UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

DOCUMENTO I

INTRODUCCIÓN Y OBJETIVOS

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Resumen

Este Trabajo Fin de Máster forma parte de un proyecto para el desarrollo de un sistema para el control de la calidad de carriles de tren y su despliegue en las instalaciones de la empresa siderúrgica ArcelorMittal.

Este proyecto emplea tecnología de visión artificial: se utilizan cuatro cámaras y cuatro láseres para captar imágenes de los carriles en movimiento, con las que luego puede reconstruirse su perfil para compararlo con un modelo prefijado.

De entre todas las funcionalidades que deben implementarse como parte del proyecto, como son la adquisición de imágenes de carriles, la medición de los mismos, la calibración de las cámaras, la comunicación, el almacenamiento y la visualización de los resultados y la especificación de modelos de carril y normas, este TFM se centra en tres de ellas y su relación con el resto del sistema: la calibración de las cámaras, el procesamiento de imágenes y la visualización y el tratamiento de los resultados obtenidos.

Abstract

This Master's Final Project is part of a larger project to develop a quality control system for ArcelorMittal's rail production facilities.

The project employs machine vision technology: four cameras and four laser emitters are used in order to capture images of moving rails. Afterwards, line extraction is applied in order to generate a complete rail profile, which is then compared with a known rail model.

Of all the features that must be implemented as part of the larger project, such as image acquisition, rail measurement, camera calibration, result communication, storage and visualization and rail model and standard management, this Master's Final Project is mostly concerned with three of them and their relationship with the rest of the system: camera calibration, image processing and treatment of measurement results.

Contenido

1. Introducción

La comprobación de la calidad de los productos es una parte esencial del funcionamiento de una fábrica. En las últimas décadas, el mercado ha ido volviéndose cada vez más exigente en lo que a este tema se refiere, con la aparición de normas reguladoras de la calidad y su gestión, en general y en determinados sectores, lo que ha provocado que las empresas que no aseguran un nivel de calidad de acuerdo a estas exigencias no puedan llegar a prosperar.

La calidad es una de las condiciones necesarias para mantener la competitividad de una empresa frente al resto de empresas del mismo sector, por lo que las políticas de calidad de las empresas van dirigidas esencialmente a dos aspectos fundamentales, relacionados entre sí:

- La satisfacción de las necesidades y expectativas del cliente por parte de la empresa suministradora, ofreciendo la calidad deseada en los productos y servicios, y además manteniendo este nivel de calidad de forma constante.
- La satisfacción de las necesidades e intereses internos de la propia organización, haciendo que la calidad tenga un coste óptimo, mediante el uso eficiente de los recursos necesarios.

El cumplimiento de la calidad es una de las mayores fuentes de gasto para las empresas actuales, pero todavía más costoso es el no cumplimiento de la calidad, ya que esto haría de la empresa en cuestión una empresa no competitiva.

Por lo tanto, mediante un sistema que asegure la calidad de un producto o una línea de productos, la empresa puede optimizar los costes, invirtiendo en la evaluación de la calidad, para poder reducir al máximo los costes de la “no calidad”.

El presente proyecto aborda la necesidad de la empresa siderúrgica ArcelorMittal de disponer de un sistema que mida y asegure la calidad de uno de los productos que esta empresa fabrica: los carriles de tren.

Un carril de tren, también llamado raíl o riel, es cada una de las “barras” metálicas sobre las que se desplazan los trenes, grúas, tranvías, etc. Los carriles son las partes fundamentales de todas las vías férreas, actuando como soporte, guía e incluso elemento conductor de la corriente eléctrica. La Figura 1 muestra un carril de ferrocarril.

Figura 1: Carril de ferrocarril

Las cualidades principales de todos los carriles son su material, su forma y su peso. La forma del carril viene definida por su perfil. Dependiendo de qué tipo de carril sea, tendrá una u otra forma.

Un perfil de carril es la vista de la sección transversal del mismo, como se puede observar en la Figura 2. Por lo tanto, en un carril se tienen infinitos perfiles de carril, dependiendo de en qué longitud del mismo se practique la sección transversal.

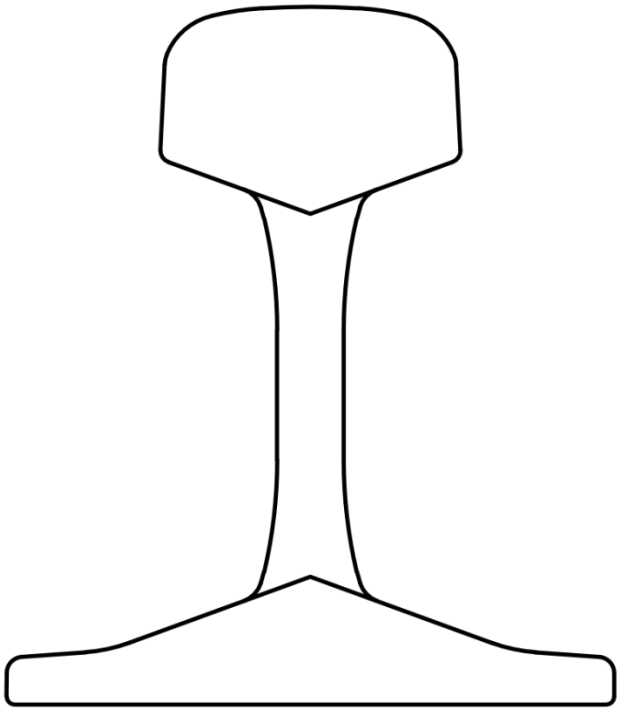


Figura 2: Perfil de un carril de ferrocarril

El sistema que se desarrolla en este proyecto tiene la finalidad de comprobar la calidad de la forma del carril mediante la medición de los valores de determinadas dimensiones. Esto debe hacerse en múltiples secciones a lo largo del carril, para conocer la evolución de estos valores.

La comprobación de la forma del carril determina su calidad. Esta comprobación se puede llevar a cabo mediante diversas técnicas, y se debe elegir entre una u otra dependiendo el nivel de automatización que se desee obtener. Este proyecto aborda la comprobación de la calidad del carril mediante técnicas de visión por computador.

2. Proceso industrial

2.1. Modelos de carril

Para poder realizar mediciones sobre los carriles que se fabrican, se debe tener un modelo de carril con el que comparar el resultado obtenido mediante el proceso industrial. Este modelo es el que se ha seguido durante todo el proceso de fabricación del carril y es el objetivo deseado. Es decir: cuanto más se parezca el producto final obtenido al modelo, mayor será su calidad.

Existen múltiples modelos de carril que se fabrican en la actualidad, dependiendo de su finalidad, bien sean para ferrocarriles, grúas, u otros usos. Estos modelos deberán ser pasados al sistema que se ha creado mediante un modelo que pueda ser entendido por el mismo. Para ello debe tener un formato que sea capaz de ser interpretado tanto por el usuario que lo crea como por el sistema que lo utiliza.

2.2. Medición de la calidad

2.2.1. Legislación y normas

Las normas relativas a las aplicaciones ferroviarias son las que recogen aspectos generales sobre la fabricación y diseño de los carriles, ya sean para tren, para grúa o para cualquier otra aplicación.

Los aspectos especificados en estas normas son muy extensos y abarcan desde la fabricación del propio carril, su identificación, su marcado y los materiales de los que deben estar compuestos, hasta las comprobaciones de seguridad que se han de realizar para asegurar su integridad y su calidad.

En la actualidad existen multitud de normas dependiendo del país o continente destinatario del carril, o bien dependiendo del uso que se le vaya a dar al carril. Algunas de las normas más comunes en la actualidad son:

- **EN**. Creadas por el *CEN*, o *Comité Europeo de Normalización*, como una norma de ámbito europeo. La norma es *EN-13674.*
- **UNE**. Estas normas han sido creadas por AENOR para ámbito español, como es la norma *UNE-EN-13674*, la cual es una traducción de la norma *EN-13674.*
- **AREMA** (*American Railway Engineering and Maintenance of way Association)*, una asociación estadounidense. En este caso la norma a utilizar es la *Arema Rails*.

En lo que respecta al control de calidad, estas normas contienen principalmente dos grandes apartados, que se tienen en cuenta para la comprobación de la integridad y calidad del producto fabricado:

- **Ensayos de calificación**, que recogen todos los requisitos de prestaciones, es decir, toda la serie de ensayos a los que el carril deberá ser sometido, como por ejemplo, ensayos de fatiga, ensayos de resistencia a la fatiga y alargamiento, etc. No se realizan sobre todos los carriles fabricados (de hecho, incluyen pruebas destructivas), sino que su objetivo es controlar la calidad del proceso de fabricación en general.

- **Ensayos de aceptación**, cuyo objetivo es verificar aquellas características de los carriles, como su dureza, sus dimensiones, el tratamiento térmico del acero, etc. Se realizan sobre todos los carriles que se fabrican.

El sistema que es objeto de este proyecto es responsable de la comprobación de las dimensiones de los carriles, dentro de los ensayos de aceptación.

2.2.1.1. Tolerancias dimensionales

En estos apartados de las normas de calidad se describe la diferencia máxima que puede existir entre el perfil del carril fabricado y el perfil normalizado. Si esta diferencia es menor que la tolerancia, entonces ese perfil cumple la norma de calidad; en caso contrario se dice que no la cumple.

Estas tolerancias se deberán verificar para una serie de dimensiones que deberán haber sido establecidas previamente, de acuerdo a la norma aplicable.

A su vez, estas dimensiones deberán haber sido extraídas de las normas. En el caso de la norma *UNE-EN-13674*, las dimensiones a tener en cuenta son las mostradas en la Figura 3.

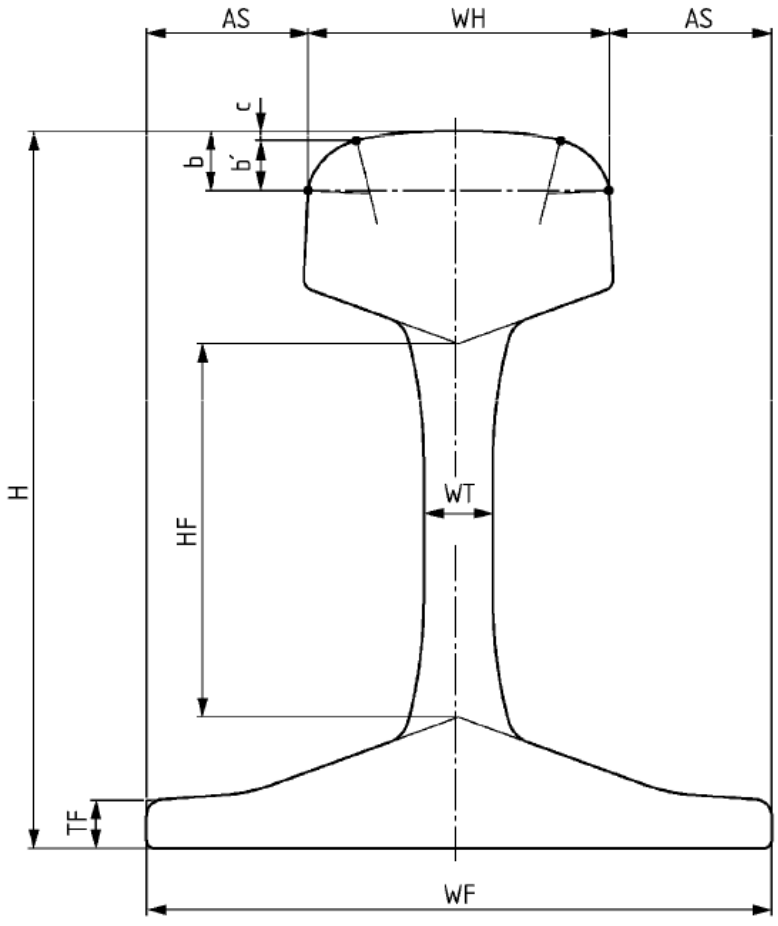


Figura 3: Dimensiones principales del carril según la norma "*UNE-EN-13674*"

La Figura 4 muestra la tabla de la norma *UNE-EN-13674*, donde se recogen todas las tolerancias para las dimensiones a medir (que se pueden ver en la Figura 3) en los perfiles de los carriles.

*Puntos de referencia (véase la figura E.1)			Clase de perfil (dimensiones en mm)		Plantillas, número de las figuras (véase el anexo E)
Ubicación/propiedad		Símbolo	X	Y	
Altura de carril [a]	< 165 mm	*H	± 0,5	+ 0,5 – 1,0	E.3
	≥ 165 mm		± 0,6	+ 0,6 – 1,1	
Perfil de la cabeza de carril – Enderezado clase A			+ 0,6 – 0,3	+ 0,6 – 0,3	E.4
– Enderezado clase B			± 0,6	± 0,6	
Anchura de la cabeza de carril		*WH	± 0,5	+ 0,6 – 0,5	E.5
Asimetría de carril		*As	± 1,2	± 1,2	E.6, E.7
Altura de la zona de embridaje	< 165 mm	*HF	± 0,5	± 0,5	E.8
	≥ 165 mm		± 0,6	± 0,6	
Espesor del alma		*WT	+ 1,0 – 0,5	+ 1,0 – 0,5	E.9
Anchura del patín del carril		*WF	± 1,0	+ 1,5 – 1,0	E.10
Espesor del ala del patín		*TF	+ 0,75 – 0,5	+ 0,75 – 0,5	E.11
Concavidad de la base del patín			0,3 máx.	0,3 máx.	

[a] La variación total de altura a lo largo de cualquier carril no debe ser mayor de 1 mm para carriles < 165 mm, y de 1,2 mm para carriles ≥ 165 mm.

Figura 4: Tolerancias dimensionales en la norma UNE-EN-13674

2.2.2. Tecnologías disponibles

Existen diversas tecnologías para llevar a cabo la comprobación de las tolerancias dimensionales mencionadas en el epígrafe anterior. Se deberá elegir entre una u otra dependiendo del nivel de precisión y automatización que se busque.

2.2.2.1. Medición convencional

Esta medición es la más sencilla y por lo tanto no tiene ningún tipo de automatización. Consiste en medir las dimensiones directamente sobre el carril fabricado de forma manual, utilizando cualquier tipo de instrumento de medición, como por ejemplo un metro o un calibre.

Este tipo de medición tiene una serie de problemas que hacen que sea inviable:

- Aumenta significativamente el tiempo de comprobación.
- Está sujeta a una mayor cantidad de errores humanos.
- Debido al problema anterior, se obtendrá una menor precisión de medición.
- En caso de que las medidas que se deseen medir no sean triviales, este método resultará muy complejo de aplicar correctamente.

2.2.2.2. Plantillas mecánicas

Este método surge como una forma de normalizar la medición que se realizaba en el caso anterior, haciendo que la forma de tomar las medidas de las dimensiones sea mucho más sencilla y más rápida que de la forma convencional.

Este método se realiza mediante la medición de unas plantillas mecánicas, que se construyen expresamente para cada tipo de carril que se desee medir, y también para cada dimensión.

Estas plantillas están sacadas directamente del prototipo que se propone en la norma para la medición de las dimensiones del carril. La Figura 5 muestra la plantilla mecánica para la medición de la altura del carril.

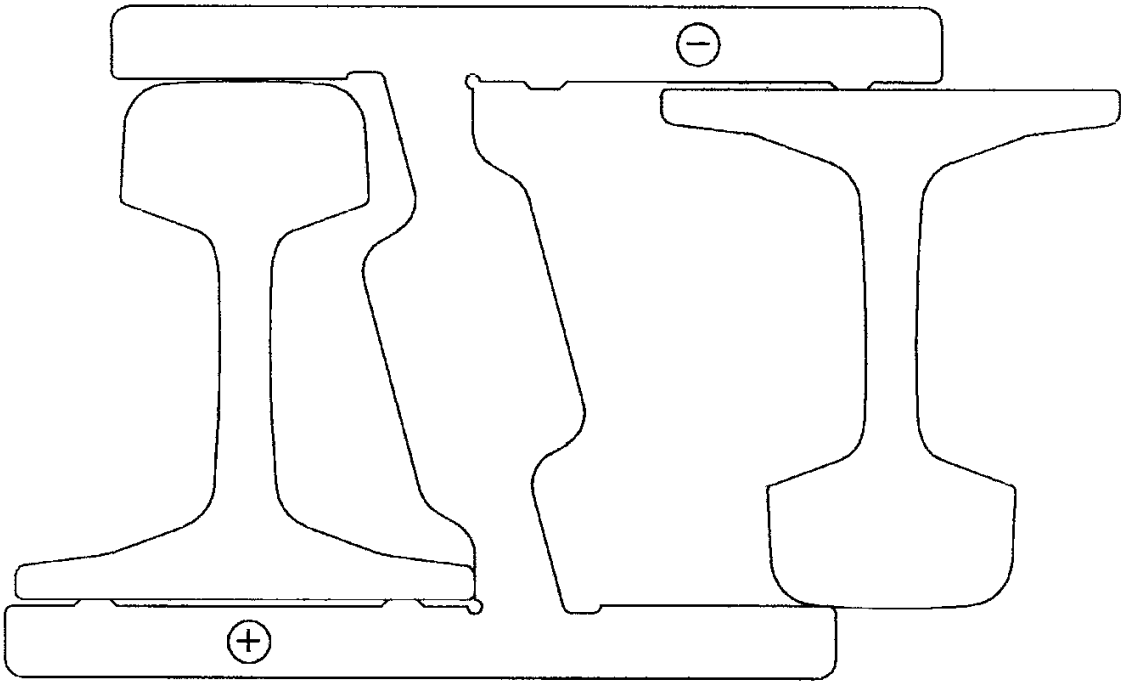


Figura 5: Plantilla mecánica para la medición de la altura del carril

Esta forma de medir las dimensiones de los perfiles del carril fabricado es mucho más precisa, y es la recomendada en las normas sobre fabricación de carriles. Sin embargo, también presenta una serie de problemas:

- Se necesita una plantilla de calibración específica para cada tipo de carril, ya que las dimensiones pueden variar entre diferentes tipos.
- Se necesita también una plantilla de calibración específica para cada norma, ya que las tolerancias admitidas en cada una de las normas varían.
- Es más sencilla que la medición convencional, pero aun así implica un tiempo de comprobación considerable.

2.2.2.3. Visión artificial

Esta tecnología surge como una automatización de las plantillas mecánicas vista en el epígrafe anterior y gracias a ella se consigue que la medición de los valores dimensionales de los carriles sean medidos de forma totalmente automática, sin necesidad de que el operario intervenga.

2.3. Triangulación óptica

La triangulación óptica es una técnica de visión artificial en la que, mediante un emisor de luz (típicamente un láser), se proyecta luz sobre la superficie que se desea inspeccionar, y una cámara situada en otra posición diferente a la del láser (típicamente a 45º de la recta perpendicular que pasa por el láser) captura la luz láser reflejada por la superficie, visualizando así el contorno de la superficie u objeto a analizar, tal y como se puede observar en la Figura 6.

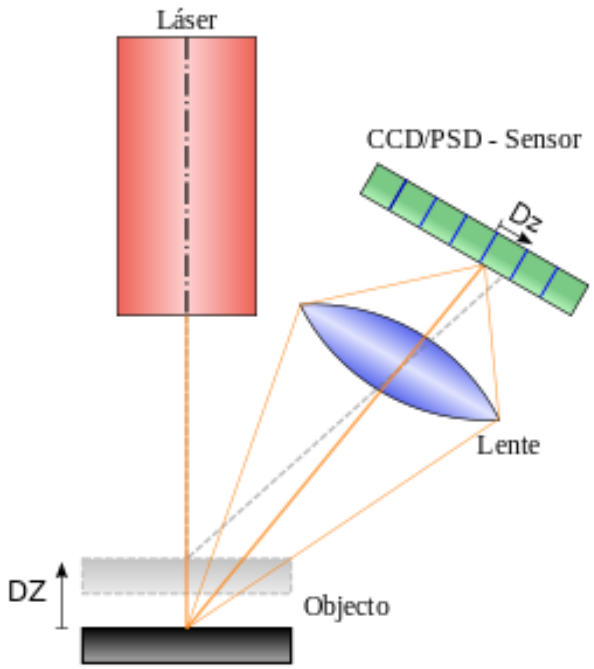


Figura 6: Esquema de la triangulación óptica

Esta medición estará condicionada totalmente a la óptica que reciba la cámara de la línea dibujada por el láser. Es decir: si hay mucha luz ambiente en la escena, la línea del láser aparecerá más desdibujada que si, por el contrario, no hubiese luz.

Por otra parte, ya que sólo interesa captar una franja muy específica de todo el espectro de luz (la que coincida con la longitud nominal del emisor láser), se podrá usar un filtro óptico que deje pasar tan sólo el color de la luz de láser, haciendo más sencillo así el trabajo de analizar la imagen recogida.

