de la imagen.

Tras una investigación más detallada sobre el uso de estas herramientas y su aplicación en otros contextos, se llegó a la siguiente conclusión: se estaba utilizando una herramienta cuyo propósito principal era la detección de textos en superficies reales, es decir, en entornos del mundo físico. Estas herramientas están diseñadas para identificar textos sobredimensionados, en movimiento, en perspectiva, entre otros, típicos de imágenes capturadas en la vida real. Sin embargo, este enfoque no se ajustaba a las necesidades del presente estudio, ya que no se requería dicha funcionalidad.

Lo que se requería realmente era un OCR que estuviese diseñado para trabajar sobre textos en superficies planas. Es decir, un entorno bidimensional donde el texto no estuviera afectado por perspectivas distorsionadas ni por fondos ruidosos. En los sinópticos los textos están en la gran mayoría de ocasiones sobre un fondo sin variación de tono o color, una capa de píxeles que raramente varía entre sí, al realizar el trabajo de fondo de diagrama.

El OCR que cumplió con los requisitos fue SuryaOCR [11]. Este es un OCR especializado en la detección de textos enfocado para páginas de libros, documentación e impresiones. Está concebido para facilitar o automatizar la detección de textos en medios analógicos, con el fin de extraer su información y traspasarla a un medio digital.

Otra parte importante es que al estar especializado en estas tareas, se puede configurar el lenguaje en el que se quiere detectar, permitiendo una mayor precisión de las detecciones. Además, la forma en la que devuelve la información es en formato de “Bounding Boxes” acompañada del texto detectado y un valor de confianza. Esto es ideal por varios motivos. El primero es que se puede asignar un treshold para eliminar detecciones imprecisas. Además, más adelante, se puede extraer el espacio que ocupaba cada texto dentro de la imagen original y se guir conservando la información de las palabras que en él había. Una vez detectados y con el objetivo de eliminar falsos positivos, se consideran las siguientes situaciones como condiciones que invalidan un texto como positivo, provocando que no formen parte de la lista final de textos detectados.

Estas son:

1. El texto no puede tener únicamente un carácter. La existencia de pilotos o luces representados en el diagrama puede llevar a confusión al OCR y que piense que está detectando la letra “O” o una “C”; debido a su semejanza con la anterior letra.

2. El texto no puede tener únicamente dos caracteres. La existencia de válvulas en forma de mariposa o de los mismos pilotos anteriormente descritos puede llevar a confusión. El OCR puede de tectar una válvula como una “XI” o un piloto como la combinación de “C)”.

3. El texto no será válido si contiene un “(” pero no su contraparte “)”. Lo mismo pasará con “[” y “]”. Habitualmente no suelen encontrarse este tipo de elementos dentro de los textos, pero si salen suelen ser emparejados. De lo contrario se trata de un error en la detección del texto proveniente de los ángulos de las letras sobre el plano.

A continuación de la detección de textos, se procede a identificar el color del fondo de la imagen. Se considera como fondo el color más predominante entre los píxeles de toda la imagen, asumiendo que será aquel que se repita con mayor frecuencia. Dado que los sinópticos están diseñados

para no estar saturados de elementos, con el objetivo de facilitar la observación del diagrama, en la mayoría de los casos el color más repetido por píxel corresponde al fondo.

Una vez identificado el color de fondo, se seleccionan las "Bounding Boxes" correspondientes y se rellenan en la imagen original con el color del fondo. En caso de que estas se encuentren dentro de un elemento, se pintan de la misma manera. Esto se hace con el propósito de evitar posibles errores en etapas posteriores, específicamente en la detección de terminales o ángulos provenientes de los textos dentro de los elementos industriales, lo que podría comprometer la precisión de las detecciones.

Finalmente, el resultado de esta fase del proceso consiste en la generación de una imagen en formato PNG. En dicha imagen, se extraen todos los textos detectados, junto con su posición específica dentro de la misma. Asimismo, se rellenan con el color del fondo las denominadas "Bounding Boxes", que representan los espacios ocupados por dichos textos.

4.2. Detección de elementos

En este segundo punto se busca la detección de todos aquellos elementos que pertenecen a la maquinaria industrial.

Para enfocar este punto hay que tener en cuenta dos cosas. La primera es que, de forma habitual, la mayoría de los elementos empleados en los sinópticos para representar maquinaria industrial proceden de librerías ya existentes. Estas librerías son creadas por grandes empresas, productoras de dispositivos industriales, y contienen todo tipo de templates de elementos. Cada elemento tiene una clase, una representación gráfica y unas características propias. Un ejemplo de esto es la librería Symbol Factory de Siemens [12]. Es sencillo acceder a los gráficos de estos puesto que solo se requiere de una licencia para usar dichas librerías.