A esta técnica se le denomina triangulación ya que cada uno de los puntos de la línea dibujada por el láser, la cámara y el emisor del láser forman un triángulo. Como la longitud del lado del triángulo definido por la cámara y el emisor del láser es conocida, al igual que el ángulo del vértice del emisor de láser y el ángulo del vértice de la cámara puede ser determinado mirando la ubicación del punto del láser en la cámara. Estos tres valores permiten determinar de forma matemática el resto de las dimensiones del triángulo y, por tanto, la posición de cada punto en el espacio.

Al conocer todos los puntos de la superficie de un objeto se pueden calcular medidas reales del objeto, e incluso reconstruir su forma en tres dimensiones.

Este sistema permite medir con gran precisión todos los puntos de una superficie u objeto, al nivel de milésimas de milímetro, por lo que es muy útil para todo tipo de procesos industriales en la actualidad. Un ejemplo de esto se puede observar en la Figura 7.

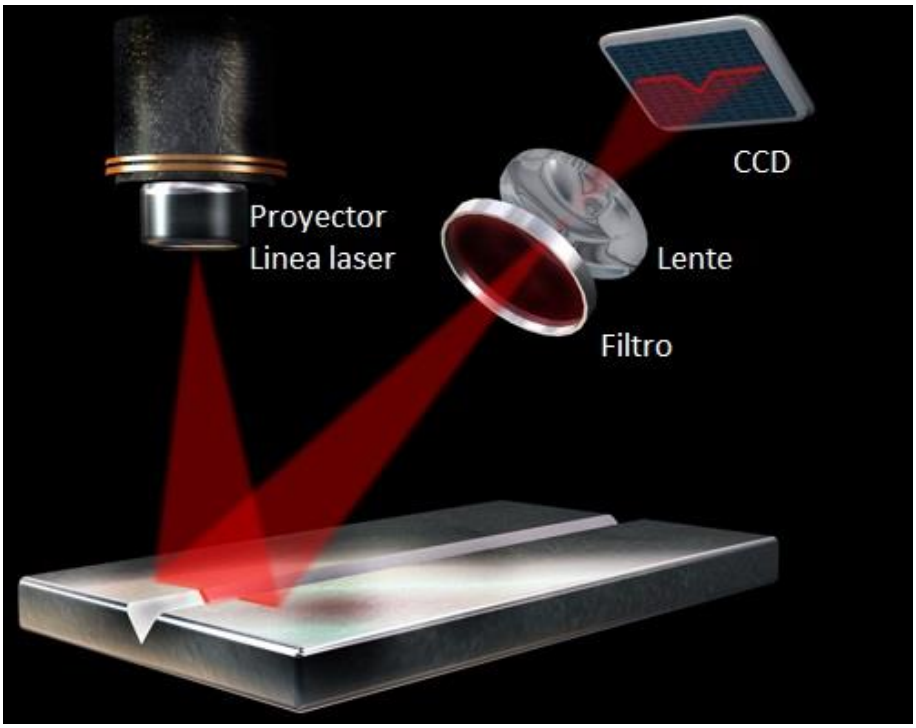


Figura 7: Triangulación óptica para la inspección de objetos

3. Sistema de medición de carriles

3.1. Objetivo

La automatización de la medición permite agilizar el proceso, que de otro modo sería un cuello de botella en la fabricación de carriles, reducir el potencial de errores humanos, almacenar fácilmente la información obtenida para fines estadísticos y presentarla de forma atractiva al cliente, llegado el caso, entre otras ventajas.

3.2. Sistema anterior

Este proyecto busca reemplazar el sistema empleado anteriormente por ArcelorMittal para la medición con el objetivo de reducir costes, modernizar los equipos empleados y obtener funcionalidades adicionales.

Este sistema antiguo es la Galga de Medición de Perfil (PMG) del instituto de investigación Joanneum Research, que emplea también tecnología de visión artificial, con cuatro cámaras y cuatro láseres que cubren todo el contorno del carril, empleando el sistema descrito en la sección 2.3. En la Figura 8 pueden verse una de las ventanas del software de la PMG, mientras que en la Figura 9 se muestra la estructura empleada para la medición.

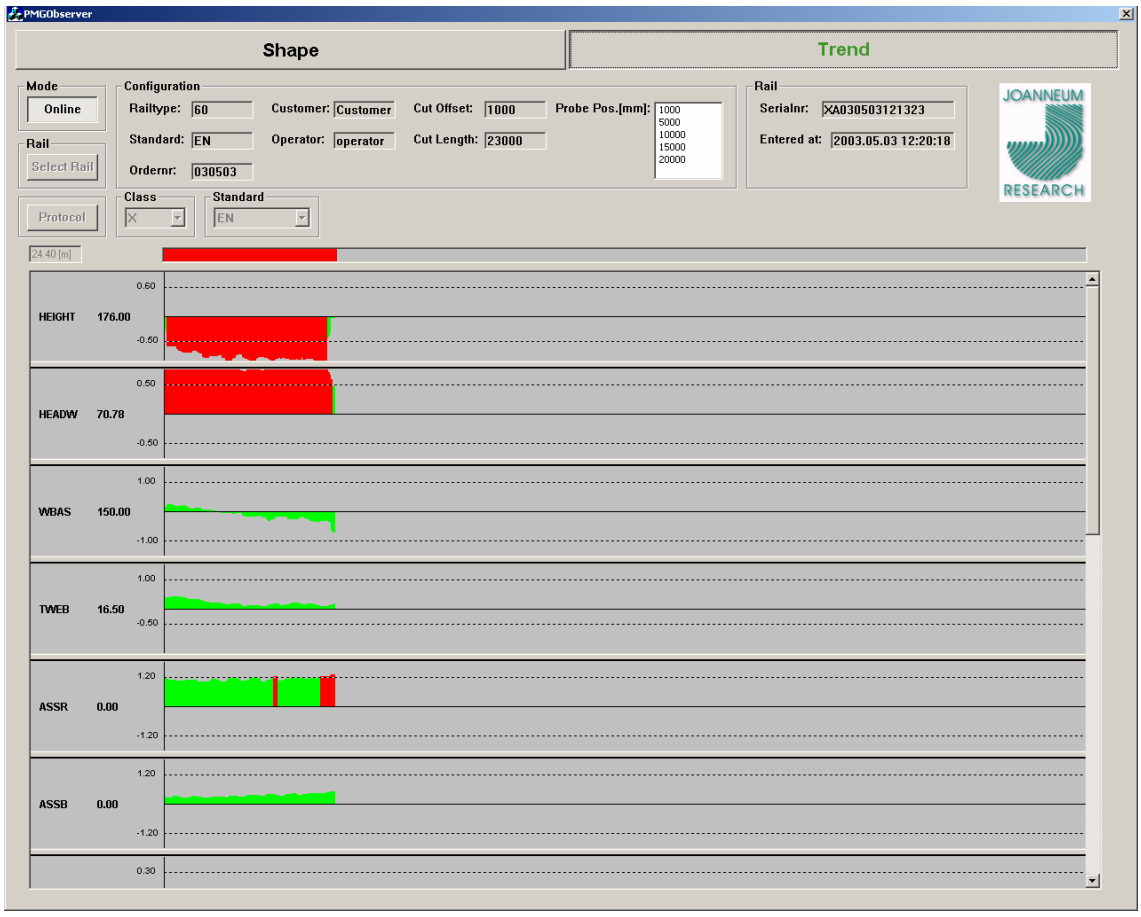


Figura 8: Interfaz gráfica del programa visualizador incluido en la PMG

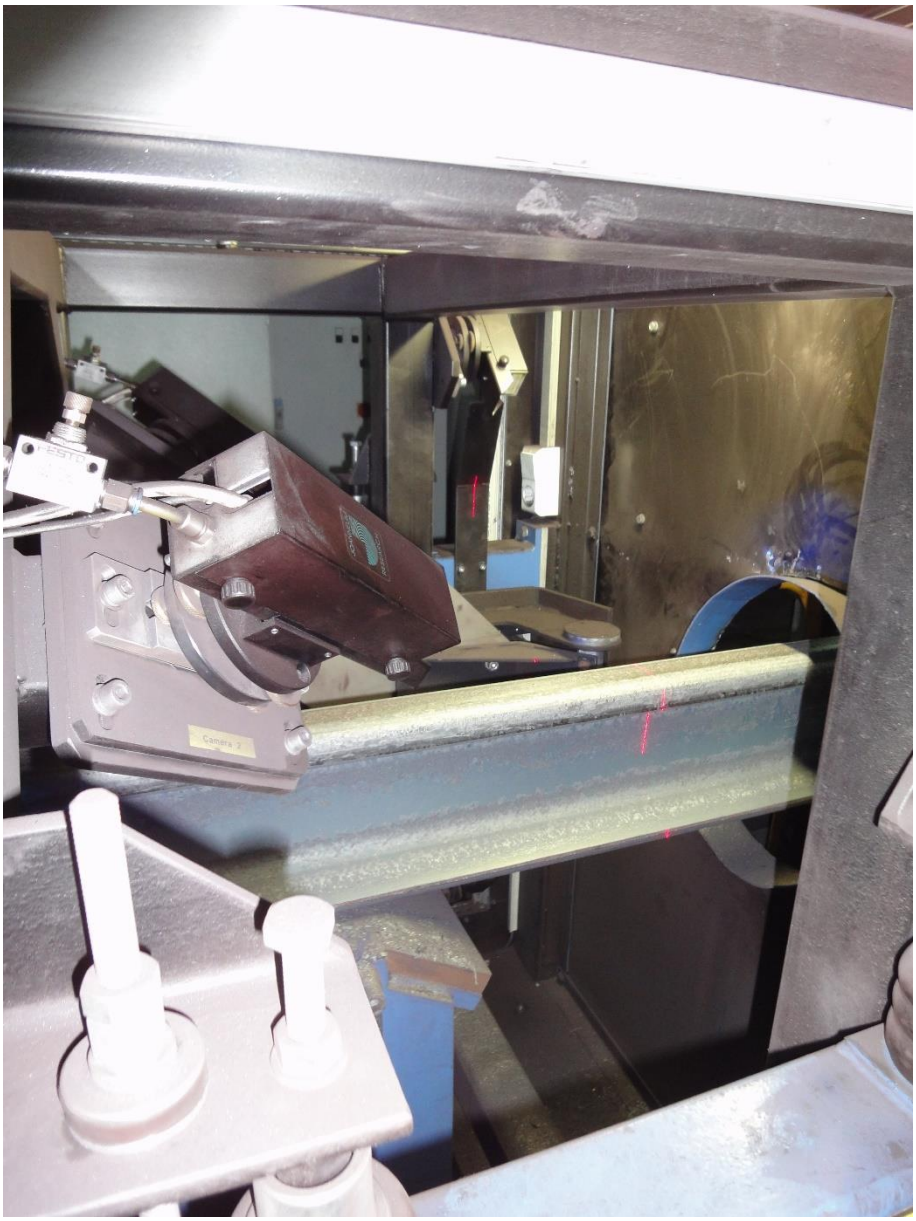


Figura 9: Vista de la PMG, incluyendo una de las cámaras

3.3. Nuevo sistema

Este proyecto trata el desarrollo de un sistema de visión artificial capaz de sustituir a la PMG. Este nuevo sistema emplea también la tecnología de visión artificial comentada en la sección 2.2.2.3, con cuatro cámaras para poder captar todo el contorno del carril.

3.4. Requisitos globales del sistema

Los requisitos generales del nuevo sistema son los siguientes:

1. Medición
   1.1. Se medirán sin necesidad de intervención humana los valores de las dimensiones[1] de los carriles según se fabriquen.
   1.2. De cada carril medido se almacenarán determinados datos:
      1.2.1. El identificador del carril.
      1.2.2. El tipo o modelo de carril.
      1.2.3. La norma seguida.
      1.2.4. La fecha y la hora de la medición.
      1.2.5. El valor de cada dimensión medida.
   1.3. Los datos almacenados se comunicarán al ordenador de proceso.
   1.4. Los operadores podrán conocer en tiempo real los resultados parciales y finales de la medición y los errores que se produzcan en ella.
   1.5. Se calibrarán los componentes ópticos del sistema cuando los operadores lo soliciten.
2. Configuración
   2.1. Podrán gestionarse los modelos o tipos de carril reconocidos por el sistema.
   2.2. Podrán gestionarse las normas reconocidas por el sistema.
3. Visualización
   3.1. Los operadores podrán acceder desde equipos distintos de la red local a los resultados de la medición según se vayan obteniendo.
   3.2. Los operadores podrán acceder a los resultados de mediciones anteriores.
   3.3. Los operadores podrán controlar qué dimensiones se muestran.
4. No funcionales
   4.1. El sistema de medición podrá procesar las secciones de carril a una velocidad no inferior a la velocidad a la que deban captarse las imágenes.
   4.2. El sistema de medición seguirá los protocolos definidos para la comunicación con el ordenador de proceso y el PLC que controlan la producción.

[1] La relación de las dimensiones medidas queda fuera del ámbito de este TFM.

4. Trabajo

4.1. Objetivos

En relación con los requisitos globales del proyecto, este Trabajo Fin de Máster trata particularmente:

- La estructura general en la que se integran las partes que conforman el proyecto.
- La calibración de las cámaras empleadas para la medición.
- El procesamiento de las imágenes previo a la medición.
- La visualización y la comunicación de los resultados de la medición.
- La implantación del sistema en la línea de producción de carriles.

4.2. Documentos incluidos

La memoria de este Trabajo Fin de Máster está compuesta por múltiples documentos principales y anexos. Los documentos principales son:

1. **Introducción y objetivos:** Este documento introduce el problema al que hace frente el proyecto, define algunos conceptos relacionados, explica brevemente la tecnología existente y establece el alcance del trabajo.
2. **Diseño general:** Detalla los componentes de que consta el proyecto y la forma en que se relacionan entre sí.
3. **Medidor:** Especifica el diseño del programa medidor, su funcionamiento y el modo en que se relaciona con otros elementos.
4. **Comunicaciones:** Explica el programa que gestiona las comunicaciones del medidor.
5. **Calibración:** Detalla el algoritmo empleado para la calibración de los elementos ópticos del medidor.
6. **Visualizador:** Especifica el diseño del programa visualizador y de la funcionalidad de visualización incluida en el programa medidor.
7. **Planificación y presupuesto:** Detalla la planificación seguida en el desarrollo del proyecto y del presente Trabajo Fin de Máster, y también el presupuesto del proyecto.

Los anexos incluidos son:

A. **Elección del sistema de visión:** Detalla el proceso seguido para determinar la geometría de la estructura empleada para la medición.
B. **Manual de ejecución de las comunicaciones:** Explica el modo de ejecutar el programa de comunicaciones y los simuladores que lo acompañan.
C. **Comunicación con el PLC:** Especifica el protocolo de comunicaciones entre el sistema y el PLC que gestiona el movimiento de los carriles.
D. **Comunicación con el PA:** Especifica el protocolo de comunicaciones entre el sistema y el ordenador de proceso.
E. **Formato del fichero de resultados:** Especifica el formato en el que se almacenan y comunican los resultados de la medición.
F. **Tests globales del medidor:** Detalla los resultados de las pruebas de medición practicadas con carriles reales.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

DOCUMENTO II

DISEÑO GENERAL

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Introducción

Este proyecto está conformado por un conjunto de entidades software y hardware que se relacionan entre sí y con elementos externos.

Ha de tenerse en cuenta la existencia de múltiples componentes, tanto hardware (como por ejemplo las cámaras, el PLC y el ordenador de proceso) como software (sistema medidor, visualizador, configurador, ficheros de configuración, etc.) que han de trabajar de forma conjunta para lograr un funcionamiento correcto.

Todo esto ha de coordinarse para producir una salida que posteriormente ha de poder ser utilizada en otros sistemas.

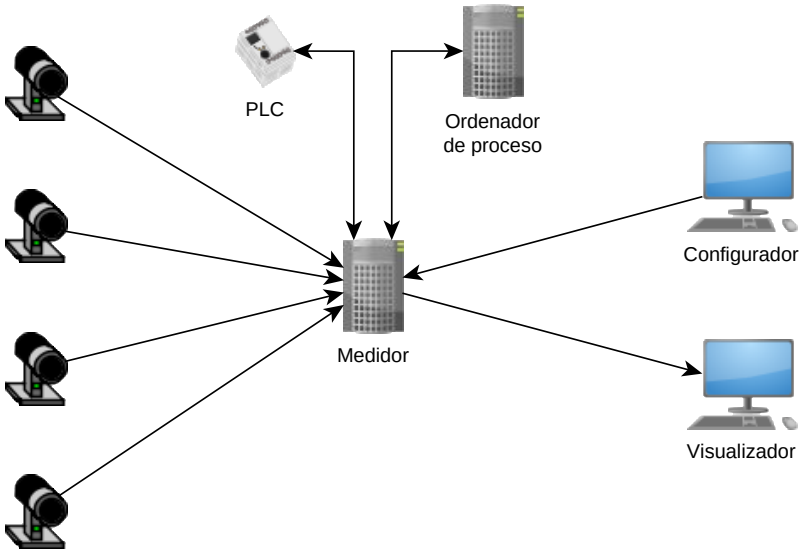


Figura 1: Diagrama simplificado del sistema

Como puede verse en la Figura 1, el sistema incluye, a grandes rasgos, los siguientes elementos:

- Un servidor de medición, o medidor, capaz de procesar las imágenes de las cámaras y extraer medidas de los perfiles que inspecciona, comunicándose también con el PLC y el ordenador de proceso.
- Un programa configurador para generar datos de perfiles y normas que luego pueda emplear el medidor.
- Un programa visualizador independiente que puede acceder remotamente a las mediciones que se estén realizando, así como a las históricas.
- Un subsistema de captura de imagen que está compuesto por cuatro cámaras, que capturan imágenes, y cuatro emisores láser que proyectan un patrón lineal sobre la superficie del carril y que se utilizan para “dibujar” su contorno.

2. Arquitectura general del sistema

El sistema implementado está formado por una serie de componentes, tanto hardware como software, así como por componentes externos que son necesarios para llevar a cabo la correcta ejecución del sistema.

De forma detallada, la estructura general del sistema puede verse en la Figura 2. A continuación, se describirá brevemente el cometido de cada módulo software implementado como parte del proyecto, así como su relación con otros elementos.

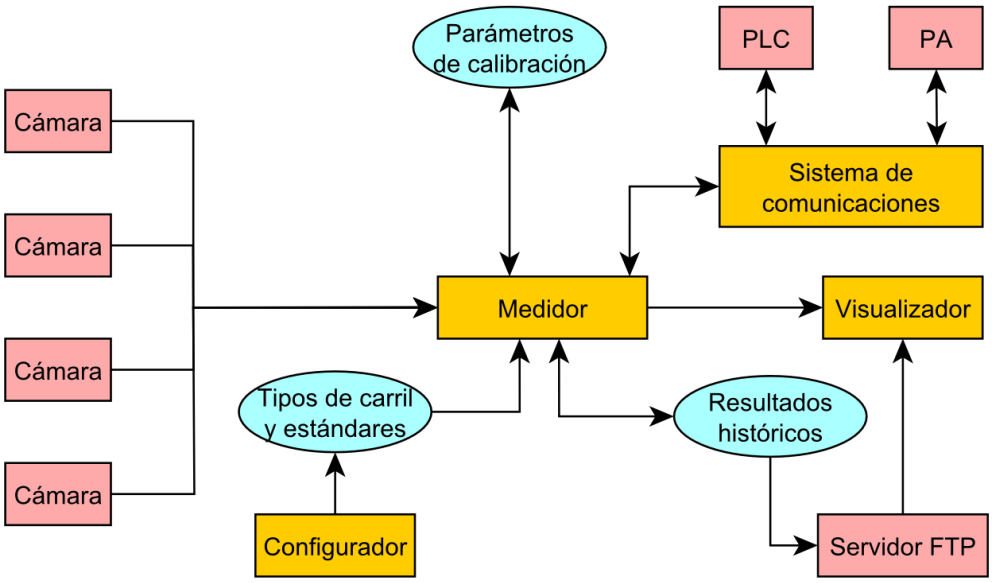


Figura 2: Estructura del sistema

En la Figura 2 se pueden observar diferentes tipos de componentes. Según su color en la figura estos son:

- En amarillo, los programas implementados como parte del presente proyecto:
  - Medidor: Lleva a cabo la medición de los carriles.
  - Sistema de comunicaciones: Comunica el programa medidor con sistemas externos.
  - Configurador: Genera un fichero de tipos de carril y normas para su uso por el medidor.
  - Visualizador: Muestra los resultados obtenidos por el medidor.
- En azul, los datos generados o consumidos por programas que forman parte del sistema.
  - Tipos de carril y estándares: Fichero que describe los tipos de carril y las normas que el programa medidor emplea.
  - Parámetros de calibración: Parámetros de las cámaras utilizadas para la medición, que el programa medidor emplea para traducir las coordenadas de los píxeles de las imágenes captadas por ellas a coordenadas del mundo.
  - Resultados históricos: Resultados de las mediciones de carriles anteriores.

- En rojo, los equipos externos o no incluidos en el proyecto, pero que tienen una estrecha relación con el mismo.
    - Cámaras: Cuatro cámaras situadas alrededor del carril que se mide y que captan imágenes para el programa medidor.
    - Servidor FTP: Servidor que proporciona los ficheros de resultados antiguos al programa visualizador.
    - PLC: Dispositivo que avisa al programa medidor de la entrada y la salida de carriles y que controla el despliegue de la plantilla de calibración.
    - PA: Ordenador de proceso, que envía al programa medidor los metadatos de los carriles que van a llegar y recibe de él los resultados de la medición.

La arquitectura software del sistema desarrollado está compuesta por varios programas junto con una serie de bibliotecas de enlace dinámico o DLL. Esta estructura se explica con mayor detalle en la sección 4.

3. Arquitectura hardware del sistema

El hardware relacionado con el sistema puede dividirse en dos tipos: el hardware específico del sistema y el hardware externo al mismo.

El hardware específico se utiliza en exclusiva para llevar a cabo las labores de este sistema:

- Cuatro cámaras, encargadas de tomar las imágenes de los perfiles cuando el *PLC* lo indique.
- Cuatro láseres, encargados de proyectar un patrón lineal que haga posible la extracción del perfil del carril.
- Una plantilla de calibración, que permite calcular la posición de las cámaras relativa al plano láser para poder interpretar correctamente las imágenes captadas.

Por otra parte, se tiene también un hardware externo, que es el encargado tanto de controlar la medición como de almacenar los resultados de las mediciones que se realicen. El hardware externo que se va a utilizar cuando el sistema se ponga en producción es el siguiente:

- Un autómata lógico programable o *PLC*. Será el encargado de controlar en qué momento se deberán tomar imágenes. Esto se realizará mediante el envío de una señal analógica (un tren de pulsos generado por un encoder) que llegará directamente a las cámaras indicando así la captura de una imagen.
  El autómata tendrá también la responsabilidad de indicar al medidor la entrada o salida de los carriles al punto de inspección. Además, existe una plantilla de calibración, una placa metálica que ha de desplegarse en paralelo al plano láser para determinar la posición de las cámaras y poder compensar su desplazamiento. Puesto que esta plantilla ocupa, cuando está desplegada, un lugar por el que deben pasar los carriles, debe existir un mecanismo para desplegarla y replegarla solamente cuando sea necesario. Este mecanismo está controlado por el medidor, y el *PLC* hace las veces de intermediario, que debe ordenar el despliegue o recogida de la plantilla de calibración cuando el medidor lo solicite, tras comprobar que el sistema está en un estado adecuado (es decir, que no hay un carril presente con el que pueda colisionar la plantilla).
- Un ordenador de proceso o *PA*. Este ordenador es el encargado de controlar y supervisar el funcionamiento de la línea de producción, por lo que será el responsable de enviar la información relativa a los carriles a medir. El ordenador de proceso puede solicitar los resultados de las mediciones de cualquier carril medido, de modo que se le envíen todas las mediciones que se estén realizando.

4. Arquitectura software del sistema

Este sistema se divide en tres programas principales:

- El programa *medidor,* que es el centro de toda la arquitectura. El medidor es el encargado de tomar las medidas de los carriles mediante técnicas de procesamiento de imágenes.
- El programa *configurador*, que es el encargado de crear, editar y borrar los ficheros de configuración que el programa medidor necesite para llevar a cabo su actividad.
- El programa *visualizador*, que procesa los archivos que genere el programa medidor mostrando su contenido al usuario final.

Una secuenciación lógica para llegar al resultado final de la medición con éxito sería la siguiente:

- El programa configurador prepara el archivo de configuración necesario para la ejecución del programa medidor. Para esto no es necesario que el medidor esté en funcionamiento, ni que se encuentre en la misma máquina.
- El programa medidor captura las imágenes que se reciben de las cámaras e inicia su procesamiento. Se extraen los contornos de los carriles y se miden los valores de sus dimensiones. Estos valores se comparan con el fichero de configuración para establecer si se cumple o no con las tolerancias preestablecidas, y el resultado se muestra al usuario inmediatamente durante el proceso de medición, y se almacena además en ficheros históricos.
- El programa visualizador se conecta al medidor desde otro equipo y muestra los resultados de la medición en tiempo real. También puede obtener los ficheros históricos y mostrarlos posteriormente. Al igual que el configurador, en este último caso el visualizador actúa independientemente del programa medidor.

El motivo por el que se ha optado por esta arquitectura es que el configurador se utilizará esporádicamente y podrá ser empleado por un usuario, como puede ser un técnico de calidad, ajeno al funcionamiento del medidor. Esto mismo sucede para el caso del visualizador.

Tanto el configurador como el visualizador pueden ejecutarse en cualquier máquina de la red local, lo que hace posible que todos los programas sean usados por personas diferentes y en diferentes momentos.

Además de los tres programas comentados antes, la arquitectura software del sistema está conformada también por una serie de DLLs que se utilizarán para realizar tareas comunes a alguno de tres programas y por un programa independiente que gestiona la comunicación del medidor con elementos externos.

La Figura 3 muestra la relación existente entre los programas creados y las bibliotecas.

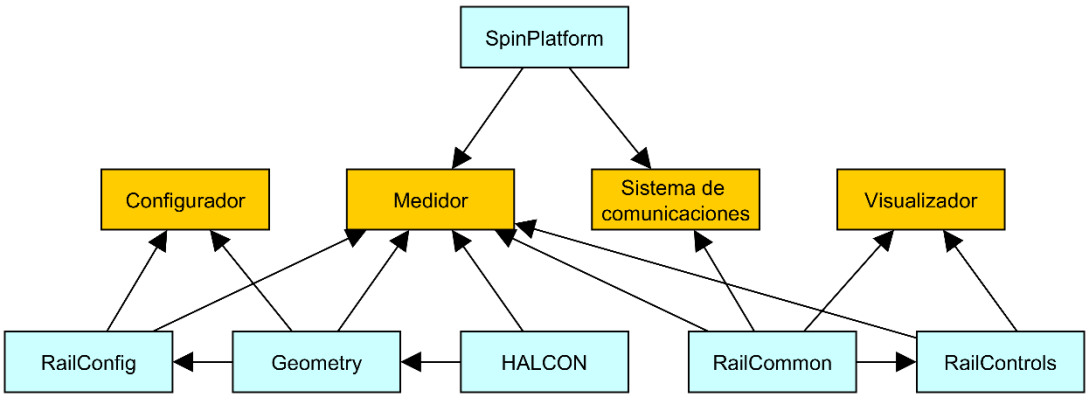


Figura 3: Relación entre programas y DLLs

En la Figura 3 se pueden observar diferentes tipos de componentes software. Dependiendo del color estos son:

- En amarillo, los programas que forman parte del proyecto.
- En azul, las bibliotecas de enlace dinámico o DLLs. Estas contienen funcionalidades comunes a varios programas del proyecto.

Tanto los programas como las bibliotecas pertenecientes al proyecto están implementados en el lenguaje de programación C#, utilizando la plataforma .NET.

4.1. Bibliotecas de enlace dinámico

4.1.1. Pertenecientes al proyecto

4.1.1.1. RailCommon

Esta biblioteca es la estructura básica del *modelo - vista - controlador* que se sigue para la visualización de los resultados. El *modelo* en esta biblioteca es la clase *Rail*, que contiene todos los resultados y los metadatos necesarios. Por otra parte está la clase *RailSource,* que actúa como controlador y que contiene una serie de métodos para exponer carriles a las diferentes vistas. Por último está la interfaz *RailView,* que actúa como *vista* y es la encargada de mostrar los resultados obtenidos, almacenados en el *modelo* y servidos por el *controlador*.

4.1.1.2. RailControls

Esta biblioteca incluye implementaciones de diferentes controles, incluyendo las vistas que implementa la interfaz *RailView*.

4.1.1.3. RailConfig

Esta DLL contiene toda la funcionalidad necesaria para generar y cargar el fichero de configuración de normas y carriles. Este fichero, generado por el programa configurador y empleado por el medidor, especifica para cada tipo de carril el contorno que debe tener idealmente (para luego compararlo con los carriles medidos) y los márgenes de tolerancia contenidos en las normas.

Esta biblioteca también es responsable de cargar y procesar los ficheros DXF que el usuario introduce en el programa configurador. Debe además almacenar, una vez leído el fichero de configuración, toda la información relativa a las normas y carriles que emplean tanto el medidor como el propio configurador.

4.1.1.4. Geometry

Esta biblioteca contiene todas las operaciones básicas de geometría que se realizan tanto en el configurador como en el medidor y en la biblioteca de configuración *RailConfig*. Esta biblioteca se utiliza para construir los perfiles mediante arcos y segmentos así como para calcular todas las dimensiones. Además, permite realizar transformaciones y todas las operaciones necesarias para llevar a cabo el proceso de medición.

La biblioteca se divide en las siguientes partes:

- *Core*: Parte central de la biblioteca, que contiene las clases de geometría básicas utilizadas en el resto de la biblioteca y en otros programas de la solución. Estas clases abarcan desde puntos o nubes de puntos hasta arcos y segmentos, así como diferentes operaciones geométricas aplicables a ellos.
- *Dimensions*: Esta parte de la biblioteca es la encargada de llevar a cabo el cálculo de las dimensiones del carril.
- *Math*: En esta parte se realizan todos los procesos que son puramente matemáticos. Estos procesos son, entre otros, la realización de aproximación de una nube de puntos a un modelo, la creación de matrices y operaciones sobre ellas, así como diferentes transformaciones geométricas.

- *Rail*: Esta es la parte encargada de crear las primitivas que componen un carril, es decir, arcos y segmentos, así como crear aproximaciones de nubes de puntos a estos segmentos e incluso crear una serie de estructuras de datos que ayuden al tratamiento de los carriles. Este tratamiento se realiza mediante una parte central llamada *RailModel*. El *RailModel* contiene toda la información relativa a los carriles, sus arcos y segmentos, así como una caché de datos sobre el propio carril, en el que se apoya el sistema de medición.
- *Registration*: Esta es la parte encargada de llevar a cabo el proceso de correspondencia entre puntos de las imágenes, los tomados mediante las cámaras y extraídos de estas, a puntos de los perfiles teóricos.

4.1.2. Preexistentes

4.1.2.1. HALCON

HALCON es una librería de visión artificial desarrollada por la empresa alemana MVTec, que incluye un entorno de desarrollo, denominado HDevelop. Además, proporciona interfaces de comunicación y drivers para numerosas cámaras de visión por computador disponibles en el mercado.

El programa medidor utiliza esta librería para calibrar el sistema de visión por computador, obtener imágenes y extraer los contornos de los carriles representados por la proyección de los patrones láser en las imágenes adquiridas. También la utiliza en menor medida la biblioteca de geometría para realizar determinadas transformaciones.

4.1.2.2. SpinPlatform

Esta parte es un conjunto de bibliotecas que conforman una plataforma desarrollada por el departamento Spin del Centro de Desarrollo Tecnológico de ArcelorMittal en Asturias.

La idea principal del uso de esta plataforma ha sido homogeneizar la estructura de todos los programas que se realicen para este departamento en concreto.

Esta plataforma contiene una serie de bibliotecas dinámicas para realizar tareas automatizadas. Cuenta con una serie de funcionalidades que abarcan desde la creación de hilos, creación y establecimiento de comunicaciones entre procesos y a través de la red, gestión y uso de sensores, etc.

Estas funcionalidades están divididas en las siguientes DLLs:

- **SpinPlatform_common:** Biblioteca principal en la que se incluye SpinDispatcher, la parte que gestiona todos los recursos e hilos de Spin, por lo que debe estar presente en todos los proyectos en los que se use esta plataforma. Aquí se incluye, además, el módulo de configuración, comunicaciones, errores y todo el conjunto de estructura de datos. También se incluye toda la gestión de hilos y comunicación entre procesos.
- **SpinPlatform_Controls:** En esta se incluyen controles genéricos implementados por Spin.
- **SpinPlatform_IO:** Aquí se encuentran los módulos de comunicación con bases de datos y de generación de consultas, un módulo para la creación de logs, así como un módulo para envío de emails y otro para comunicaciones FTP.

- **SpinPlatform_Sensors:** Integra los módulos que permiten comunicarse con sensores de instalaciones industriales.

En este proyecto se utiliza la DLL que contiene *SpinPlatform_common*, la parte central de esta plataforma, así como parte de la DLL de entrada y salida, *SpinPlatform_IO*.

4.2. Programas de la solución

4.2.1. Programa configurador

El cometido fundamental de este programa es preparar un archivo de configuración en XML, que el medidor usa después. Este fichero contiene toda la información necesaria sobre los carriles y las normas que se usan en el medidor.

Este programa se puede ejecutar de forma offline. Esto quiere decir que no se tiene por qué ejecutar a la vez que el programa medidor sino que, por el contrario, se puede utilizar para crear los archivos de una forma totalmente independiente del proceso de medición. Esto aporta una serie de ventajas, como son el crear archivos desde otro ordenador que no es el que ejecuta el programa medidor, o ser ejecutado por otras personas ajenas al proceso de medición.

Los archivos de configuración que este programa genera son complejos, puesto que contienen mucha información relativa a perfiles, normas y tolerancias, por lo que el uso de este programa es recomendable para la generación de los mismos. De todas formas se permite que un usuario avanzado pueda modificar este fichero de forma manual.

4.2.2. Medidor

El medidor constituye la parte fundamental del sistema construido en el proyecto. Implementa el subsistema de visión por computador y procesamiento de imágenes y se comunica con el resto de librerías del sistema para proporcionar las dimensiones del carril inspeccionado.

Para un funcionamiento correcto el medidor debe de estar calibrado. La calibración de un sistema de visión por computador basado en triangulación óptica mediante láser incluye parámetros intrínsecos (que permiten corregir las distorsiones ópticas del conjunto sensor más lentes) y parámetros extrínsecos (que permiten conocer la posición exacta en el espacio de cada uno de los puntos del patrón láser que observa la cámara).

Necesita conocer también los datos identificativos de los carriles que mide, proporcionados por el ordenador de proceso, y el momento en que un carril entra y sale de la zona de inspección (de la zona en la que el carril intersecta con el plano formado por los patrones lineales proyectados por los emisores láser).

También ha de tener acceso a los estándares y las definiciones de los tipos de carril que se van a medir. Esta información viene generada por el configurador.

En cuanto a los resultados, los publica de tres formas distintas: los almacena localmente, los envía a los visualizadores que se encuentren conectados al medidor y los envía al sistema de comunicaciones, para que los remita al ordenador de proceso.

4.2.3. Sistema de comunicaciones

Este sistema controla todas las comunicaciones existentes con el medidor y los agentes externos presentes, como son el *PLC* y el ordenador de proceso.

Este sistema se divide en tres grandes partes:

- La comunicación entre el medidor y el sistema de comunicaciones.

- La comunicación entre el ordenador de proceso y el sistema de comunicaciones.
- La comunicación entre el PLC y el sistema de comunicaciones.

El sistema de comunicaciones es, por tanto, el encargado de centralizar los mensajes que van desde los extremos y redirigirlos de forma adecuada al lugar de destino. Gracias a este sistema, el medidor se abstrae de las tareas de comunicaciones de red.

Para poder probar adecuadamente la comunicación con los agentes externos se ha implementado un emulador de PLC y un emulador de ordenador de proceso. Mediante estos dos emuladores se simula el comportamiento de los elementos reales que están presentes en el entorno de producción. Estos emuladores se utilizan durante la fase de desarrollo y experimentación en laboratorio y en la instalación industrial para detectar incidencias en la red o en el software de red los equipos externos que necesita el sistema.

4.2.4. Programa visualizador

Los archivos generados por el programa medidor contienen información histórica que puede resultar muy útil para varias aplicaciones en la instalación industrial, como puede ser la gestión de calidad o atención a reclamaciones de clientes sobre los carriles fabricados.

Por lo tanto, el programa visualizador puede llevar a cabo un análisis histórico de la producción de una forma totalmente independiente del proceso de medición, lo que implica que no se tiene que realizar este análisis en el mismo ordenador donde se están tomando los datos, sino que se pueden compartir, mediante una red local por ejemplo, los archivos que contengan la información histórica y realizar el análisis en cualquier otro lugar, como lo es por ejemplo el despacho de los ingenieros de calidad.

Por lo tanto, como se ha explicado anteriormente, con la finalidad de poder visualizar y analizar esta información desde cualquier PC, se incluye en el proyecto un programa visualizador más completo que el de tiempo real que ejecuta el medidor, y que es a su vez totalmente independiente de este.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

DOCUMENTO III

MEDIDOR

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Introducción

El programa medidor es el núcleo del sistema. Su cometido es captar imágenes procedentes de las cámaras, combinarlas para generar contornos, comparar los contornos con los perfiles generados por el programa configurador y medir el grado en que se ajustan a estos, para luego enviar los resultados al ordenador de proceso, almacenarlos localmente y servirlos en tiempo real al programa visualizador.

Como puede verse en la Figura 1, el medidor consume las salidas o produce las entradas de cada uno de los programas del sistema, y de él dependen, de un modo u otro, todos sus componentes.

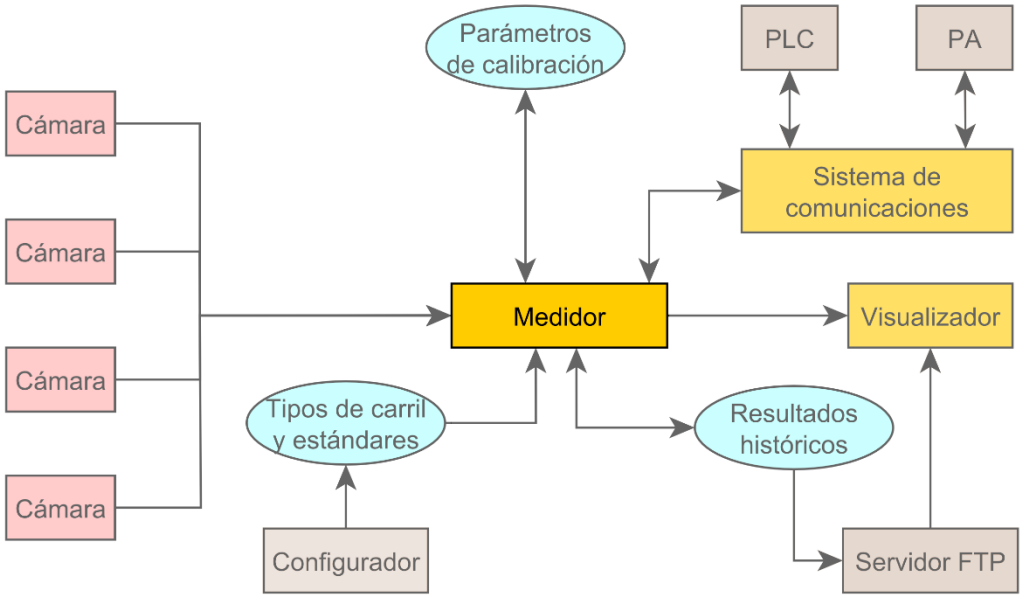


Figura 1: Diagrama del sistema, mostrando solamente los elementos relacionados con el medidor

El funcionamiento del medidor ha de ser en tiempo real, de modo que el sistema procese todas las imágenes adquiridas por las cámaras al ratio de adquisición configurado. Para ello, el medidor sigue una estructura *pipeline*, en la que cada fase del procesamiento de las imágenes y perfiles se lleva a cabo en un hilo distinto, desacoplando cada una de las fases mediante el paradigma productor/consumidor, y los hilos se comunican entre sí a través de colas compartidas y eventos.

2. Soporte hardware

El hardware de medición, cuya organización puede verse en la Figura 2, está formado por una estructura metálica hueca de forma ortoédrica. Esta estructura es atravesada en línea recta por los carriles que se han de medir.

En un plano perpendicular a la recta por la que pasan los carriles, llamado el plano láser, se encuentran cuatro emisores láser, dispuestos de tal forma que cubren todo el contorno de los carriles.

Formando un plano paralelo al de medición se encuentran cuatro cámaras que apuntan a la intersección entre el plano láser y la recta que recorren los carriles. Cada una de las cámaras capta por completo la parte del contorno del carril cubierta por uno de los punteros láser.

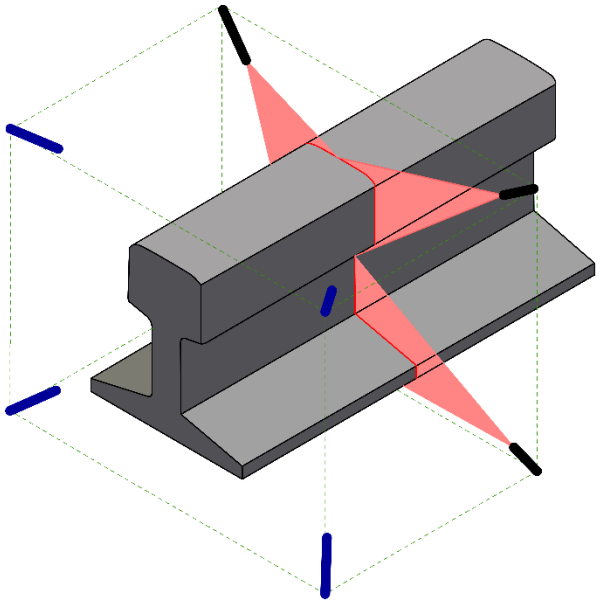


Figura 2: Esquema de la estructura empleada para la medición

2.1. Prototipo de laboratorio

Durante el desarrollo del sistema se construyó un prototipo de pruebas. En esta estructura se instalaron las mismas cámaras y emisores láser que en el sistema de producción.