Por otro lado, hay que tener en cuenta que aquellos elementos que no están englobados en estas librerías, es porque los ha diseñado la compañía responsable de la creación del sinóptico para un uso más específico de su cliente. Estos templates gráficos son considerablemente más difíciles de obtener.

Hay que indicar, además, que para secciones distintas dentro de una misma planta industrial, la representación gráfica de los elementos no suele variar. Es decir, una bomba de una clase A representada en la nave norte de una empresa, suele ser representada exactamente igual que otra bomba de la misma clase A situada en la nave sur de la misma.

Bajo este pretexto empleamos Template Matching [13]. Ya que todos los elementos dentro de una misma factoría tienden a ser muy parecidos entre sí.

La idea es que, inicialmente, sea el propio usuario el que vaya seleccionando un elemento de cada clase sobre la imagen original. Una vez seleccionado uno de cada, el programa va preguntando por la clase de cada uno, y, al introducirlas, crea una librería para esa factoría. Para la que para cada clase seleccionada realiza el siguiente proceso de data augmentation: Recorta el área de la imagen seleccionada por el usuario para esa clase en concreto. A continuación, va aumentando lentamente cada una de sus dos dimensiones hasta llegar a doblar su tamaño, y hace lo propio con la inversa, es decir, la va empequeñeciendo hasta llegar a la mitad de su tamaño.

Una vez hecho esto, aplica Template Matching para cada una de las imágenes de la clase creadas con la imagen original, en busca de coincidencias. Esto lo realizará hasta cuatro veces para cada imagen, buscando el patrón de la imagen cada 90º.

Si ya existía previamente una librería, antes de pedirle al usuario realizar la selección de elementos, busca coincidencias usando Template Matching. En caso de encontrar las, le muestra al usuario por pantalla todas aquellas clases detectadas. Después de esto el proceso sigue con normalidad.

Para no tener inconsistencias con las detecciones automáticas, hay una serie de heurísticas y técnicas que empleamos para eliminar la mayoría de falsos positivos.

El primero es el uso de Intersection over Union [14]. Calcula la proporción entre el área de superposición de dos "Bounding Boxes", o maquinaria industrial en nuestro caso, para saber si una contiene una intersección o es un subconjunto de la otra. En caso de haberla, tomamos la decisión de eliminar de la lista aquella "Bounding Box"que más píxeles comparte con el color de fondo. Pues se considera que la que menos píxeles comparte es la que más contenido tiene y por lo tanto la que más probable será de ser la detección correcta. Este proceso es aplicado a aquellas áreas detectadas pertenecientes a la misma clase, porque al compartir similitudes entre sí, a veces Template Matching da falsos positivos únicamente en algunas secciones de un elemento.

Para aquellas áreas pertenecientes a diferentes clases, pero con intersecciones o subconjuntos entre sí, es aplicada otra heurística. Para estos casos se comprueba si una es un subconjunto de otra, es decir, si su área está contenida entera mente dentro de otra área. Si es el caso, aquella que es un subconjunto de otra, queda eliminada. Además de esto, una vez ha hecho todas esas comprobaciones entre clases, aplica otra heurística. Esta no distingue entre clases. Si un área tiene más de un 80 % de sus píxeles del mismo color que el fondo, se considera un falso positivo y que por lo tanto debe ser eliminada.

Finalmente, acabado todo este proceso de detección de elementos y con las "Bounding Boxes" detectadas, se procede a pintar con el color del fondo todos aquellos elementos que han sido detectados. El resultado es, tanto una imagen sin textos ni maquinaria; como una lista con todos los elementos detectados, la clase a la que pertenecen y su posición en la imagen.

4.3. Detección de caminos

En este punto la imagen no tiene textos y no tiene elementos, con lo cual, lo único que debería de quedar en esta imagen son los propios caminos que unen todos estos elementos del diagrama.

Estos caminos son aquellos que comunican áreas entre sí o que comunican la maquinaria que forma parte del proceso industrial mismo. Más allá de una indicación al usuario o al operario de qué zonas están conectadas no tienen una funcionalidad como tal.

Inicialmente se crea un grid para aligerar las siguientes fases. Cada posición del grid corresponderá a un píxel de la imagen resultante del Punto 2. Para cada una de estas posiciones en el grid, se colocan una por una todas las áreas

en los píxeles que representan. El resultado de esto es un grid donde cada píxel es 0 o el ID de aquel elemento que se ha detectado en ese punto.

Es entonces donde a partir de la lista de elementos detectados, se construyen uno por uno un área de un par de píxeles alrededor de su "Bounding Box", con la premisa de que, en caso de existir un camino que salga de este elemento, contendrá un color de píxel distinto al del color del fondo. En el caso de encontrarlo, se asigna a una lista de visitados cada píxel que esté a una distancia de 1 con él en cualquiera de las direcciones.