Para las pruebas se emplearon segmentos de carril de entre 30 y 40 centímetros de longitud, que se colocaban sobre una superficie situada en perpendicular al plano láser, como puede verse en la Figura 3.

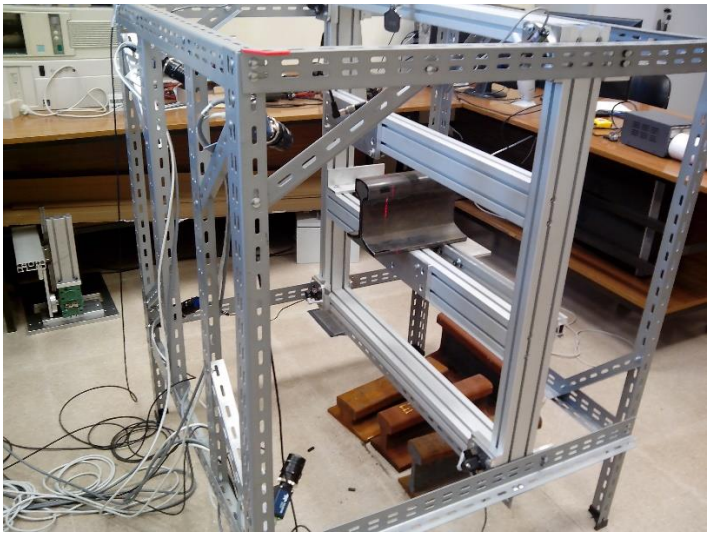


Figura 3: Prototipo de laboratorio, midiendo una sección de carril

Esta estructura también se utilizó para desarrollar y probar el procedimiento de calibración, para lo que se instaló la plantilla descrita en la sección 2.4 sobre un soporte vertical, como puede verse en la Figura 4.

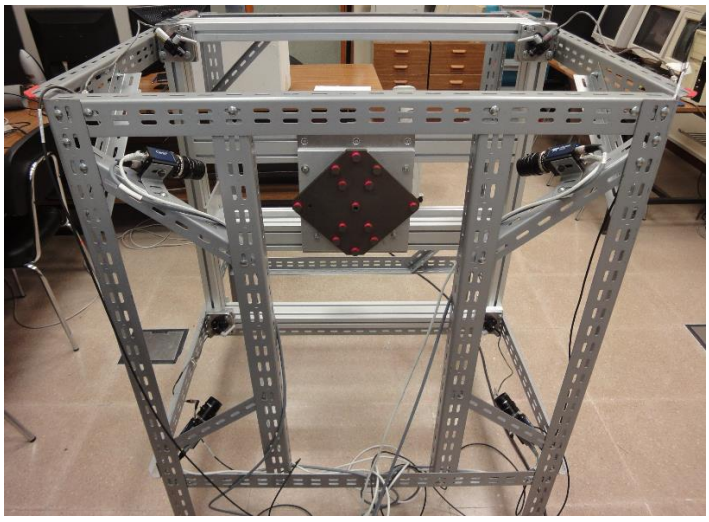


Figura 4: Prototipo de laboratorio, calibrando las cámaras

2.2. Sistema de producción

En producción, el equipo de medición se encuentra al final del tren de carril. Allí llegan los carriles recién fabricados, después de pasar otras inspecciones.

En la Figura 5 puede verse el sistema en producción en la instalación industrial.

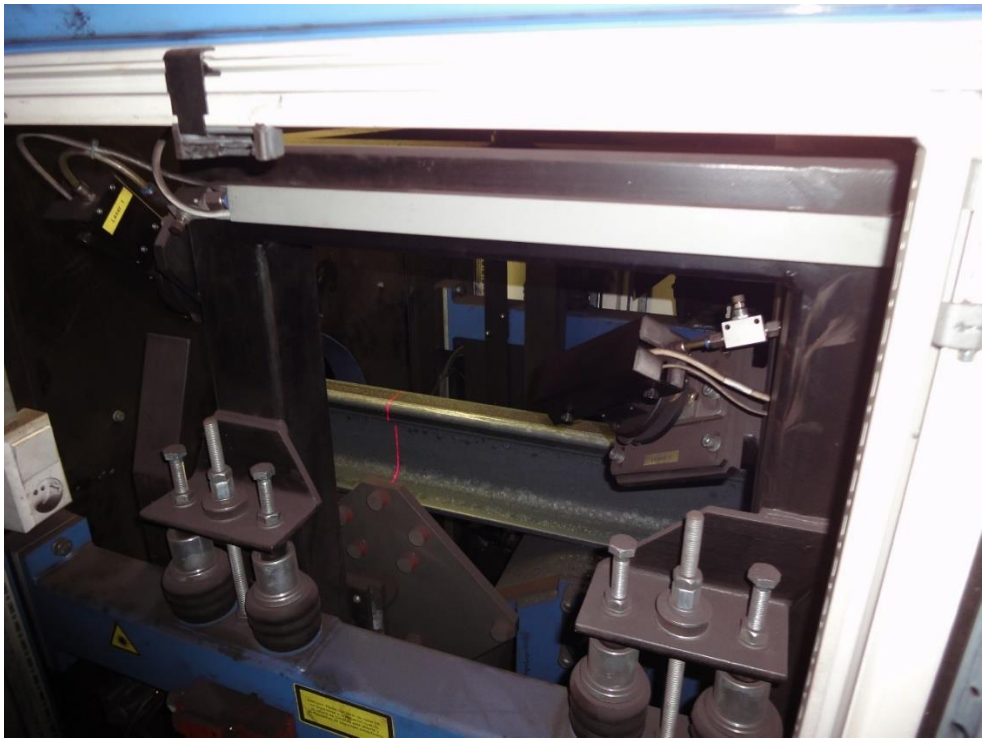


Figura 5: Sistema de producción

2.3. Cámaras

Se emplean cuatro cámaras DALSA Genie HM1400 (similares a la que puede verse en la Figura 6). Se trata de cámaras con sensor matricial CMOS, monocromo y con una resolución de 1400x1024 píxeles, que se conectan al ordenador de medición a través de una interfaz Ethernet utilizando el protocolo GigE Vision.

Figura 6: Cámara DALSA Genie HM1400

Hay que evitar que la misma cámara capte la luz procedente de distintos láseres para realizar un procesamiento óptimo. Para ello, se emplean láseres con distintas longitudes de onda, 660

nm y 695 nm (los láseres contiguos emiten en una longitud de onda diferente); y cada cámara cuenta con un filtro pasa-banda que deja pasar solamente la luz de longitudes de onda cercanas a las del láser correspondiente, de modo que los láseres contiguos al correspondiente a esa cámara no son visibles.

2.4. Plantilla de calibración

Para realizar la calibración del sistema, que se describe en la sección 7.6, el sistema de medición incluye una plantilla de calibración, que puede verse en la Figura 7.

Esta plantilla es una placa sobre la que se disponen varios cilindros en posiciones conocidas. Para calibrar, la plantilla se coloca en perpendicular a la recta que recorren los carriles y en paralelo al plano láser, de modo que los cilindros de la plantilla atraviesen este plano láser.

Cuando no se está calibrando, la plantilla de calibración se repliega de modo que los carriles puedan pasar a través del equipo de medición. Como se describe en la sección 3.1, el PLC es el encargado de desplegar y replegar la plantilla cuando el programa medidor lo requiere.

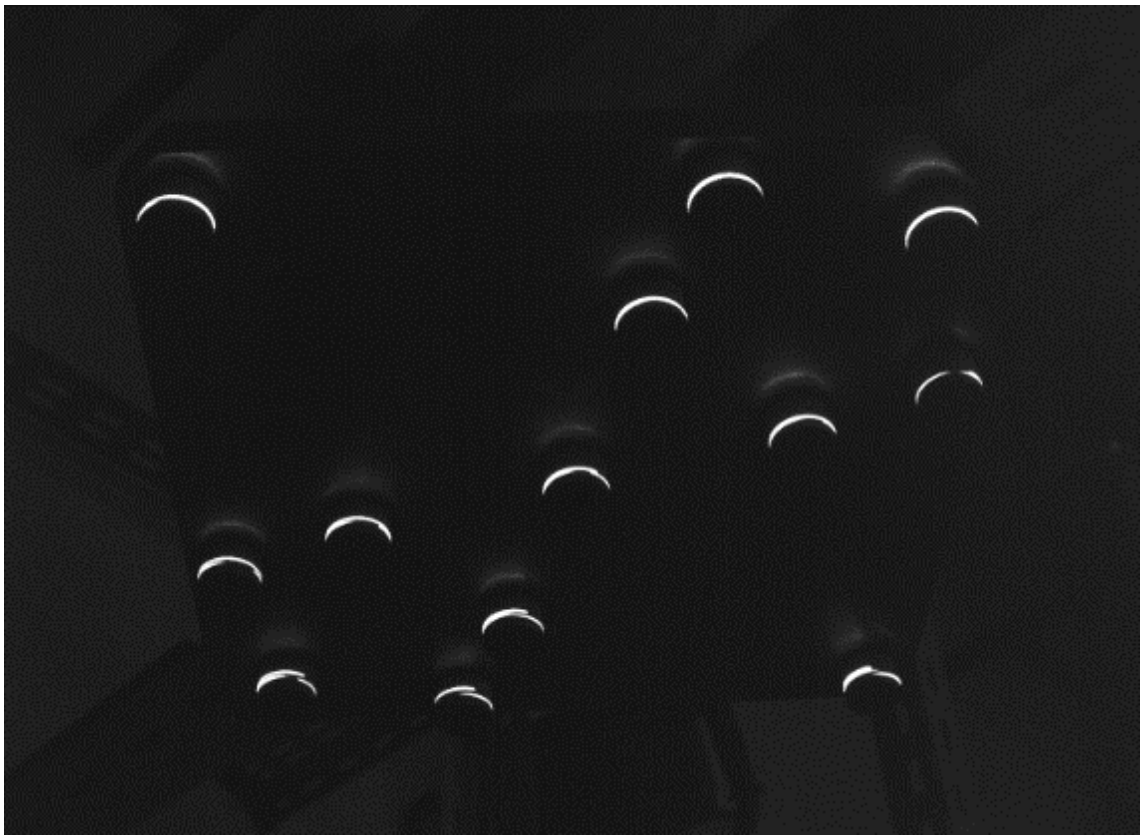


Figura 7: Líneas dibujadas por los láseres en la plantilla de calibración

3. Comunicaciones

Como se ha visto, para funcionar correctamente el medidor necesita comunicarse con elementos de control externos los cuales serán vistos en las secciones 3.1 y 3.2. Estos son necesarios para conocer determinados metadatos de los carriles entrantes y el momento en que comienzan y terminan, así como para controlar el movimiento de la plantilla de calibración.

Puesto que estos equipos pueden modificarse o sustituirse independientemente del programa medidor, surge la necesidad de aislarlo de ellos y encapsular las partes dependientes de estos equipos en un proceso independiente que pueda adaptarse a otros equipos externos distintos sin necesidad de modificar el propio programa medidor.

Este proceso secundario, denominado *RailCommunications*, se describe en el documento IV - Comunicaciones.

La comunicación entre el programa medidor y RailCommunications se realiza por medio de WCF (Windows Communication Foundation). El programa medidor expone un servicio WCF, con la interfaz que se describe en el documento IV - Comunicaciones, y *RailCommunications* se conecta a él.

Al comenzar la medición o la calibración, el programa medidor ejecuta automáticamente *RailCommunications* (este comportamiento puede desactivarse, por ejemplo, si se desea ejecutar *RailCommunications* desde una máquina distinta), que a su vez se conecta a los equipos externos y al programa medidor.

3.1. Encoder y PLC

El modo de medición está controlado por un dispositivo externo, un controlador lógico programable o *PLC*. Por otra parte, el disparo de las cámaras ordenado, de forma indirecta, por otro dispositivo externo llamado *encoder*; el tren de pulsos generado por el *encoder* en el avance del carril es retransmitido por el PLC a cada una de las cámaras.

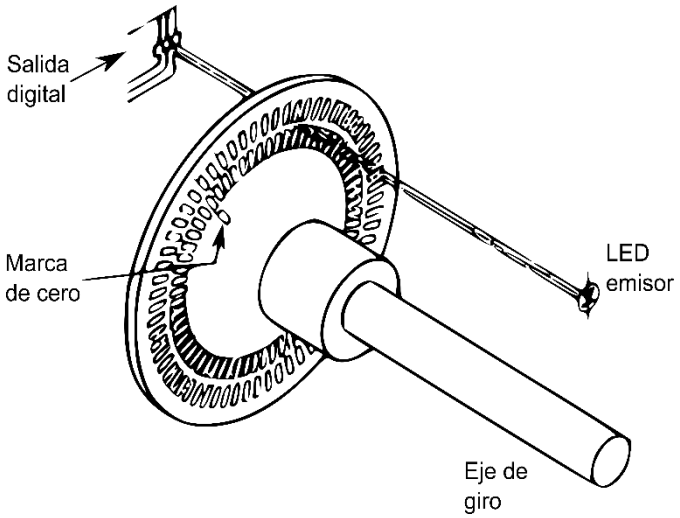


Figura 8: Esquema de un Encoder

Tal y como se puede ver en la Figura 8 el *encoder* es un dispositivo que cuenta con un disco que gira de forma solidaria a un eje. Este disco posee unas marcas por donde puede pasar un haz de luz emitido por un LED que pertenece al propio *encoder.* Cuando este haz de luz pasa por uno de las marcas, incide en un detector óptico, genera un pulso interno.

El eje del encoder se moverá en función del movimiento del carril que está pasando en ese momento por el sistema, haciendo que el disco gire un ángulo determinado, por lo que el haz de luz pasará por las marcas a un ratio dependiente de la velocidad de avance del carril.

Por último, el propio *encoder* generará un pulso cada una cierta cantidad de pulsos de luz que haya recibido, produciendo un flanco ascendente, como puede verse en la Figura 9, que se puede utilizar para cualquier fin, en este caso disparar la captura de imágenes en el sistema de medición.

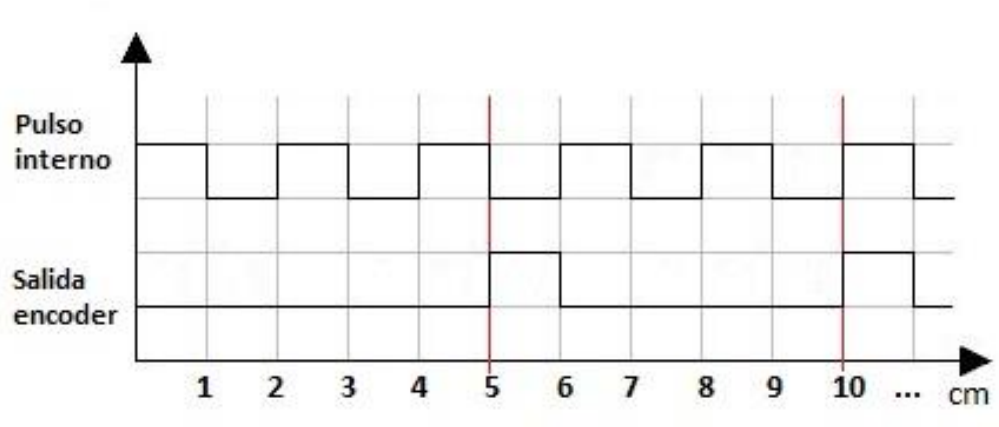


Figura 9: Salida del encoder

Cada vez que se genere el pulso ascendente, llegará al PLC. Este tendrá en cuenta el avance o el retroceso que ha sufrido el carril, y dependiendo de esto, transmitirá esta señal a todas las cámaras del sistema.

Hay que tener en cuenta que por razones de producción el carril también puede retroceder. Durante el retroceso el *encoder* continuará generando pulsos. Sin embargo, estos pulsos serán filtrados por el PLC, es decir, no se enviarán a las cámaras, haciendo que para el sistema sea totalmente transparente este escenario. Lo mismo pasa tras un desplazamiento *hacia atrás*, cuando el carril vuelve a avanzar hacia adelante, los pulsos generados deberán ser también filtrados por el PLC hasta que el carril llegue a la posición en la que comenzó a retroceder.

El funcionamiento del medidor en el modo de medición depende completamente del PLC, que actuará como maestro, siendo el medidor el esclavo. El PLC controla al medidor mediante el envío de mensajes de red.

Los mensajes que se pueden intercambiar entre el PLC y el programa de comunicaciones, *RailCommunications,* pueden ser en ambos sentidos, actuando este último como intermediario. Estos mensajes son los siguientes:

- Desde el PLC hacia el medidor pasando por *RailCommunications*:
    - El PLC envía el mensaje "*ENTER*" para indicarle al medidor que un nuevo carril ha entrado en la zona de medición y por lo tanto debe empezarse a medir.
    - El PLC envía el mensaje "*EXIT*" para indicarle al medidor que el carril acaba de abandonar la zona de medición y por lo tanto se debe dejar de medir.
- Desde el medidor hacia el PLC pasando por *RailCommunications*:
    - El medidor puede enviar el mensaje “Deploy calibration template” indicando al PLC que debe desplegar la plantilla de calibración. Esta acción será llevada a cabo cuando el PLC estime oportuno, es decir, cuando ningún carril este pasando en ese momento por la zona de medición, y siempre y cuando no se encuentre ya desplegada la plantilla de calibración.
    - El medidor puede enviar el mensaje “Retract calibration template” indicando al PLC que debe replegar la plantilla de calibración. Al igual que para el

mensaje anterior, el PLC la llevará a cabo cuando se estime oportuno, en este caso, cuando la plantilla de calibración este desplegada.

Hay que tener en cuenta que los dispositivos externos al sistema, es decir, el *encoder* y el *PLC*, son elementos totalmente independientes desde el punto de vista del medidor: por un lado las cámaras reciben pulsos eléctricos del encoder, captan imágenes y las envían al medidor, y por otro lado este recibe mensajes directamente del PLC.

3.2. Ordenador de proceso

El ordenador de proceso es el encargado de controlar la producción del tren de carriles.

En relación al programa medidor, el ordenador de proceso puede enviar o recibir los siguientes mensajes:

- Indicar al medidor el tipo y norma del siguiente carril a analizar por el mismo. Esto se lleva a cabo mediante el mensaje enviado del ordenador de proceso al medidor llamado *NewRailData*.
- Solicitar al medidor un análisis de un carril que ya haya sido procesado. Mediante el mensaje del ordenador de proceso hacia el medidor llamado *RequestOneRailMeasurementResults.*
- Recibir del medidor el análisis del último carril medido. Esto se lleva a cabo mediante el mensaje enviado del medidor al ordenador de proceso llamado *LastRailMeasurementResults*

Para llevar a cabo estas operaciones, el ordenador de proceso se comunicará con el programa de comunicaciones *RailCommunications*, que actuará de intermediario, de la misma forma que en el caso del PLC.

El ordenador de proceso tendrá, por lo tanto, todos los análisis de los carriles medidos por el medidor durante su ejecución, para que, posteriormente, la persona encargada de tratar la información pueda hacerlo de forma sencilla.

4. Interfaz gráfica

La interfaz gráfica del medidor está formada por un formulario principal, que se muestra a pantalla completa, y varias ventanas secundarias empleadas para consultar y configurar parámetros, etc.

4.1. Formulario principal

El formulario principal, que puede verse en la Figura 10, ocupa la pantalla completa por dos motivos. El primero es que la complejidad de la interfaz hace aconsejable tratar de aprovechar todo el espacio disponible. El segundo es que de este modo se impide que el usuario lo redimensione, lo que durante la medición podría suponer una ralentización considerable.