Este proceso acabará cuando ya no haya más píxeles por visitar en su cercanía. Al llegar a esta situación, se busca en el grid para su posición actual si existe en su cercanía el área de algún elemento detectado en el Punto 2. Si es así, se considerará que existe un camino que conecta el elemento desde donde parte, hasta el elemento que ha detectado cuando no podía continuar, creando así una conexión entre el elemento origen hasta el elemento final y guardando la misma, así como el path de unión entre estos dos.

En caso de no ser así, y haberse acabado el camino, es considerado y guardado en la lista como un punto terminal. Esto no tiene nada de malo. En los diagramas industriales hay puntos terminales que no llevan a otro camino como tal y que simplemente están ahí para indicar que el flujo continúa en otra sección. Se guardará como un terminal con ID propio, y se guardan también sus coordenadas.

Para las bifurcaciones de caminos, el programa lo tiene en cuenta y una vez ha acabado uno, sigue por el otro. En caso de encontrar dos conexiones, una para cada bifurcación, guarda cada una como distintas en entradas separadas.

Se va repitiendo este proceso para este elemento hasta que no encuentre más caminos no visitados que salgan de él. En ese caso, pasará al siguiente elemento.

Hay que tener en cuenta que para este punto únicamente deberían quedar caminos, con lo cual si aún existen elementos que el usuario o el programa no ha detectado, son marcados y formarán parte del path de un punto a otro si están conectados. Se pueden aplicar mejoras al respecto, que se detallan en la sección 6.

Para terminar, se devuelve un listado de todas las conexiones entre elementos, así como los caminos que conectan cada uno de ellos entre sí. Aquí ya no se devuelve ninguna imagen, puesto que el proceso entero ha finalizado.

Aun así, queda el último paso de todos.

4.4. Creación del diagrama final

El último apartado del proceso consiste en recopilar todos los datos obtenidos de cada uno de los puntos y transformarlos al formato SVG.

Recapitulando, a partir de una sola imagen, hemos obtenido lo siguiente:

1. Un color de fondo: considerado como el color predominante en la imagen.
2. Punto 1: hemos obtenido los textos y su posición.
3. Punto 2: hemos obtenido la clase de cada elemento de maquinaria, y su posición.
4. Punto 3: hemos obtenido los caminos y conexiones que conectan todos los elementos entre sí.

Con toda esta información podemos confeccionar la imagen en el formato SVG. Cada uno de los puntos mencionados será una capa en el SVG final.

En la primera capa, siendo la más baja, encontramos el fondo. Formado por píxeles del color más predominante.

En la segunda capa encontramos los textos detectados. Insertados según el tamaño en el que han sido detectados. Pero adaptándolos a un treshold por etapas según el tamaño original en píxeles que tenían sus "Bounding Boxes".

En la tercera capa, situada bajo la capa de texto, tenemos la maquinaria detectada. Para cada clase detectada, el programa irá a buscar su template en formato SVG dentro de la librería. Una vez encontrado, lo adjuntará a la capa 3 dentro del SVG resultante. Este será redimensionado y rotado para mantener la máxima fidelidad al diagrama original.

En la cuarta capa, situada entre las capas de texto y maquinaria, se pintarán todos los píxeles que se encuentren en los paths de las conexiones.

Toda la imagen, así como cada una de las capas es compatible con todos los editores que se emplean en el mercado común de diagramas industriales.

Obteniendo como resultado en una imagen perfecta mente modificable, modular y moldeable según las preferencias del usuario. Además de en las estructuras de datos que contienen la información de todos y cada uno de los elementos del diagrama.

5 RESULTADOS

En cuanto al resultado final podemos destacar varios aspectos. Vamos a ir paso por paso.

Primeramente, en la sección del OCR podemos observar en la Figura 5 respecto a la imagen original mostrada en la Figura 6 que hay algunos textos que no han sido detectados como deberían. Esto queda mejor ejemplificado en las imágenes resultantes del proceso del OCR.

Se puede ver como hay algunos textos que no han sido identificados. Esto se debe a que o no han superado el threshold, o las heurísticas.

En cuanto a la detección de elementos observamos en la Figura 7 que efectivamente los elementos han sido detectados. Así como sus duplicidades dentro del diagrama. Todos aquellos cuyo template se encontraba dentro de las librerías han sido añadidos a la imagen. Aquellos cuyo template no estaba, han sido resaltados con una "Bounding box". Y aquellos resaltados en naranja han sido obviados por el autor a propósito con el fin de mostrar como sería un resultado si estos no se detectaran.

Cabe destacar que es observable cómo los elementos que han sido insertados desde la librería en formato SVG se adaptan a las "Bounding Box" de las detecciones. Siendo reescalados para encajar a la perfección dentro del diagrama según las dimensiones definidas por las detecciones.

En la detección de caminos, se encuentra uno de los aspectos más interesantes del proceso. Este método mostrado en la Figura 8 ilustra de manera clara y resaltado en naranja cómo se realiza la identificación de los caminos, evidenciando que no existe distinción entre un elemento de maquinaria y un camino propiamente dicho. El algoritmo se limita a buscar píxeles que contrasten con el fondo, conectando una "Bounding Box" con otra. Además, cabe destacar que ciertos elementos fueron excluidos intencionalmente en

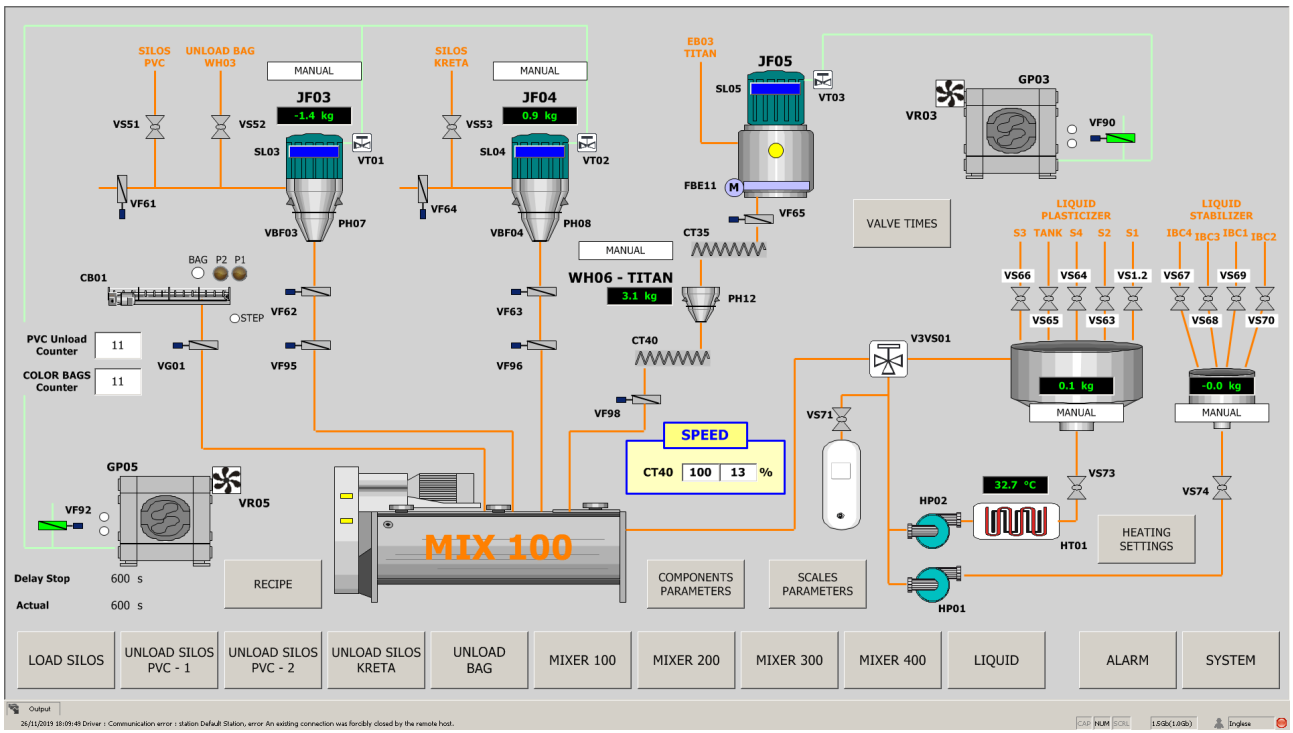

Fig. 5: Imagen del sinóptico original. Esta es la imagen que se toma como input.