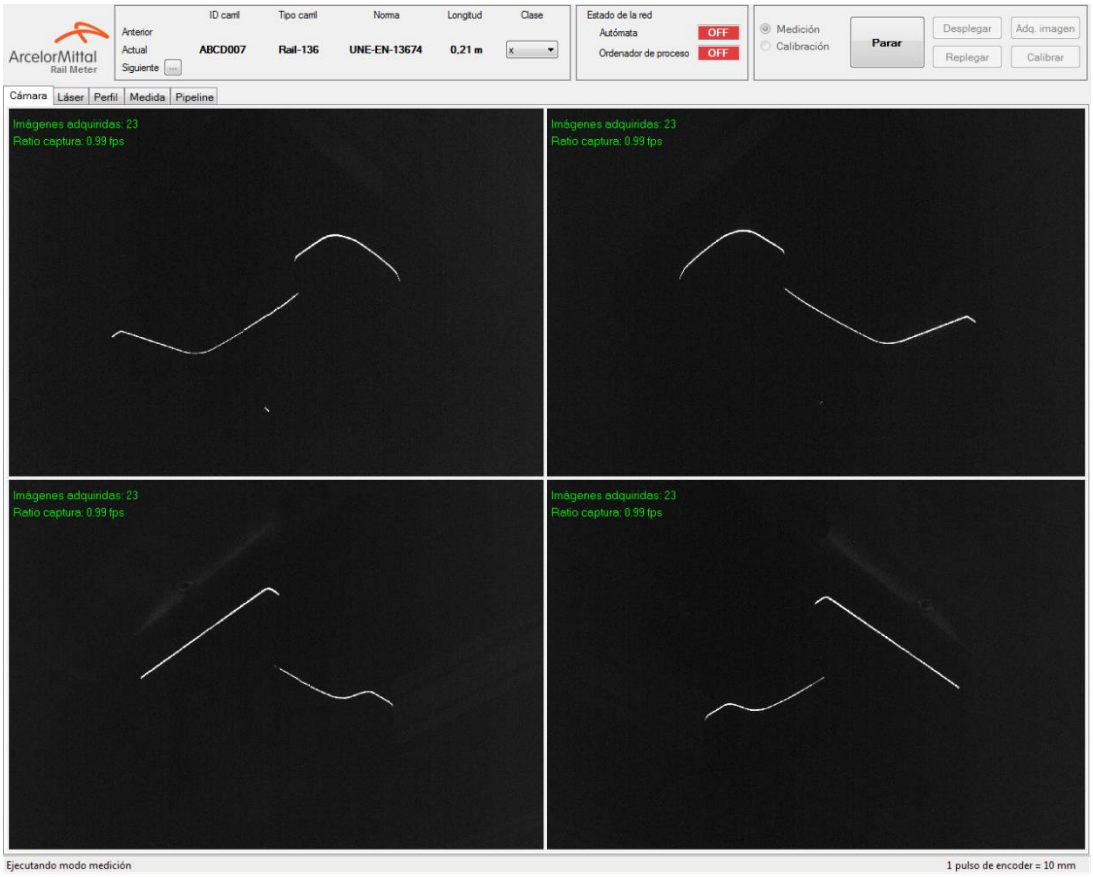


Figura 10: Interfaz gráfica del medidor

Este formulario principal cuenta con una serie de pestañas que muestran resultados de distintas fases de la medición, y también con una barra en la zona superior que incluye la mayor parte de los controles de uso habitual y proporciona información sobre el estado del sistema.

Los elementos que forman esta barra superior son los siguientes, de izquierda a derecha:

- Indicador de carriles: Incluye el identificador y el tipo, así como la norma empleada, del carril que se está midiendo en un momento determinado (*Actual*), el último carril que se midió (*Anterior*) y el siguiente carril que se medirá (*Siguiente*). Del actual y el anterior muestra también la longitud medida, y del actual permite además seleccionar

la clase de tolerancia que se muestra en la pestaña Medida. Este ajuste no tiene ningún efecto en los resultados que se almacenan ni en lo que se muestra en el programa visualizador si está conectado. Un botón permite fijar los parámetros (es decir, identificador, tipo y norma) del carril siguiente, como se especifica en la sección 4.2; si bien típicamente se establecen de forma automática con los datos recibidos del ordenador de proceso (sección 3.2).

- Indicador de estado de la red: Muestra el estado de la conexión (conectado, *ON*, o desconectado, *OFF*) con el autómata (PLC) y el ordenador de proceso (PA).
- Panel de control: Permite seleccionar un modo, entre el de medición y el de calibración. En el modo de medición, un botón permite arrancar el proceso de medición o pararlo, cambiando el texto del mismo (Arrancar/Parar) según el caso. En el modo de calibración, existe el mismo botón de arrancar/parar, además de otros dos para solicitar al PLC que despliegue o repliegue la plantilla, uno para refrescar las imágenes (Adq. imagen) que se muestra y otro para tratar de calibrar con las últimas imágenes tomadas.

En cuanto a las pestañas de resultados, son las siguientes:

- Pestaña Cámara: Muestra la última imagen capturada por cada cámara. Para cada cámara señala además el número de imágenes capturadas en total y por segundo. En la Figura 10 puede verse el contenido de la pestaña en el modo de medición, y en la Figura 11 en el modo de calibración.

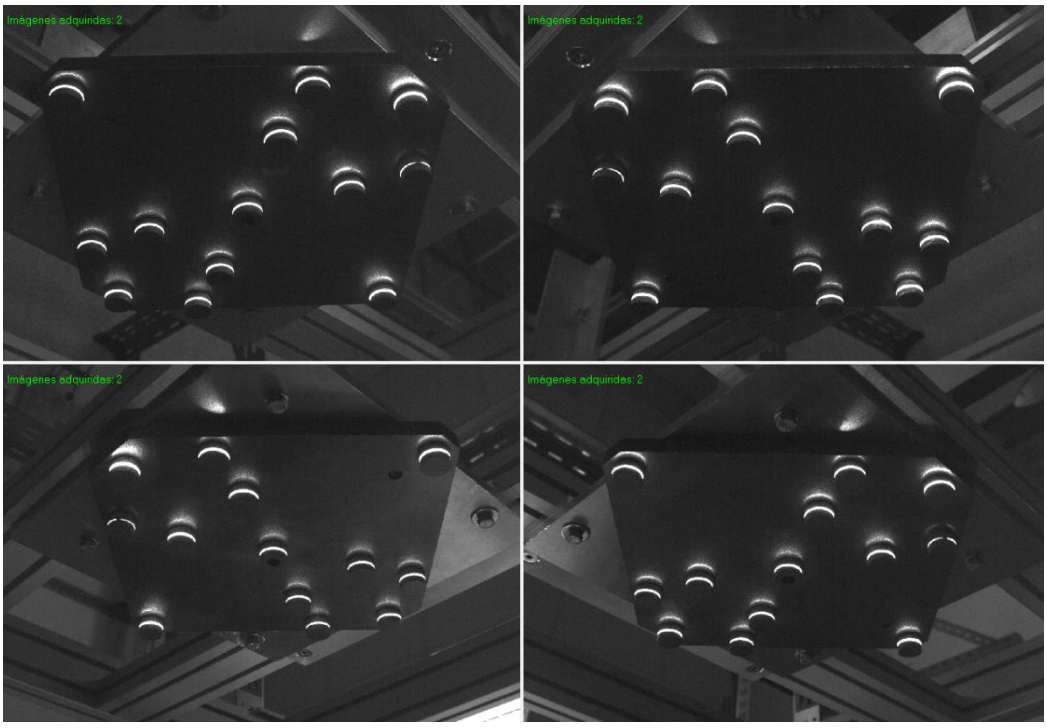


Figura 11: Pestaña Cámara durante la calibración

- Pestaña Láser: Muestra las líneas láser extraídas de la última imagen capturada por cada cámara. Además, muestra para cada cámara el número de segmentos extraídos,

el tiempo que se tardó en extraerlos, y el número de imágenes procesadas. Esta pestaña puede verse en la Figura 12.

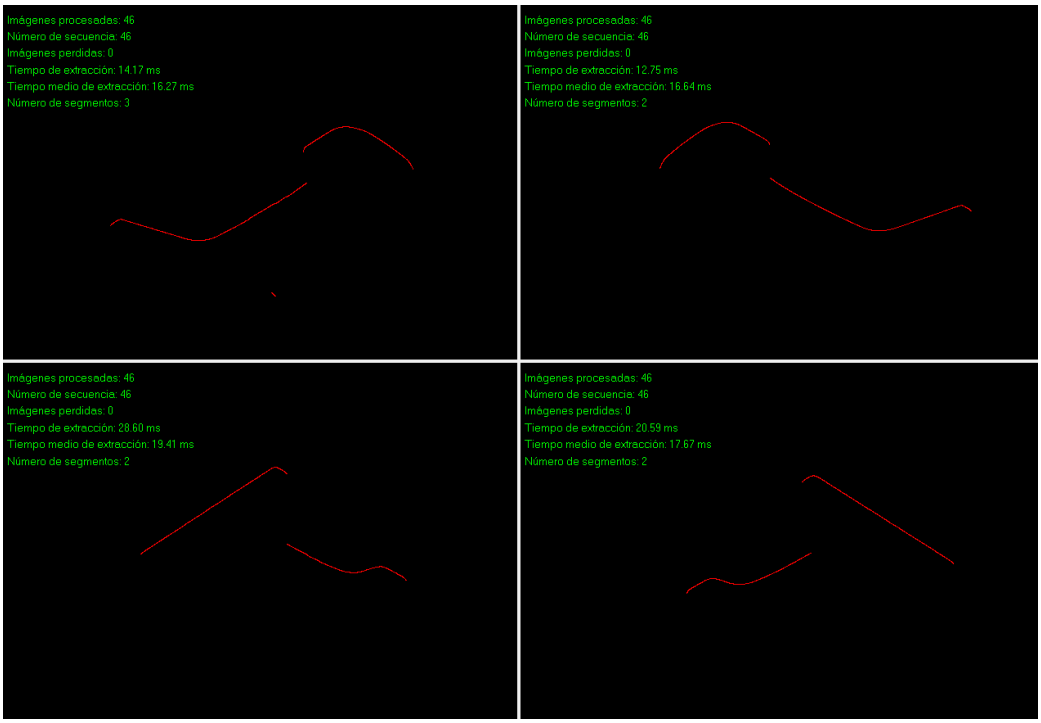


Figura 12: Pestaña Láser durante la medición

- Pestaña Perfil: Muestra las líneas procedentes de cada cámara, traducidas a coordenadas del mundo y combinadas en una única imagen. Indica también el número de carriles completos procesados, la longitud total de la parte del carril que se ha medido, etc. Para el modo Medición puede verse en la Figura 13. Para el modo Calibración, en la Figura 14.

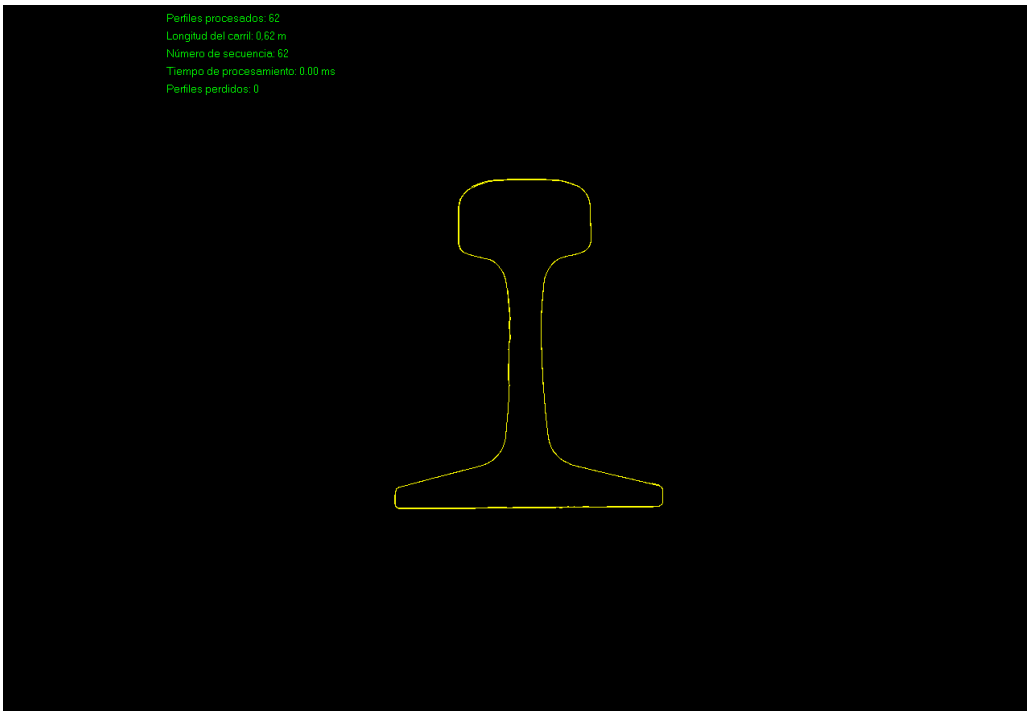


Figura 13: Pestaña Perfil durante la medición

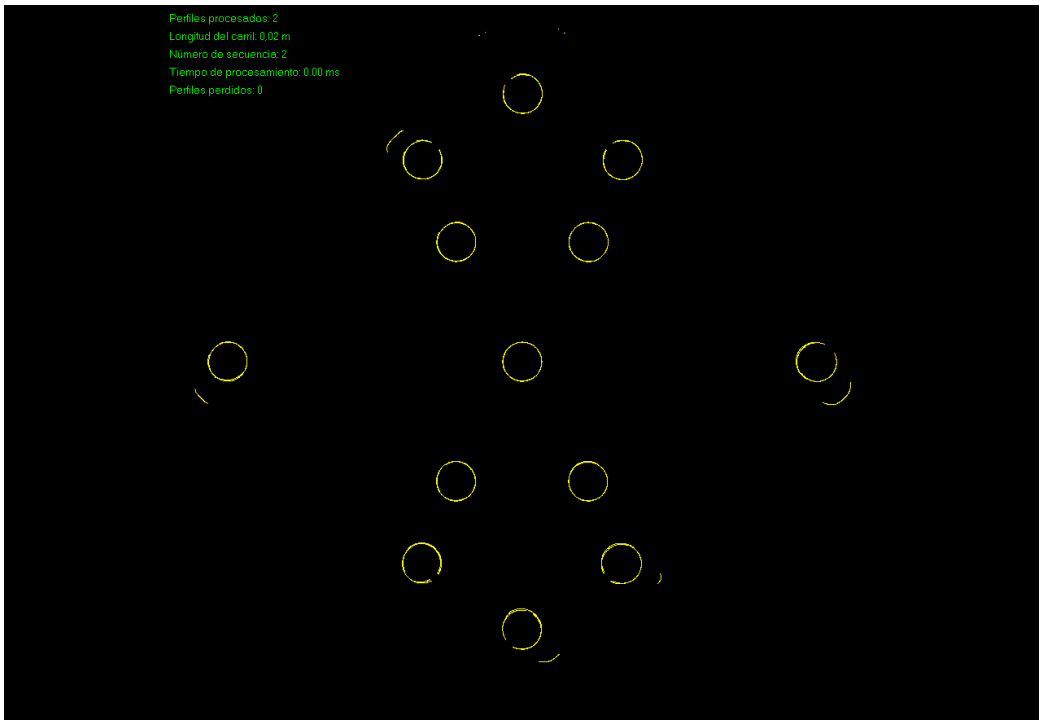


Figura 14: Pestaña Perfil durante la calibración

- Pestaña Medida: Muestra el progreso de la medición y los valores de las dimensiones en una posición concreta. Emplea la funcionalidad de visualización. Puede verse en la Figura 15.

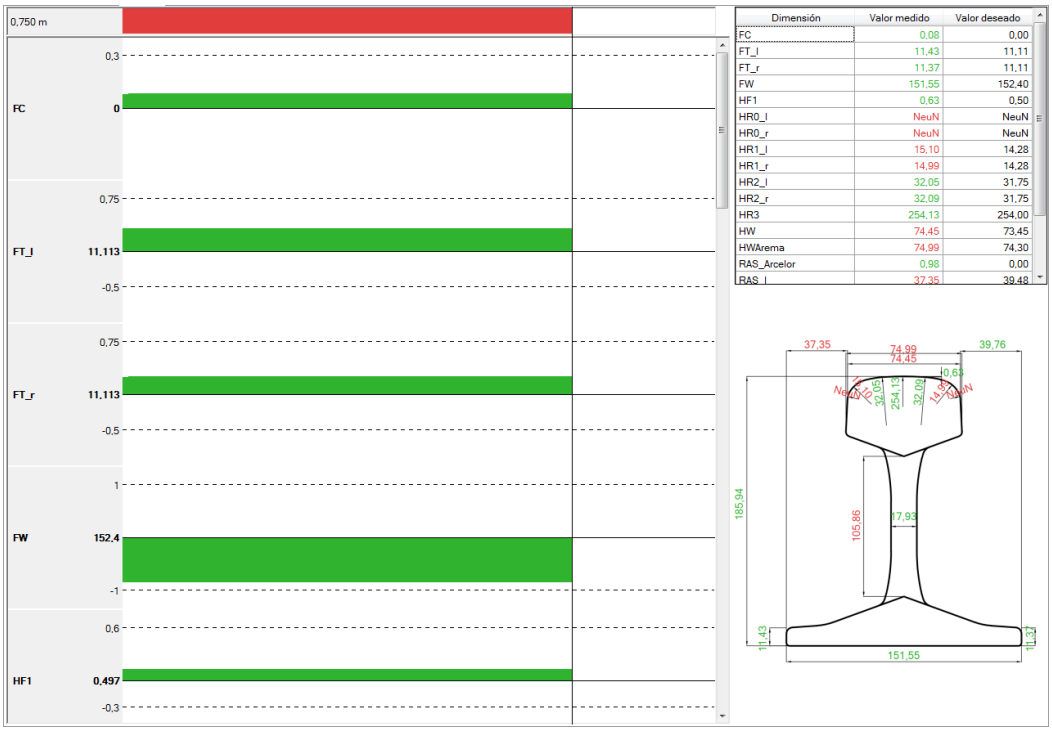


Figura 15: Pestaña Medida durante la medición

- Pestaña Pipeline: Muestra el número de elementos que se encuentran en las colas del pipeline así como el tiempo invertido en visualizar las dimensiones (pestaña Medida). Esta pestaña puede verse en la Figura 16.

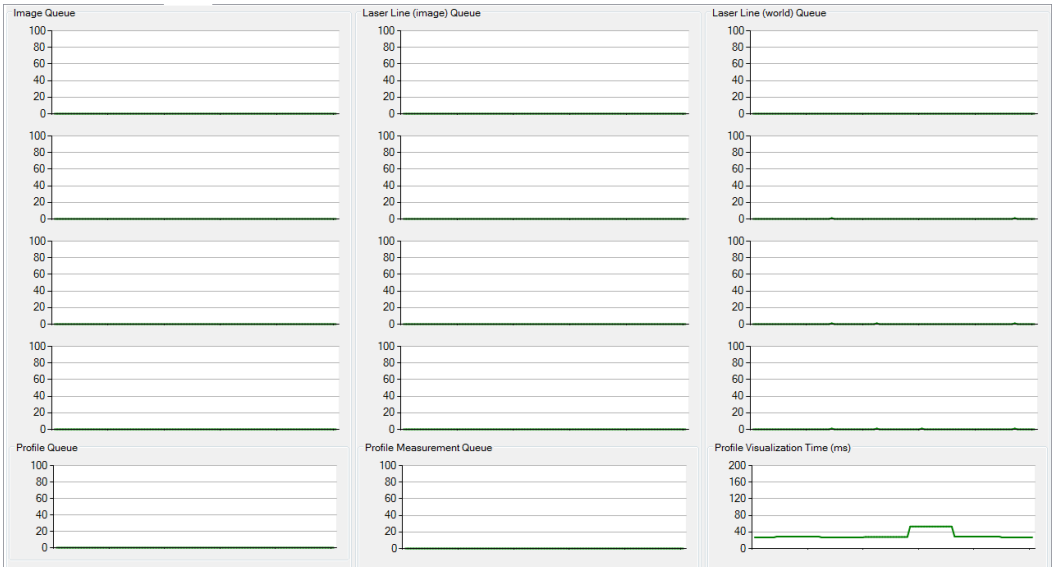


Figura 16: Pestaña Pipeline durante la medición

4.2. Ventana de entrada manual de carriles

Esta ventana, accesible solamente en el modo de medición, pulsando el botón que aparece en la línea que muestra el carril *Siguiente* en la cabecera, permite especificar los metadatos del siguiente carril que se medirá.

Normalmente no es necesario especificar a mano estos parámetros, sino que se rellenan automáticamente con información recibida del ordenador de proceso. Solamente en casos excepcionales, principalmente cuando el ordenador de proceso no está disponible, tiene sentido emplear esta funcionalidad.

La información que se puede especificar es la siguiente:

- ID del carril: Identificador del siguiente carril. No se comprueba si su formato se ajusta a uno preestablecido ni si es único, si bien la existencia de múltiples carriles con el mismo identificador puede provocar resultados anómalos.
- Incremento ID: Cantidad en que se incrementa el identificador para generar el identificador del carril siguiente cuando este no se especifica por parte del ordenador de proceso ni manualmente. El modo en que se genera el identificador incrementado es el siguiente: el último grupo de uno o más dígitos contiguos que aparezca en el identificador anterior se interpreta como un número, y el nuevo identificador es el texto que precede al grupo de dígitos, seguido del número incrementado en la cantidad especificada, seguido del texto que sigue al grupo de dígitos. Si no aparece ningún grupo de dígitos, el identificador resultante es el que se generaría, del modo que se especificó anteriormente, para un identificador formado por el identificador anterior seguido del dígito 0.
- Tipo de carril: Tipo del siguiente carril. Ha de elegirse de entre los tipos de carril admitidos por el medidor. Por defecto es el mismo que para el carril anterior.
- Norma del carril: Norma que ha de seguir el siguiente carril. Ha de elegirse de entre las normas conocidas por el medidor. Por defecto es la misma que para el carril anterior.