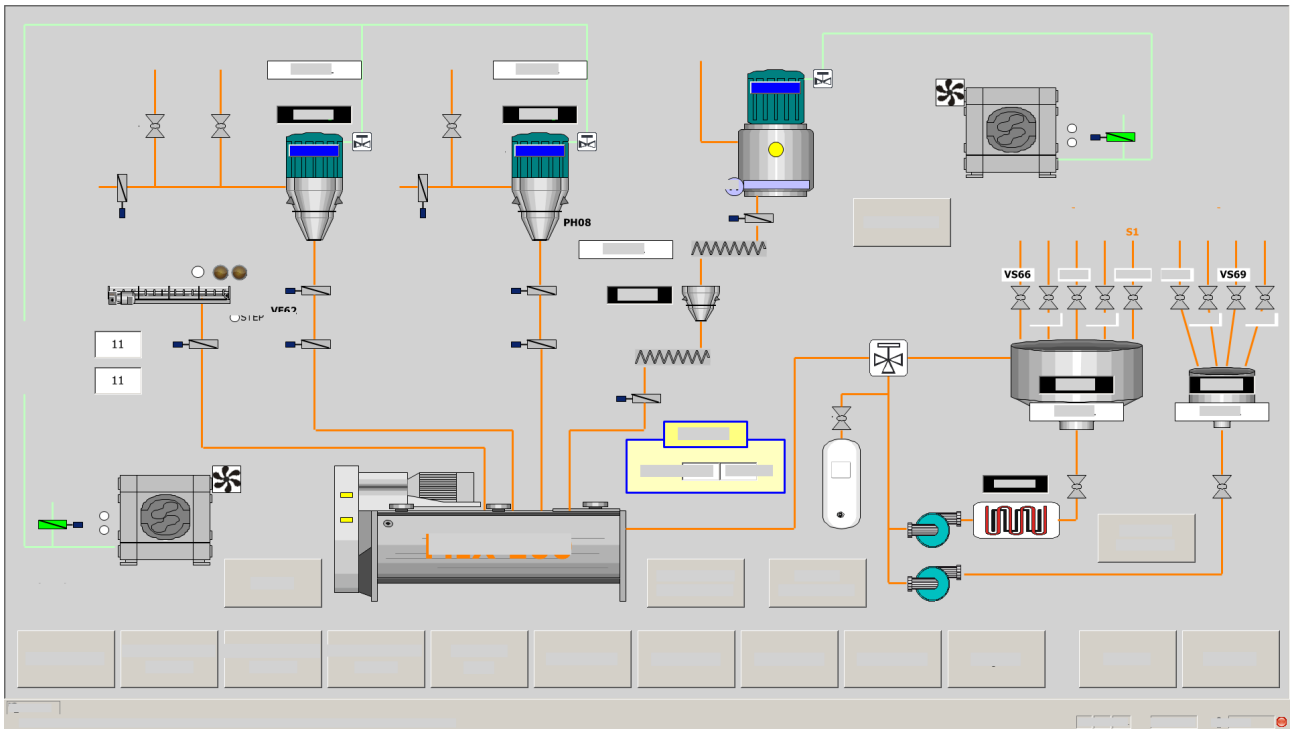

Fig. 6: Imagen después del procesado del OCR. Se puede observar como existen textos que no han sido detectados. También se muestra que en las zonas en las que antes había texto, ahora se han pintado con el color de fondo.

la fase dos, con el propósito de observar su representación en esta etapa.

Asimismo, existen otros elementos, como ciertas tablas o botones presentes originalmente en la imagen, que han sido identificados como posibles caminos. Esto se debe a que, en este punto, el algoritmo ha detectado un elemento que no cumplía con la heurística mencionada en la fase dos, ya que su "Bounding Box" no contenía un 80 % o más de píxeles del mismo color que el fondo. Por este motivo, dichos elementos han quedado resaltados como posibles caminos, ocupando el espacio correspondiente a un botón.

El último punto a tener en cuenta es que no nos hemos enfocado en otros elementos como tablas o botones que se pudieran encontrar en el diagrama original. Esto no estaba planteado para abarcarse desde el principio y queda reflejado en el diagrama final. Es una posible mejora que se podría añadir.

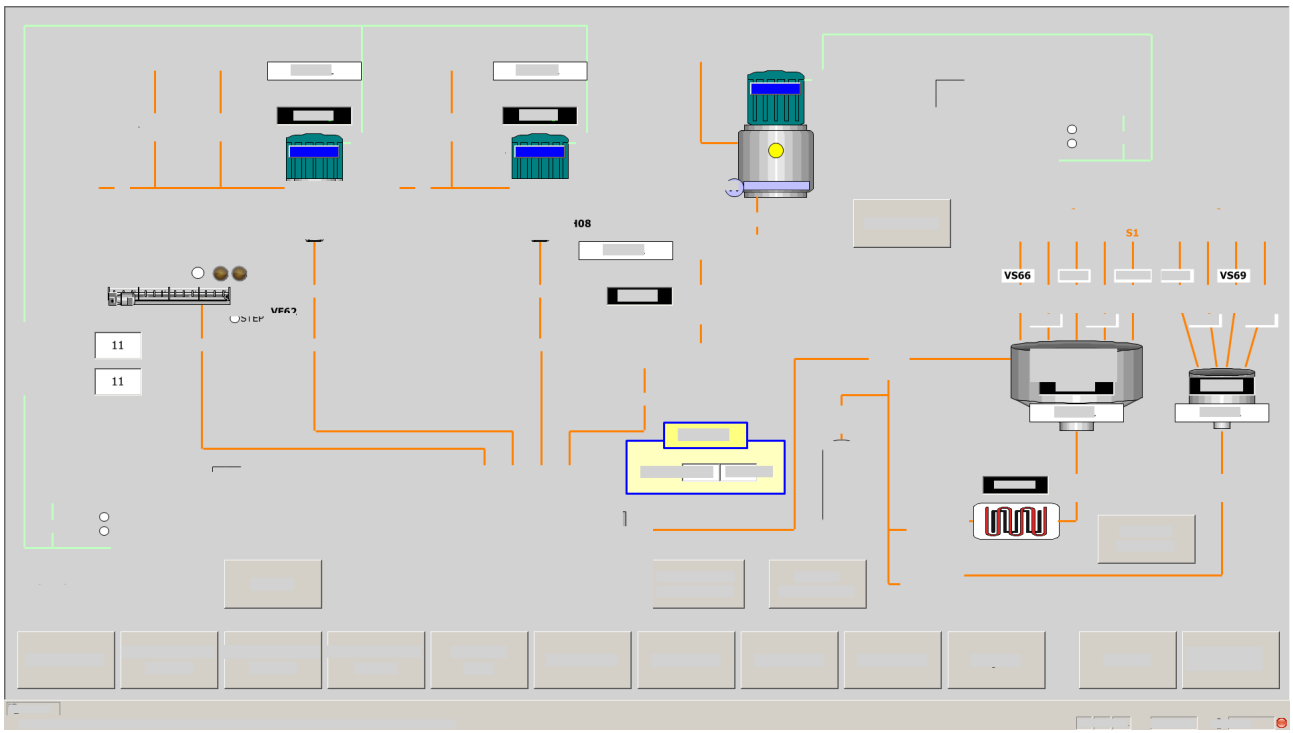

Fig. 7: Imagen después del procesado del Template Matching. También se muestra que en las zonas en las que antes había elementos, ahora se han pintado con el color de fondo. Quedando únicamente los caminos y aquellos elementos no detectados. Tales como maquinaria y botones o tablas.