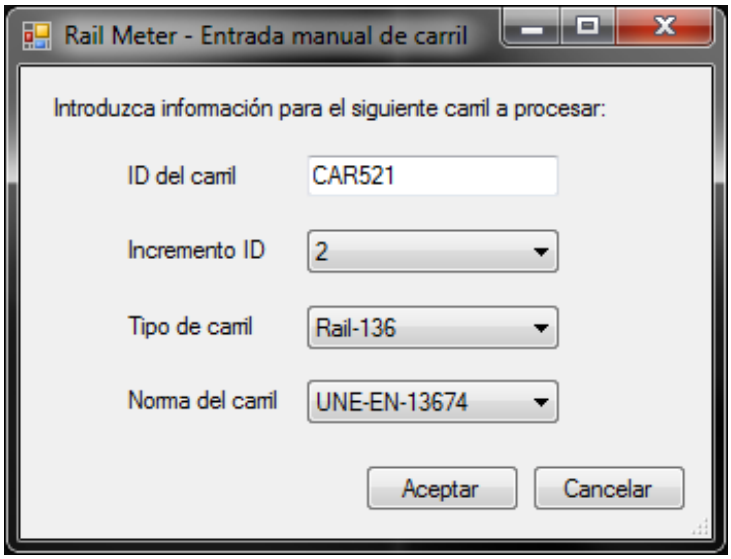


Figura 17: Ventana de entrada manual de carriles

4.3. Ventana de resultados de calibración

Una vez realizada la calibración, se muestra esta ventana (visible en la Figura 18), que incluye un informe de los resultados y la calidad de los parámetros calculados.

Este informe incluye una estimación del error medio y el error máximo (máximo de los errores de todos los cilindros y todas las cámaras). Además, para cada cámara especifica su tipo, el número de cilindros encontrados en la imagen captada, el número de correspondencias entre cilindros y puntos de la imagen que se seleccionaron finalmente, el error medio, la desviación típica y el error máximo.

Con esta información, el usuario ha de elegir si desea utilizar los parámetros de calibración computados o desea descartarlos (y capturar una nueva imagen para repetir la calibración o bien emplear los parámetros anteriores).

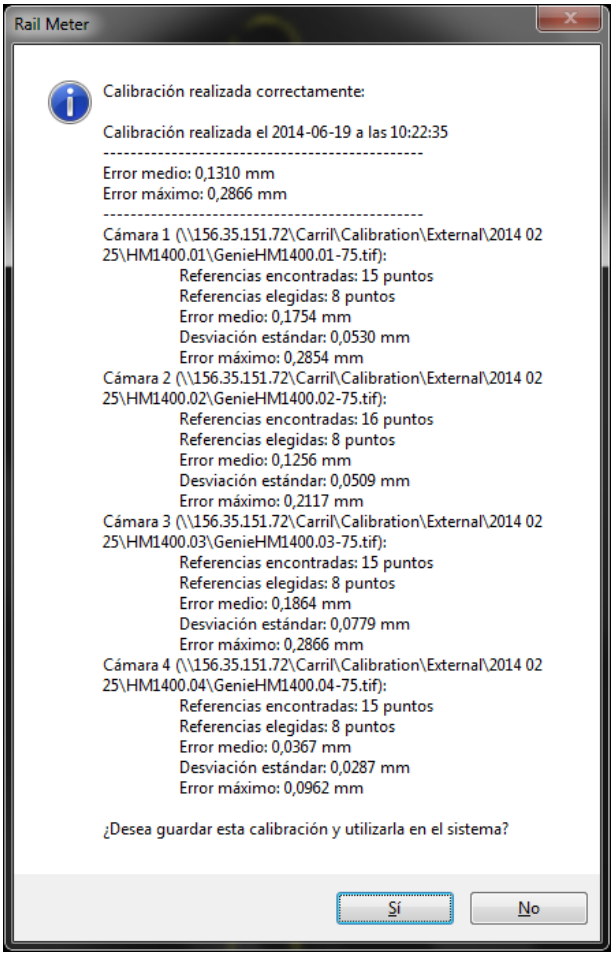


Figura 18: Ventana de resultados de calibración

4.4. Ventana de configuración

La ventana de configuración permite modificar determinados parámetros del sistema medidor, que afectan a la forma en que se calibra y mide, los puertos en los que escucha, etc.

La lista completa de parámetros modificables es la que aparece en la sección 6.1.

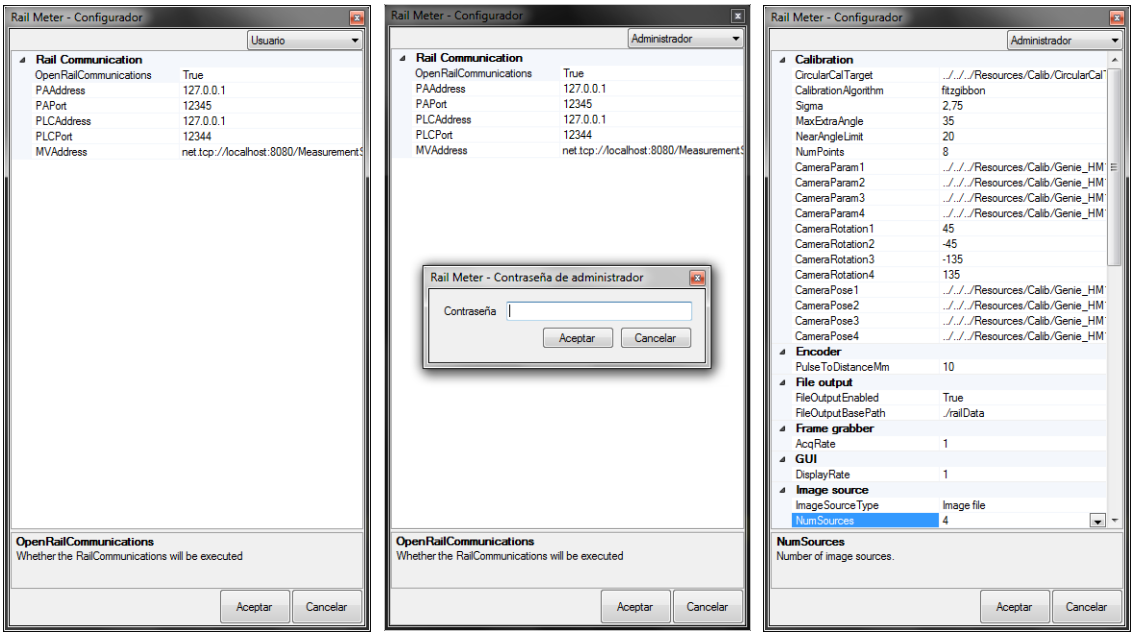


Figura 19: Vistas posibles de la ventana de configuración

Esta ventana tiene dos vistas posibles: la de Usuario y la de Administrador. Como puede verse en la Figura 19, la vista de Usuario muestra un número limitado de opciones. La vista de Administrador muestra todas las opciones disponibles, pero está protegida por contraseña.

Si se ha modificado algún valor de la configuración y se ha pulsado el botón “Cancelar” se muestra un mensaje indicando que la configuración ha sido cambiada, como se puede ver en la Figura 20.

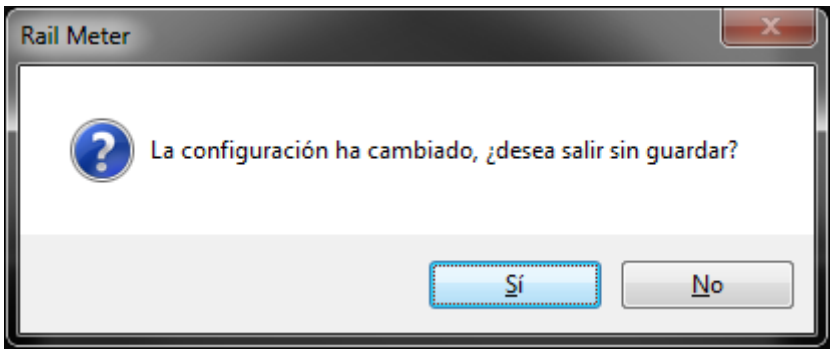


Figura 20: Salir sin guardar con la configuración cambiada

Al pulsar el botón aceptar la ventana se cierra, y si se han realizado cambiados en la misma, estos serán guardados de forma automática en la configuración del sistema.

4.5. Ventana de generación de eventos

Con la finalidad de poder generar eventos de forma independiente tanto del ordenador de proceso como del PLC se ha implementado una ventana que permite al usuario llevar a cabo las acciones necesarias para iniciar la medición del carril simulando el comportamiento del hardware externo.

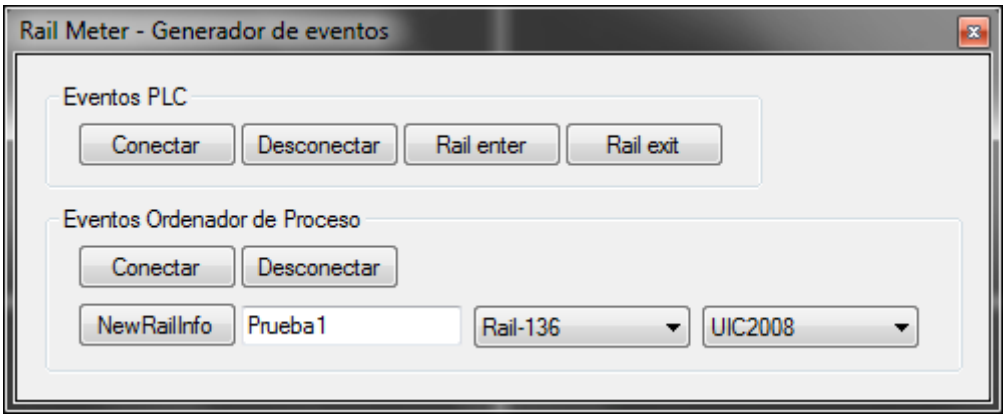


Figura 21: Ventana de generación de eventos

La Figura 21 muestra la ventana mediante la que el usuario puede enviar los siguientes mensajes al medidor:

- Simular la conexión o desconexión del PLC, haciendo así que el mismo pase al estado conectado o desconectado, respectivamente.
- Indicar el inicio o final de un carril, haciendo que el medidor inicie o finaliza la medición de un carril.
- Simular la conexión o desconexión del ordenador de proceso. Al igual que para el caso del PLC, actualizará el estado del ordenador de proceso haciendo que pase al estado conectado o desconectado respectivamente.
- Enviar nueva información de carril, la cual será usada para identificar el siguiente carril que se vaya a medir. Esto se consigue mediante la inserción de un nuevo identificador en el campo de texto al lado del botón *NewRailInfo*, y seleccionando el tipo de carril y la norma con la que se va a medir en los comboboxes destinados para estos fines, a la derecha del cuadro de texto anterior.

La Figura 22 muestra las opciones disponibles para la selección del tipo de carril mediante el uso del combobox destinado para este fin.

Estas opciones habrán sido extraídas del fichero de configuración que se ha generado mediante el uso del configurador. Este fichero es cargado por el medidor y muestra todos los carriles en este combobox, así como todas las normas en el combobox análogo.

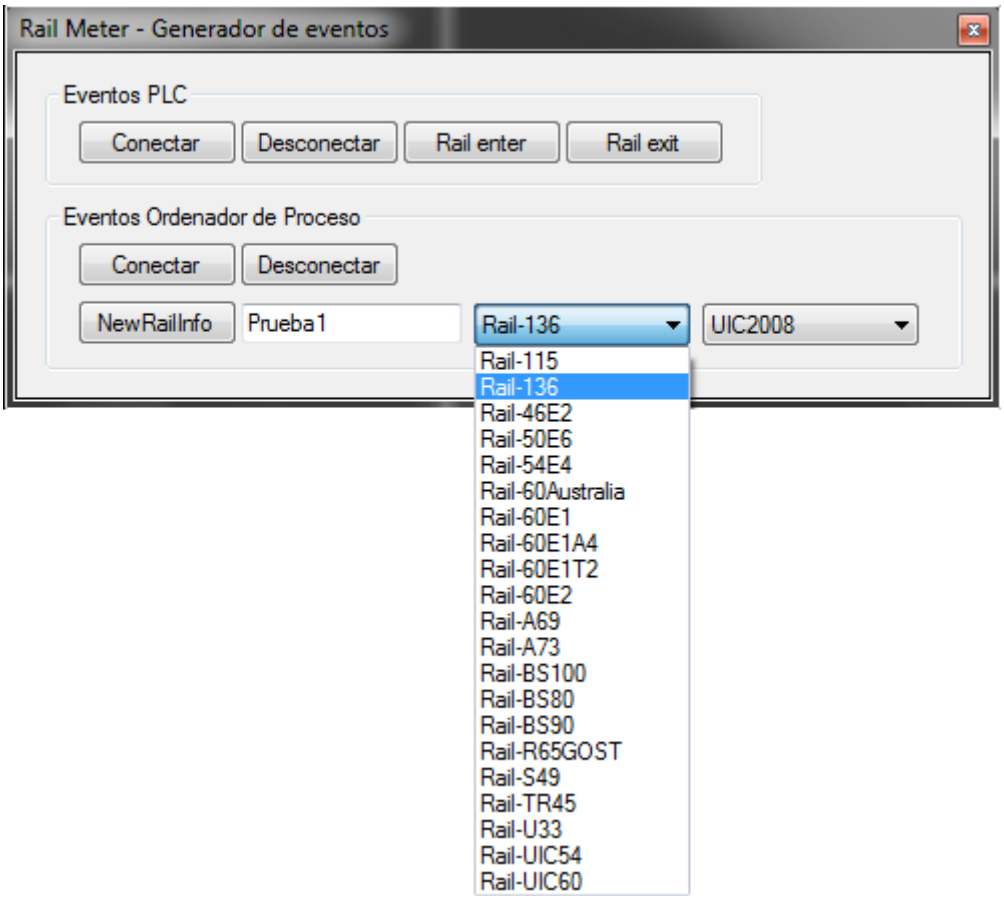


Figura 22: Combobox de selección de tipo de carril desplegado en la ventana de generación de eventos

4.6. Ventana de configuración del *encoder*

La ventana de la configuración del *encoder* que se puede ver en la Figura 23 permite al usuario configurar la distancia que ha recorrido el carril entre cada pulso generado por el *encoder*.

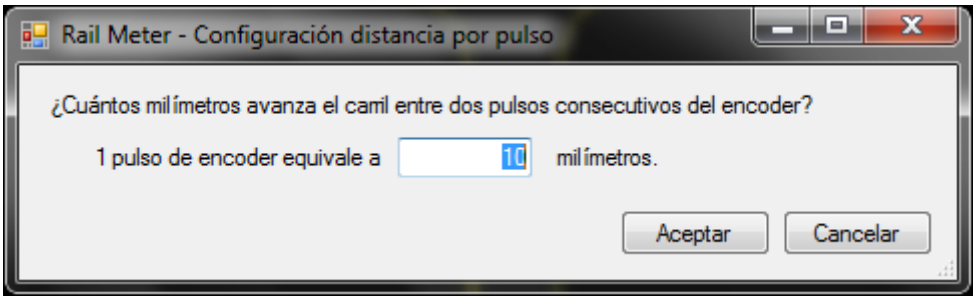


Figura 23: Ventana de configuración del encoder

Esta variable puede ser modificada también mediante el uso de la ventana de configuración vista en la sección *4.4*, en el parámetro de configuración PulseToDistanceMm.

Para modificar el valor de esta variable se debe modificar el valor del campo de texto de la ventana y pulsar el botón aceptar, esto mostrará un mensaje de advertencia, mostrado en la Figura 24.

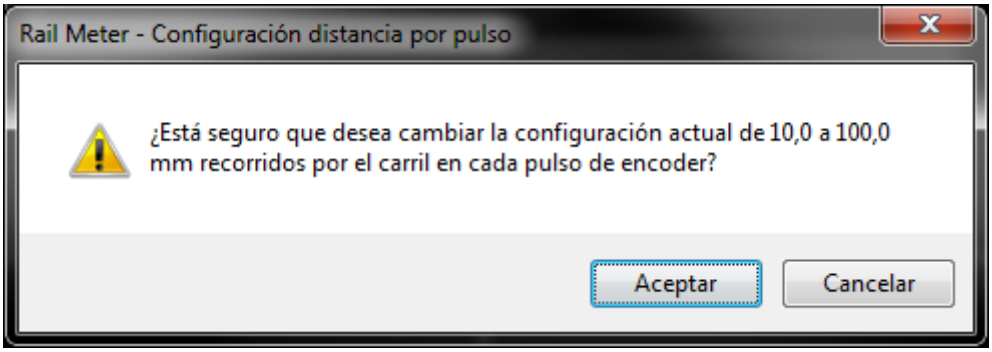


Figura 24: Confirmación del cambio de la configuración del encoder

En caso de cancelar este mensaje, los cambios no surtirán efectos. Si, por el contrario, se pulsa el botón aceptar, la configuración del encoder será grabada y todos los carriles que se midan a partir de entonces se hará con la nueva configuración.

Esta ventana de configuración podrá usarse solamente cuando el medidor esté en el modo de medición y no se esté midiendo ningún carril. En caso de que esto se realice en cualquier otro momento se mostrará un aviso como el que se observa en la Figura 25.

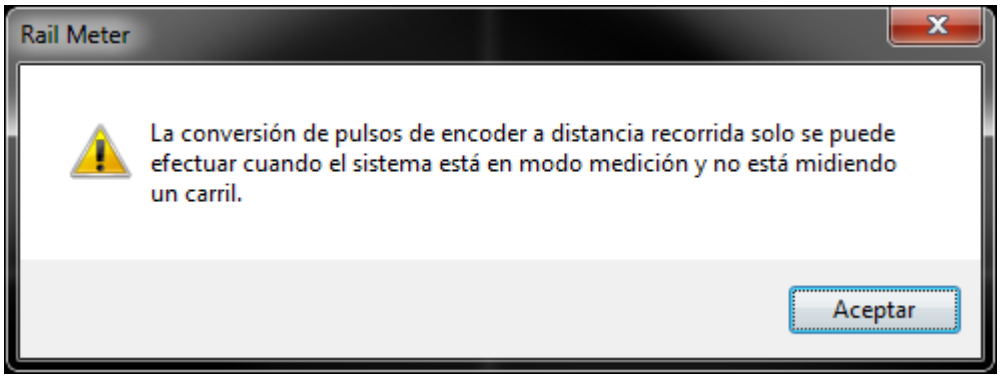


Figura 25: Aviso de cambio de configuración del encoder en un momento no permitido

4.7. Ventana de cambio de contraseña de administración

La ventana de cambio de contraseña de administración deberá ser usada cada vez que se desee cambiar la contraseña utilizada para acceder a la configuración general mediante la vista de administrador, como se ha explicado en la sección 4.4. Este cambio de contraseña se puede llevar a cabo mediante una ventana destinada para tal fin como la mostrada en la Figura 26.

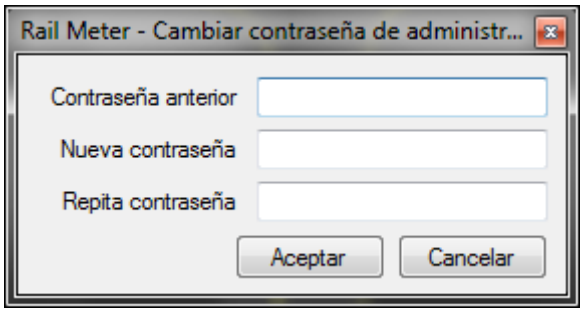


Figura 26: Ventana de cambio de contraseña de administrador

Para realizar el cambio de contraseña se debe introducir la contraseña anterior del sistema en el primer campo de texto, y después introducir en los dos siguientes la nueva contraseña. Una vez realizado esto se debe pulsar el botón aceptar y el cambio de contraseña se grabará en el fichero destinado especialmente para tal fin, como puede verse en la sección 6.2.

Se pueden presentar tres escenarios en caso de querer realizar un cambio de contraseña, estos se ven reflejados en los mensajes de la Figura 29:

- En caso de que las nuevas contraseñas no coincidan se mostrara el mensaje “Las contraseñas no coinciden” no permitiendo al usuario cambiarlas.
- En caso de la contraseña de administración previa no haya sido introducida de forma adecuada, es decir, no sea la misma contraseña, se mostrara el mensaje “Contraseña incorrecta”, no permitiendo al usuario cambia la contraseña.
- Por último, en caso de haber realizado de forma correcta el cambio de contraseña, se mostrará el mensaje “La contraseña ha sido cambiada”, momento en el cual se grabará la configuración al fichero adecuado, y pudiendo utilizar esta contraseña en el futuro.