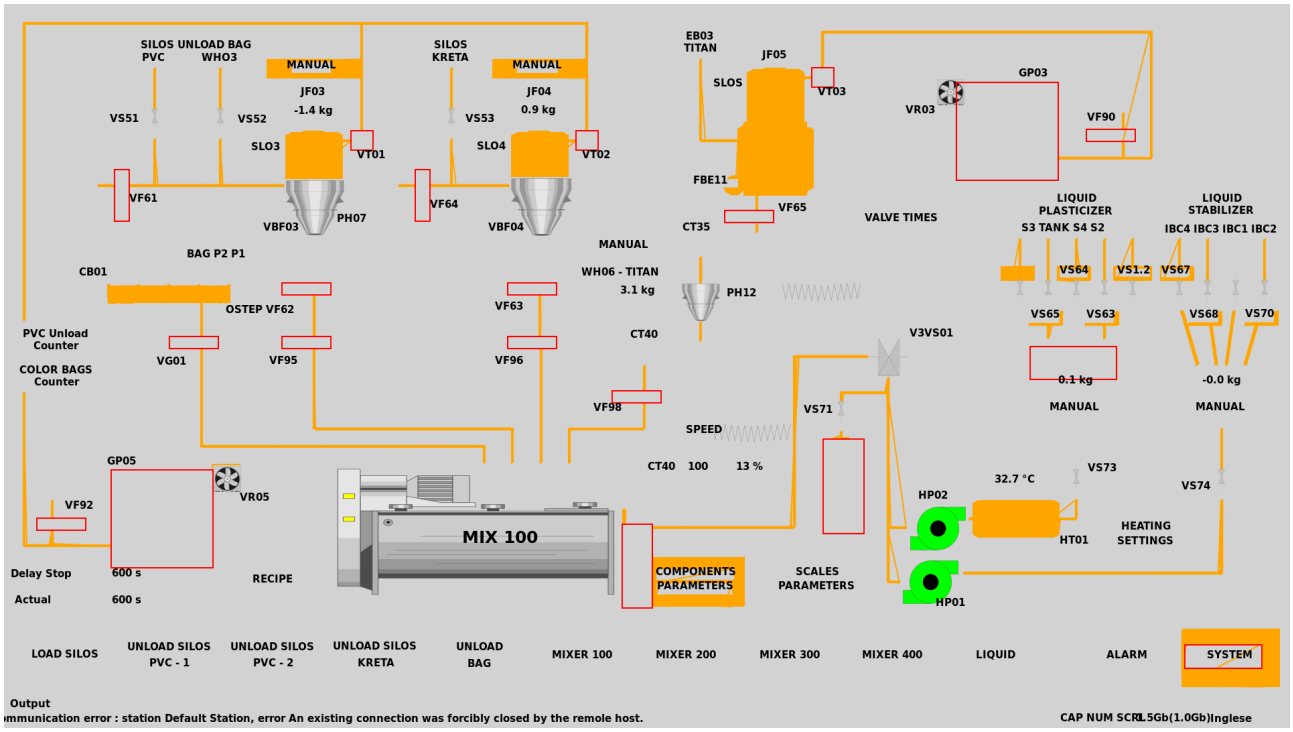

Fig. 8: Imagen final en formato SVG. Se puede observar la construcción de los caminos. Así como la no distinción entre camino y elemento no detectado. Es visible también el deplazamiento de los elementos insertados como SVG. Y la correcta señalización de las “Bounding boxes” de aquellos elementos que no disponían de template en la librería.

6 Conclusión

El objetivo principal de este Trabajo de Final de Grado es el de diseñar una herramienta que ayudase al usuario en la tarea de la renovación de sinópticos. Este objetivo se ha cumplido mediante la implementación de un programa en Python capaz de desgranar una imagen de entrada. Convertirla en un diagrama en formato vectorial escalable como SVG, a la vez que obtiene toda la información de cada uno de los elementos que se hallan en este diagrama.

Este programa permite aligerar la carga de trabajo de un planteamiento de un proyecto de este estilo, reduciendo de

unas cinco a diez horas a unos escasos 10 minutos para una pantalla.

Si bien los resultados no son perfectos, sí que es efectista en cuanto a su objetivo principal que era aligerar todo este proceso inicial a la hora de realizar una renovación de sinóptico, manteniendo la escalabilidad y la maleabilidad de los elementos contenidos en la pantalla.

Cabe destacar los obstáculos que se han ido encontrando por el camino. En un principio se querría haber utilizado un modelo basado en YOLO [15], [16] para la detección de elementos, así como una mejora del OCR basada en cuadrantes para una mayor precisión en la detección.

Otro aspecto por destacar ha sido la falta de datasets tanto de elementos de maquinaria industrial como de sinópticos. Esto ha sido así debido a la naturaleza privada que suele haber en la transparencia a la hora de crear sinópticos. Al final son un producto que crea una empresa y que es de difícil acceso para un usuario promedio.

Aun y con todo esto ha sido realmente fructífero el aprender a manejar estas tecnologías que hemos visto a lo largo de estos años de carrera. Si bien habíamos tocado herramientas como los OCR o Template Matching, me ha servido para aprender las flaquezas que tienen estas en su uso práctico.

Aun con todo, resulta pertinente proponer una serie de mejoras que han sido consideradas a lo largo del desarrollo de este proyecto. Entre dichas mejoras, se plantea la posibilidad de alojar las librerías en la nube, con el objetivo de incrementar la portabilidad y escalabilidad del programa. Esta implementación no solo permitiría optimizar el rendimiento del sistema, sino también facilitar su adaptación a diversos entornos y necesidades. Además, la implementación de YOLO, a la hora de detectar los elementos, eliminaría la necesidad de estas mismas librerías. Otro añadido sería la mejora de las heurísticas de creación de caminos. Aplicar métodos como Manhattan [**?**] para crear los mismos; modificar el algoritmo para que no haya cortes entre ellos; y también detectar posibles elementos, no detectados previamente, mediante el análisis de los clústeres de píxeles que se detectan en la pantalla. Otra mejora más sencilla podría ser la detección de patrones en las direcciones de los píxeles de los caminos para ser agrupados e insertados en el SVG resultante como líneas o líneas poligonales.

A modo de mejora del entorno del programa, se podría implementar un sistema software de modificación del sinóptico en tiempo real, el cual permitiese que el usuario pudiese deseleccionar clases que han sido detectadas. Y a su vez se pudieran modificar las características de sus áreas.

Otra posibilidad podría ser que los elementos una vez insertados aparte del template que cogen de la librería también estuviesen acompañados de características; propias de cada elemento y de cada clase particular. De esta forma se permitiría la realización o implementación de estados dentro de los propios elementos; así como de diferentes variables pertenecientes al mismo; y condiciones que hagan que se activen no se desactiven.