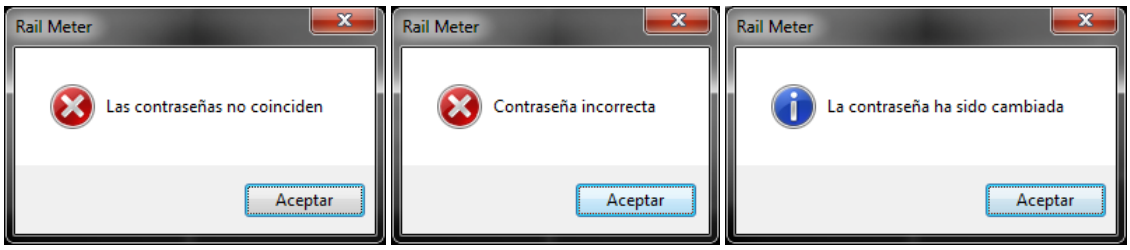


Figura 27: Cambio de contraseña

Por otra parte, si la contraseña no se ha cambiado nunca, o el fichero de la contraseña del medidor ha sido eliminado, se mostrará un mensaje como el que se puede ver en la Figura 28 a la hora de intentar acceder a la vista de administrador de la ventana de configuración.

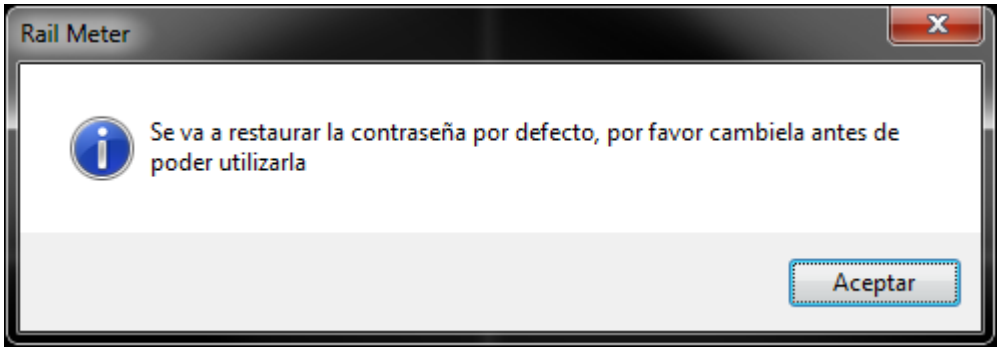


Figura 28: Aviso de restauración de la contraseña por defecto

Para poder restablecer la contraseña de forma adecuada, se debe introducir en la ventana de la Figura 26, en el campo contraseña anterior, la contraseña por defecto. Esta contraseña por defecto no se podrá cambiar ya que está especificada en el código del propio medidor.

Para poder utilizar la contraseña de administración deberá ser modificada previamente, la Figura 29 muestra el mensaje de error que se muestra en caso de que la contraseña no haya sido modificada antes de intentar acceder a la vista de administrador en la ventana de configuración vista en la Figura 19.

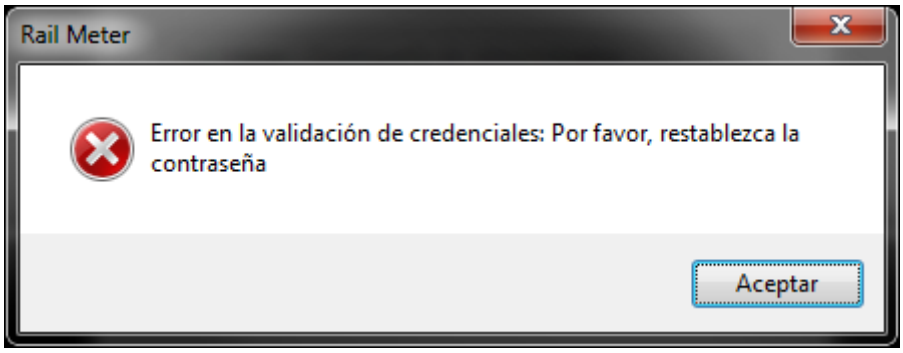


Figura 29: Error de la validación de credenciales

5. Modos de funcionamiento

5.1. Modo de medición

Este modo se utiliza para llevar a cabo la medición de los carriles en el sistema. Este modo se inicia cuando entra un carril al sistema de medición y debe ser procesado para que el usuario del sistema compruebe la calidad de fabricación del mismo.

El diagrama de secuencia que se puede observar en la Figura 30 muestra la secuenciación de mensajes que se ha de producir para el inicio de la medición de carriles, desde que el operador inicia el programa hasta que arranca el modo de medición y el sistema comienza a procesar carriles.

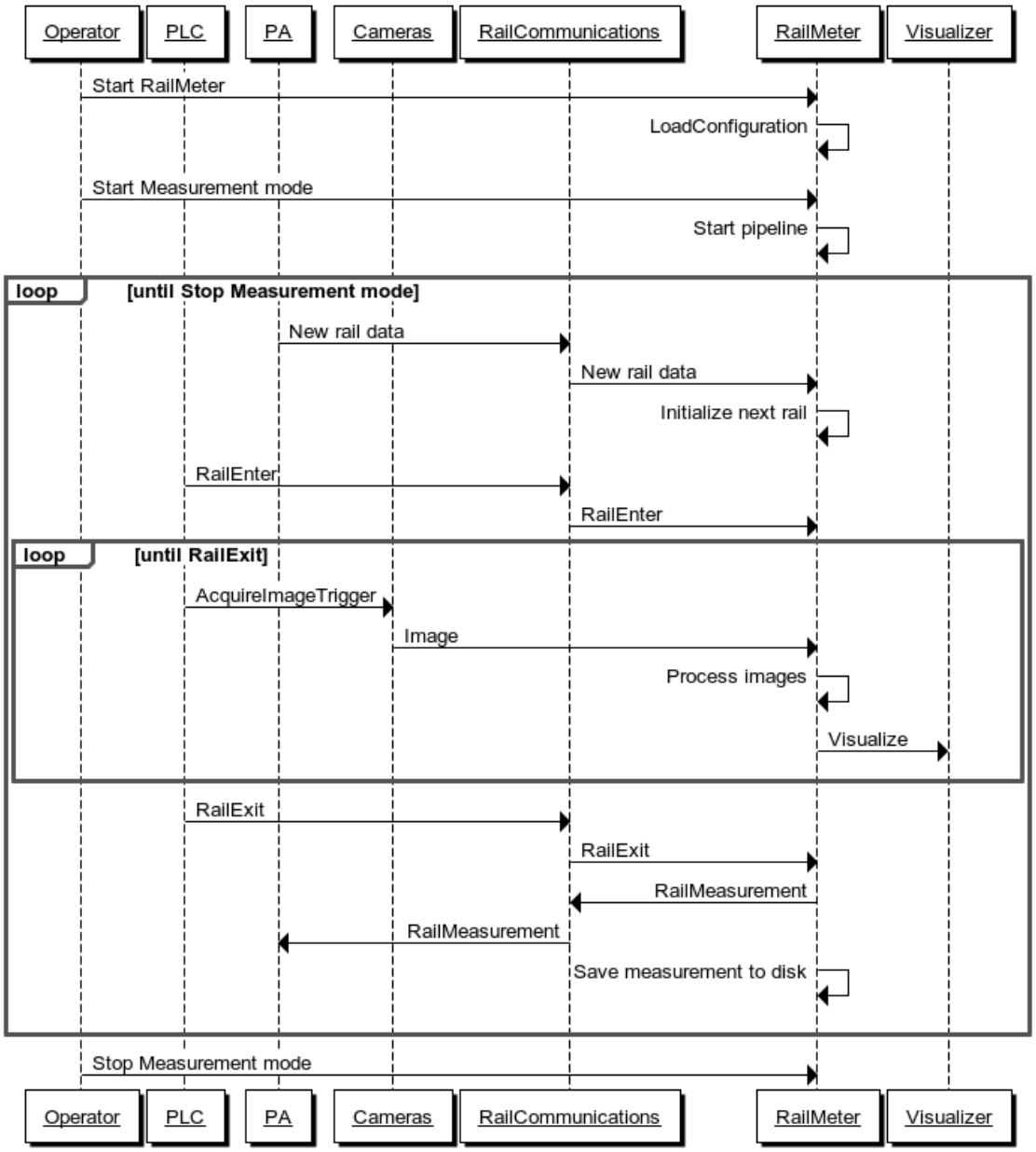


Figura 30: Diagrama de secuencia del modo de medición

Una vez que se ha iniciado el programa, este carga toda la configuración, tanto de normas como de perfiles, creada mediante el configurador. Este archivo de configuración es el llamado "RailStd.xml" y es cargado por el medidor al inicio del mismo. Se cargará también la propia configuración interna del programa.

Para llevar a cabo un proceso de medición correcto se debe recibir previamente el tipo de carril que se va a analizar y sobre que norma se debe contrastar. Estos datos serán recibidos a través de la red como se ha explicado en el apartado 3.

Una vez sean conocidos los datos del siguiente carril a analizar se debe extraer tanto el modelo como el propio conjunto de tolerancias de la configuración cargada inicialmente. De este modo se inicializa el siguiente carril a analizar.

Una vez se ha realizado todo este proceso el medidor está listo para iniciar la medición del carril.

Este modo de medición estará funcionando sin interrupción hasta que el operario decida terminarlo.

5.1.1. Medición de carriles

La medición de los carriles es llevada a cabo mediante la ayuda de una DLL creada especialmente para este fin, llamada *Geometry*. Esta librería contiene una serie de estructuras de datos, así como de primitivas geométricas y operaciones entre estas que facilitará el análisis de la información.

Las cámaras capturan imágenes del carril que pasará por el plano láser y se realizarán una serie de transformaciones sobre estas imágenes. Estas transformaciones se llevan a cabo de forma secuencial en una estructura de tipo *Pipeline* creada para tal fin; esto se explicará con mayor detalle en la sección 7.2.

5.1.2. Representación de los resultados

Mientras el carril está siendo medido, cada sección analizada se irá visualizando en todos los programas visualizadores que tenga en ese momento el medidor conectados. Estas secciones serán almacenadas también en memoria para un guardado de las mismas en el momento de finalización del carril.

Una vez se ha terminado la medición del carril, es decir, el PLC ha enviado a través del programa de comunicaciones *RailCommunications* el mensaje *RailExit*, los datos deben ser almacenados en disco (en el formato que se describe en el anexo E - Formato de fichero de resultados) y también deben ser enviados por red hasta el ordenador de proceso, mediante el uso del programa de comunicaciones como intermediario. Esto se consigue mediante el mensaje *LastRailMeasurement*.

5.2. Modo de calibración

Para poder combinar las imágenes obtenidas de las cámaras es necesario poder traducir los puntos de dichas imágenes a puntos del plano en el que se realiza la medición. Para ello han de conocerse los valores de determinados parámetros de las mismas.

Con este fin se utiliza una plantilla de calibración, una placa sobre la que se disponen varios cilindros, cuyas posiciones entre sí son conocidas. Se trata de detectar los cilindros en las imágenes captadas, calcular sus posiciones relativas y obtener los parámetros que permitan realizar la traducción deseada de la imagen al plano láser.

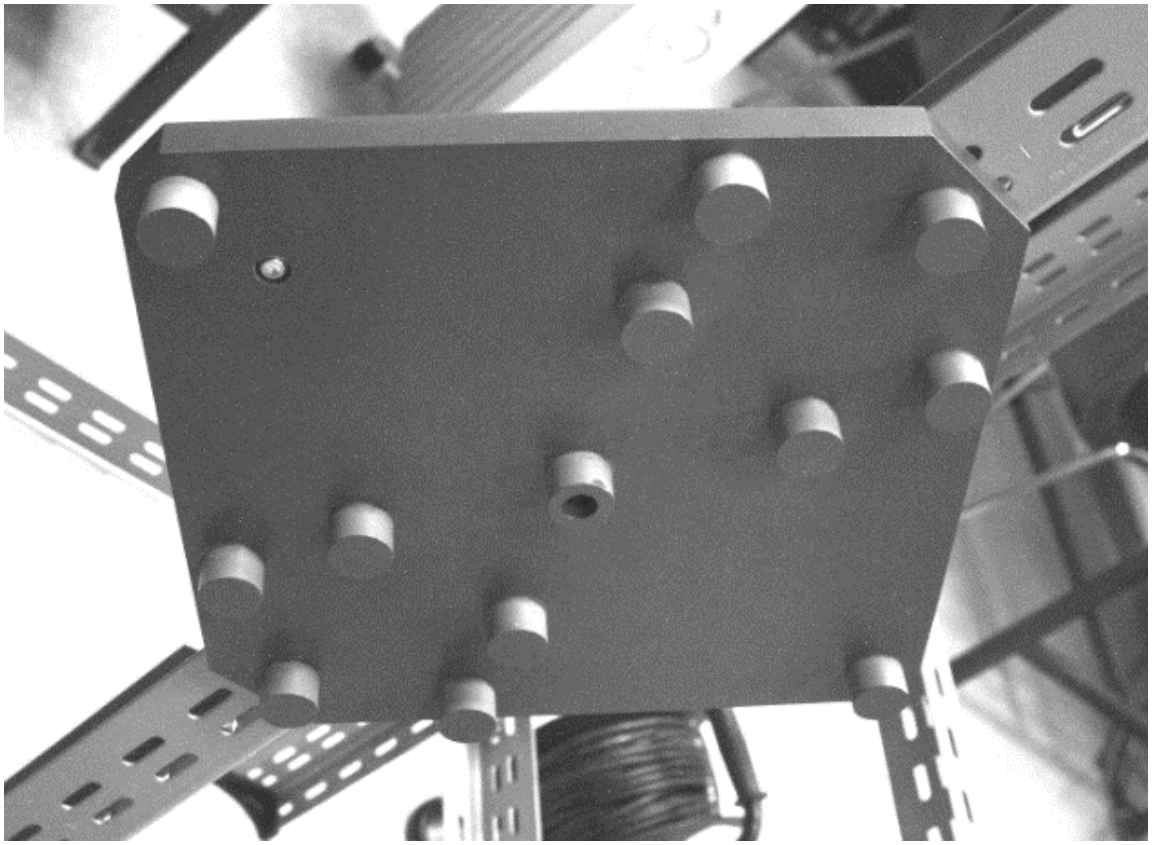


Figura 31: Plantilla de calibración, vista desde una de las cámaras utilizadas

De este modo, desde cada una de las cámaras es visible el contorno de algunos de los cilindros. Conociendo el ángulo aproximado de cada cámara y el patrón que siguen los cilindros en la plantilla, es posible asociar cada contorno a un cilindro. Comparando las posiciones relativas de los contornos y el patrón de los cilindros es posible hallar la posición de cada una de las cámaras en relación con un marco de referencia conocido, que es el plano láser. Puede verse una explicación más detallada del proceso de calibración y el algoritmo que se describe en el documento V - Calibración.

5.2.1. Funcionamiento

Como puede verse en la Figura 32, el operador es quien dirige el funcionamiento del modo de calibración. Una vez activado, debe elegir cuándo desplegar la plantilla; cuando ésta esté desplegada, el operador debe seleccionar qué imagen utilizar para la calibración; finalmente, el

operador debe decidir si acepta los resultados de calibración o no. Si no los acepta, puede repetir el proceso o continuar empleando los parámetros de calibración anteriores.

En el diagrama se muestran los mensajes AcquireImageTrigger, del PLC a las cámaras, e Image, de las cámaras al medidor, como posteriores a la solicitud de adquisición de imágenes por parte del operador. Sin embargo, realmente estos mensajes se producen constantemente, pero el medidor ignora las imágenes que recibe salvo cuando se encuentra en el modo de medición y cuando debe adquirir imágenes para calibrar.

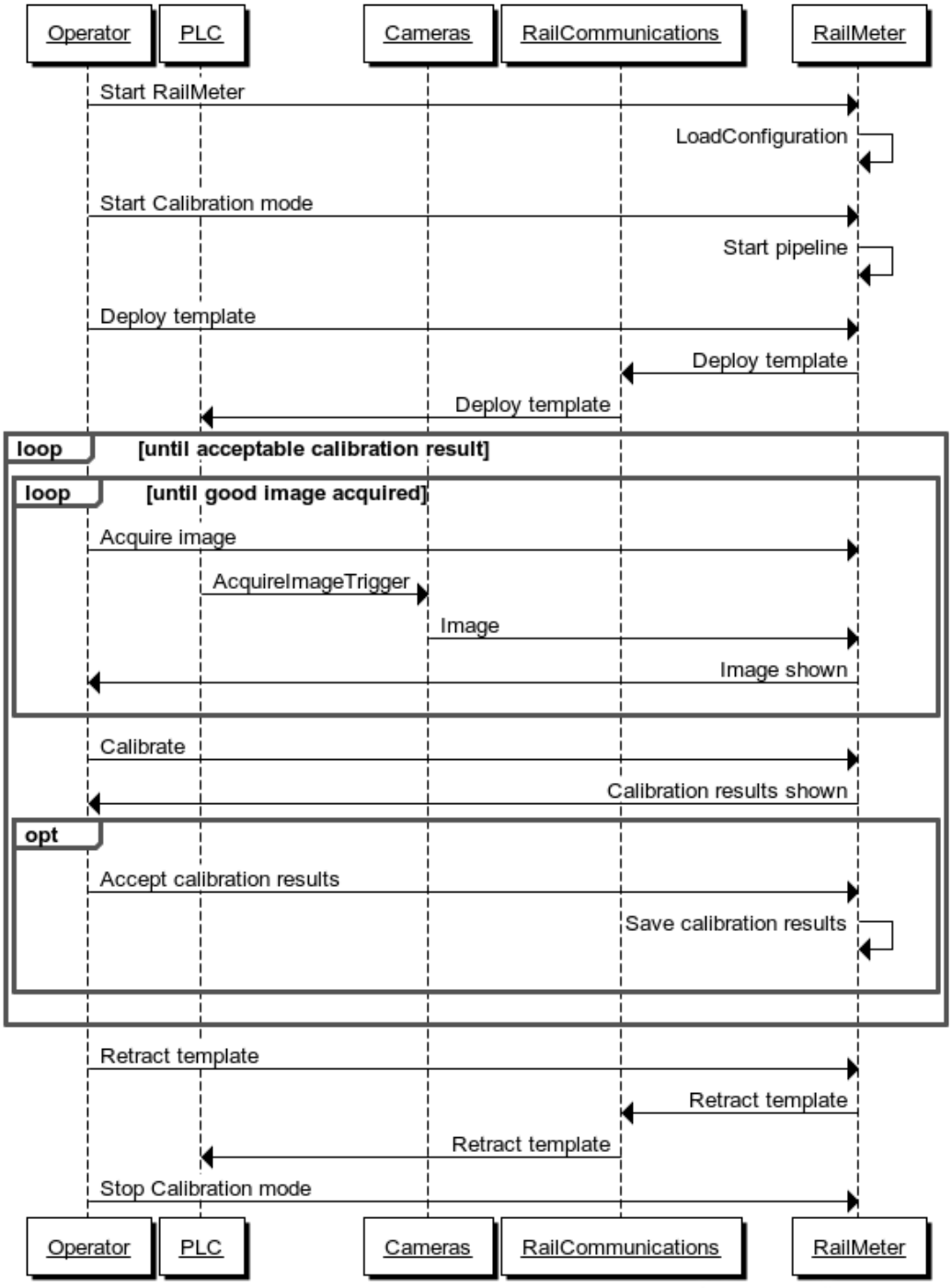


Figura 32: Diagrama de secuencia del modo de calibración

6. Configuración del sistema

6.1. Parámetros del medidor

Los parámetros de configuración disponibles son los siguientes:

- Calibration**:**
  - CircularCalTarget: Ruta del fichero que especifica las posiciones de los cilindros en la plantilla de calibración.
  - CalibrationAlgorithm: Nombre del algoritmo empleado para el ajuste de contornos a elipses en la calibración.
  - Sigma: Intensidad del desenfoque gaussiano aplicado.
  - MaxExtraAngle: Ángulo máximo (en grados) bajo la horizontal para el que no se descartan los contornos.
  - NearAngleLimit: Ángulo máximo de diferencia entre el cilindro de la plantilla de la calibración y un círculo de la imagen para que este pueda considerarse equivalente.
  - NumPoints: Máximo número de puntos que se usarán en la calibración.
  - CameraParam(1-4): Ubicación del fichero del que se leerán los parámetros intrínsecos de cada una de las cámaras.
  - CameraRotation(1-4): Ángulo de rotación aproximado de cada una de las cámaras, en grados.
  - CameraPose(1-4): Ubicación del fichero del que se leerán en modo de medición y en el que se escribirán en modo de calibración los parámetros extrínsecos de cada una de las cámaras.
- Encoder:
  - PulseToDistanceMm: Distancia en milímetros entre pulsos del *encoder*.
- File output:
  - FileOutputEnabled: Si se guardan en fichero los resultados de la medición.
  - FileOutputBasePath: Ruta en la que se almacenarán los resultados de la medición.
- Frame grabber:
  - AcqRate: Número de imágenes tomadas por segundo (no es aplicable al modo de captura desde cámaras).
- GUI:
  - DisplayRate: Número de imágenes que se mostrará cada segundo.
- Image source:
  - ImageSourceType: Tipo de medio desde el cual se adquirirán las imágenes para medir, en este caso se pueden dar los siguientes valores:
    - Image File: Usar como fuente de imágenes ficheros físicos que las contengan.
    - Folder: Usar como fuente, un conjunto de imágenes que estarán presentes en un directorio.
    - GigEVision: Driver libre GigEVision para captura de las imágenes de las cámaras.