En conclusión, este trabajo ha permitido avanzar en las fases iniciales del diseño de sinópticos, así como de la renovación de los mismos. Se ha conseguido que, a raíz de una imagen, se nos permita extraer muchísima información que no teníamos previamente, y que resulta de gran utilidad sobre todo en las fases iniciales de este tipo de proyectos. Se ha obtenido una mejora tanto en el tiempo que se tarda en empezar con el diseño de estas pantallas, como manteniendo el diseño original. Esto también nos permite ahorrarnos todo el proceso de trabajar en un entorno antiguo o en un entorno desfasado acompañado de un incremento gigantesco de la velocidad a la que realizamos esta tarea.

AGRADECIMIENTOS

Como autor me gustaría agradecer a varias personas.

En primer lugar, a Elitza Maneva por su ayuda y el ha ber confiado en mí. Haberse adaptado a las circunstancias y haber creído en el proyecto cuando se veía todo negro.

En segundo lugar, a Felipe Lumbreras por su inestima ble ayuda. Por haber acudido cuando lo he necesitado, siempre lo más rápido posible. Y por darme otro punto de vista de las cosas.

En tercer lugar, a mi psicóloga Cecilia, por haber cui dado de mí en el lado más emocional. Por haberme impulsado en todos los sentidos para con este proyecto. Y por estar siempre presente.

En cuarto y último lugar, a mi familia y amigos. Los que están y los que por desgracia ya no están. Por haber con fiado en mí a pesar de todo. Y por estar siempre cuando se les necesita, y cuando uno no sabe lo mucho que les necesita.

Sin todos vosotros esto no hubiese sido posible.

Desde lo más profundo de mi corazón, gracias.

REFERENCIAS

[1] Sicma21, "SCADA: ¿Qué es y cómo funciona?" 2021, [Online]. Available: https://www.sicma21.com/scada-que-es-y-como-funciona/. [Accessed: Sep, 17, 2024].

[2] TCI Cutting, "¿Qué es la Industria 4.0 y cómo mejora las empresas?" 2024, [Online]. Available: https://www.tcicutting.com/es/noticias/que-es-la-industria-4-0-y-como-mejora-las-empresas/. [Accessed: Sep, 17, 2024].

[3] S. Choi, Q.-Y. Zhou, and V. Koltun, "Robust reconstruction of indoor scenes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5556–5565.

[4] J. Chai, H. Zeng, A. Li, and E. W. Ngai, "Deep learning in computer vision: A critical review of emerging techniques and application scenarios," *Machine Learning with Applications*, vol. 6, p. 100134, 2021. [Online]. Available: \url{https://www.sciencedirect.com/science/article/pii/S2666827021000670}

[5] X. Ma, Y. Zhou, X. Xu, B. Sun, V. Filev, N. Orlov, Y. Fu, and H. Shi, "Towards layer-wise image vectorization," 2022. [Online]. Available: https://arxiv.org/abs/2206.04655

[6] C. Liu, J. Wu, P. Kohli, and Y. Furukawa, "Raster-to-vector: Revisiting floorplan transformation," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2214–2222.

[7] IBM, “Reconocimiento óptico de caracteres (OCR),” 2024, [Online]. Available: https://www.ibm.com/es-es/think/topics/optical-character-recognition. [Accessed: Oct. 3, 2024].

[8] JaidedAI, “EasyOCR,” 2024, [Online]. Available: https://github.com/JaidedAI/EasyOCR. [Accessed: Oct. 3, 2024].

[9] int3l, “pytesseract: Wrapper for Google’s Tesseract-OCR Engine,” 2018, [Online]. Available: https://github.com/h/pytesseract. [Accessed: Oct. 4, 2024].

[10] Eduardo Zepeda, “OCR con Tesseract, Python y Pytesseract,” 2021, [Online]. Available: https://coffeebytes.dev/es/ocr-con-tesseract-python-y-pytesseract/. [Accessed: Oct. 5, 2024].

[11] VikParuchuri, “Surya,” 2024, [Online]. Available: https://github.com/VikParuchuri/surya. [Accessed: Feb. 1, 2025].

[12] Inductive Automation, “Symbol Factory,” 2024, [Online]. Available: https://www.docs.inductiveautomation.com/docs/8.1/ignition-modules/symbol-factory. [Accessed: Nov. 5, 2024].

[13] N. S. Hashemi, R. B. Aghdam, A. S. B. Ghiasi, and P. Fatemi, “Template matching advances and applications in image analysis,” 2016. [Online]. Available: https://arxiv.org/abs/1610.07231

[14] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, “ Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression ,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2019, pp. 658–666. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00075

[15] Ultralytics, “Datasets,” 2024, [Online]. Available: https://hub.ultralytics.com/datasets?tab=all. [Accessed: Nov. 7, 2024].

[16] Enes Zvornicanin, “YOLO Algorithm,” 2024, [Online]. Available: https://www.baeldung.com/cs/yolo-algorithm. [Accessed: Nov. 8, 2024].