        - SaperaLT: Driver propio del fabricante de las cámaras DALSA que se utilizan en el sistema para captura de imágenes.
        - Video File: Usar como fuente un video que contenga una secuencia de imágenes de un carril.
    - NumSources: Número de cámaras que estarán presentes en el sistema.
    - GrabTrigger: Tipo de disparador que indicará cuando se capturarán las imágenes, se pueden dar dos tipos:
        - Hardware: El disparo se realizará mediante la recepción de pulsos. Estos pulsos habrá sido generados por un dispositivo hardware.
        - Software: El momento del disparo será indicado al sistema mediante software.
    - ImageAcquisitionMode: Modo de adquisición de imágenes. Se pueden dar cuatro casos diferentes:
        - Sync: Espera hasta que haya imágenes disponibles en la fuente, y entonces las utiliza.
        - Async: Espera en otro hilo hasta que haya imágenes disponibles en la fuente, y entonces las utiliza.
        - Async + Callback: Cuando la fuente recibe una nueva imagen, dispara un evento que provoca la ejecución de un *callback* encargado de procesarla.
        - Async + Callback + Worker: Cuando la fuente recibe una nueva imagen, dispara un evento que provoca la ejecución de un *callback* que la procesa en un hilo distinto.
    - Source0(1-4): Fuente de las imágenes que el sistema va a utilizar como entrada para la medición, una por cada número de cámaras en el sistema, con un máximo de cuatro.
    - SaperaLTCameraType: Ruta del fichero de configuración de las cámaras (*.ccf*).
- Localization:
    - Language: Idioma en el que se muestra la interfaz gráfica del sistema. Se han implementado dos. Siempre que se cambie esta variable, el sistema deberá ser reiniciado para que los cambios surtan efecto:
        - Es: El idioma del medidor se cambiará a español.
        - En: El idioma del medidor se cambiará a inglés.

      La modificación de esta variable conlleva el cambio de cultura del medidor completo. Por ejemplo, los puntos pasaran de ser separadores decimales en inglés, a ser separador de millares. Por lo tanto, si se realiza este cambio se deberá cambiar también todos los valores que utilicen variables enteras.

- Logging:
    - SaveRawImages: Variable booleana que indica si se deben guardar las imágenes tal y como el medidor las ha introducido al sistema, esto puede ser necesario para su futuro tratamiento de modo offline.
    - SaveRawImagesPath: Ruta de acceso a la carpeta donde se van a almacenar las imágenes que se han captado por parte del medidor (si así se indica en la anterior variable).

- SaveRawImagesNumRails: Número de railes que se van a grabar a partir del inicio del medidor.
- SaveRailImagesNumRailsGap: Número de carriles no guardados entre guardados sucesivos. Es decir, si este número es 10, se guardará el carril número 1, el carril número 10, etc.

  El uso de esta variable *SaveRailImagesNumRailsGap* y la anterior, *SaveRawImagesNumRails*, tiene como objetivo no saturar el disco duro del computador donde se encuentre el programa medidor, ya que el almacenado de todas las imágenes medidas de un carril, tiene como media un peso aproximado de 5 GB.
- SaveRawProfiles: Variable booleana que indica si se deben guardar las nubes de puntos que el medidor ha generado en la etapa ProfileGenerator.
- SaveRailProfilesPath: Ruta de acceso a la carpeta donde se van a almacenar las nubes de puntos que se han generado por parte del medidor (si así se ha indicado en la anterior variable).
- EstimatedRailLengthM: Estimación de la longitud de los carriles a medir, en metros.
- EstimatedFramesPerM: Estimación de la velocidad de captura, en imágenes por segundo.

- Network
  - ServerEnabled: Variable booleana que indica si el servidor que envía los datos a los programas visualizadores, se encuentra activado o no.
  - ServerPort: Puerto del servidor anterior.
- Pipeline
  - QueueSize: Tamaño de las colas del pipeline, es decir, el número máximo de elementos que se pueden almacenar en cada una ellas.
  - OtherChartsSize: Escala de las gráficas que no son las anterior, en este caso este valor representa el número de milisegundos.
- Rail Communication**:**
  - OpenRailCommunications: Variable booleana que indica si debe ejecutarse o no el proceso *RailCommunications* cuando se requiera.
  - PAAddress: Dirección IP del ordenador de proceso.
  - PAPort: Puerto TCP en el que escucha el ordenador de proceso.
  - PLCAddress: Dirección IP del PLC.
  - PLCPort: Puerto TCP en el que escucha el PLC.
- Rail configuration XML
  - ConfiguratorXMLFile: Ruta en la que el medidor busca el fichero XML que contiene los perfiles y normas. Este fichero es generado por el programa configurador.
- Rail Id
  - Increment: Incremento que sufrirá el identificador del carril actual cuando acabe el carril y entre uno nuevo en el sistema.
  - MaxIncrement: Incremento máximo del siguiente identificador autogenerado.
  - MaxLength: Tamaño máximo del identificador de carril, en caracteres.
  - MinIncrement: Incremento mínimo del siguiente identificador autogenerado.

6.2. Contraseña de administrador

La contraseña de administrador será utilizada para ocultar al usuario los parámetros de configuración avanzados que solo pueden ser cambiados por el administrador del sistema.

Esta contraseña será almacenada en un archivo diferente al de configuración. Este archivo será oculto por motivos de seguridad y tendrá un nombre poco descriptivo, para que no se pueda llegar a modificar por error, pero lo suficientemente descriptivo como para que se entienda que es un archivo del sistema.

El nombre elegido para el archivo que va a almacenar la contraseña es *RailMeter.nfo*. Este archivo contendrá únicamente la contraseña usada para acceder con el rol de administrador a la configuración del sistema. La contraseña será almacenada en este fichero con un cifrado *AES* (*Advanced Encryption Standard*).

Si este archivo fuese eliminado del sistema, el medidor no permitiría al usuario acceder a la configuración avanzada de usuario, haciendo que éste tuviese que restablecer la contraseña antes de poder acceder, esto se realizaría mediante el uso del formulario explicado en la sección 4.7. En este caso la contraseña previa que se debe introducir sería la contraseña por defecto del medidor, que está fijada dentro del propio ejecutable.

7. Diseño software

El programa medidor emplea múltiples hilos para procesar simultáneamente las imágenes obtenidas de todas las cámaras al tiempo que mantiene actualizada la interfaz gráfica y atiende conexiones. Para coordinar el funcionamiento de los hilos de procesamiento de imágenes se utiliza un sistema de colas y eventos que se describe en la sección 7.2.

Como se ha visto anteriormente, el medidor tiene dos modos diferentes de funcionamiento: el de medición y el de calibración, que se han explicado en la sección 5.

Se implementa también una lógica de estados, descrita en la sección 7.2.2, para regular la transición entre estos modos y su funcionamiento interno.

En cuanto a la interfaz gráfica, implementada con la tecnología de Windows Forms, se describe en la sección 4.

7.1. Estructura general

Puede considerarse que el programa medidor se divide en cinco componentes diferenciados:

- **Control:** Comunica la interfaz con los restantes componentes y estos entre sí, gestiona la configuración y proporciona funcionalidades comunes.
- **Interfaz:** Se comunica con el usuario y con elementos externos al sistema. Incluye la interfaz gráfica (GUI), pero también la comunicación de los resultados a través de la red. No incluye la comunicación con el ordenador de proceso, el PLC ni las cámaras, que forman parte de la medición y la adquisición.
- **Adquisición:** Captura imágenes de las cámaras, las procesa y combina y las pone a disposición de los componentes de interfaz, medición y calibración.
- **Medición:** Dados los perfiles de carriles generados por el componente de adquisición, computa los valores de sus dimensiones y los envía a la interfaz.
- **Calibración:** Dadas las imágenes obtenidas por el componente de adquisición, calcula los parámetros de las cámaras y los envía al componente de control.

La relación entre estos componentes puede verse en la Figura 33.

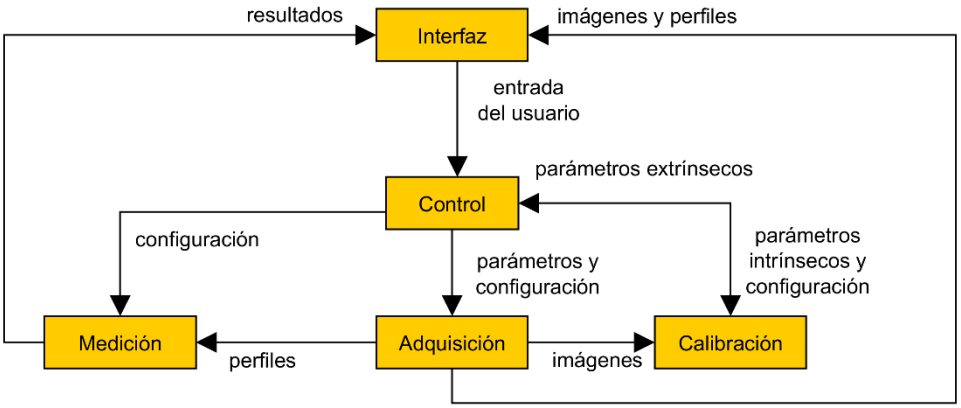


Figura 33: Componentes del programa medidor

7.2. Control

Este componente se encarga de comunicar la interfaz que se verá en la sección 7.2.2 con los restantes componentes y a estos entre sí. Debe gestionar de forma correcta la configuración del programa completo, así como proporcionar funcionalidades comunes.

En la Figura 34 se puede observar el diagrama de clases de alto nivel que conforman el componente de control del medidor.

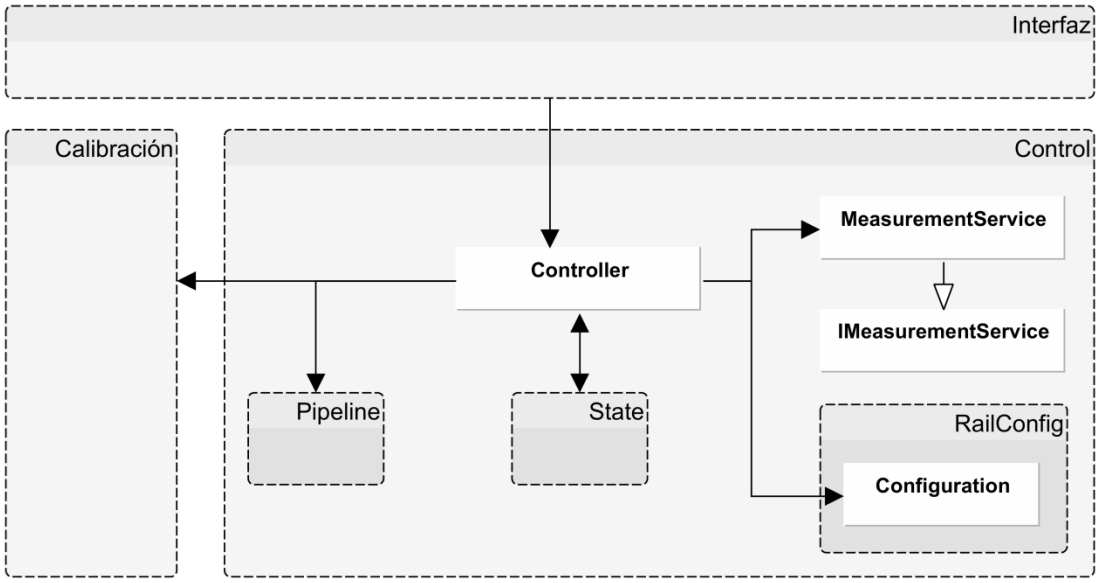


Figura 34: Diagrama de clases de la parte *Control* del medidor

El componente de *control* se encarga de gestionar la interfaz WCF que en el diagrama mostrado en la Figura 34, se muestra con el nombre *IMeasurementService*. Se puede observar que existe una clase llamada *MeasurementService* que implementa el interfaz *IMeasurementService*.

La interfaz IMeasurementService será el contrato que el medidor expone para comunicarse con el programa de comunicaciones *RailCommunications*. Esta interfaz se describe en el documento IV - Comunicaciones de forma más detallada.

La parte principal de la que se encarga la parte de control es del despliegue del *pipeline* del medidor. Este *pipeline* es el encargado de adquirir, tratar las imágenes, llevar a cabo el proceso de medición de dimensiones y exponer los resultados a todas las partes interesadas, como se puede observar en la sección 7.2.1.

El componente de control también se encarga de mantener, gestionar y realizar todas las transiciones entre estados del patrón State que se implementa en el medidor y que se explica con mayor detalle en la sección 7.2.2. También lleva a cabo el proceso de calibración del sistema. Esto se explica con más detalle en la sección 7.6.

Por último, el componente de *control* es el encargado de cargar el archivo XML que contiene las normas y modelos de carril. Este archivo es el generado mediante el programa

configurador. Este componente deberá cargar todos los datos necesarios para llevar a cabo el proceso de medición explicado en la sección 7.5. Esto se lleva a cabo mediante el uso de la clase *Configuration* de la biblioteca *RailConfig*, que queda fuera del ámbito de este TFM.

7.2.1. Estructura *Pipeline*

Para llevar a cabo las tareas de medición y calibración se usara una estructura llamada p*ipeline* o tubería, esta se usará para realizar las acciones en el modo medición y se aprovechará, aunque en menor medida también para el modo de calibración.

La Figura 35 muestra una descripción gráfica de un *Pipeline*.

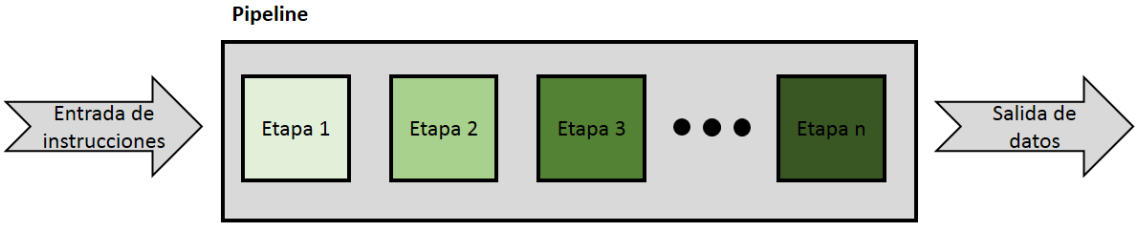


Figura 35: Estructura *Pipeline*

Los datos entran en el *pipeline* y se van procesado de forma secuencial en cada etapa hasta que llegan a la última y salen de la estructura cuando ya se hayan realizado todas las acciones y transformaciones necesarias, en una *pipeline* se pueden tener datos de forma continua en todas las etapas de la misma, es decir, no se espera a que un dato haya pasado por todas las etapas para introducir uno nuevo, de esta forma, se consigue un mayor aprovechamiento de los recursos disponibles.

El sistema implementa una estructura *pipeline* para el tratamiento y procesado de las imágenes tomadas, tal y como se puede ver en la Figura 36.

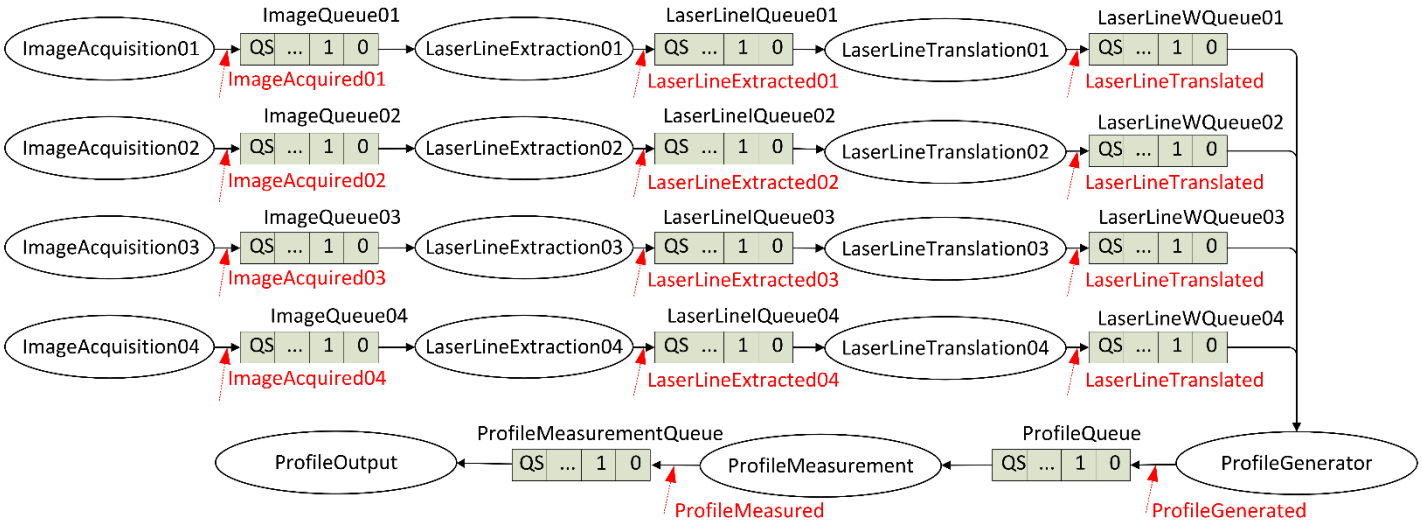


Figura 36: Pipeline del medidor

La captura de imágenes y entrada por lo tanto para el pipeline, se realiza mediante hardware como se ha explicado en la sección 3.1 y se procesa como se indica en la sección 7.4.

La estructura de este pipeline está formada básicamente por una serie de acciones, las cuales estarán secuenciadas. El resultado final de una acción sirve como entrada para la siguiente, todo ello conformando la estructura en pipeline.

Esta estructura pipeline está replicada en sus etapas iniciales para cada una de las cámaras del sistema.

El pipeline está formado por tres entidades básicas, un conjunto de hilos, mostrados en la Figura 36, mediante elipses, estos hilos procesan la salida del hilo anterior, que éste habrá dejado en una cola, mostrada mediante los rectángulos. Una vez finalizado el procesamiento depositan el resultado en la cola de salida.

Estos hilos empiezan a procesar cuando el hilo predecesor de la estructura termina, y dispara el evento correspondiente, el cual en la Figura 36 se puede observar mediante flechas de color rojo con el nombre del evento generado en el mismo color.

El primer conjunto de hilos *ImageAcquisition*, adquiere las imágenes de las cámaras y las almacenan en la siguiente cola de salida, *ImageQueue*. Cuando esto se produce se genera un evento llamado *ImageAcquired*, haciendo que se ejecute el siguiente hilo.

En la segunda etapa de este pipeline se extraen los elementos que son procesados por el siguiente hilo, *LaserLineExtraction*. Éste, tal y como indica su nombre, extrae las líneas láser, en coordenadas de la imagen, es decir, en píxeles las cuales son situadas en la cola de salida *LaserLineIQueue*, disparando tras hacer esto el evento *LaserLineExtracted*.

Para la tercera etapa, se extraen de la cola anterior. Esto es realizado por el siguiente hilo de ejecución *LaserLineTranslation*, el cual es el encargado de realizar la traducción de pixeles de las imágenes a coordenadas reales del mundo. Esto es posible realizarlo gracias al proceso de calibración. Este proceso de calibración deberá haber sido llevado a cabo antes del inicio del proceso de medición. Una vez este hilo ha terminado con la traducción de las coordenadas, introduce las imágenes en la siguiente cola de imágenes, *LaserLineWorldQueue*, y dispara el evento *LaserLineTranslated*.

La cuarta etapa recoge las imágenes de las colas anteriores y las junta en una única estructura, formando así una nube de puntos en coordenadas reales para el carril. Esto es realizado por el hilo *ProfileGenerator*. Una vez terminado este proceso, las cuatro imágenes que han sido utilizadas para construir la nube de puntos del perfil, se habrá convertido en una única. Esta será dejada en la siguiente cola del Pipeline, *ProfileQueue*. Tras realizar esto, el hilo disparará el evento ProfileGenerated.

Esta nube de puntos almacenada en la cola anterior, es consumida por el hilo *ProfileMeasurement*, el cual se encarga de realizar todas las mediciones dimensionales del carril. Éstas son las dimensiones que posteriormente se mostrarán al usuario mediante una serie de gráficos. Estas mediciones son dejadas en la cola *ProfileMeasurementQueue*, y posteriormente se dispara el evento *ProfileMeasured*.

Por último, el hilo de ejecución *ProfileOutput*, se encarga de transmitir estas mediciones que van siendo extraídas de la cola anterior a las vistas que sean necesarias. Estas vistas serán

tanto las gráficas finales para el usuario, como la escritura de los ficheros con los resultados de la medición.

Se puede observar que hay tres etapas que presentan un comportamiento diferente al resto, estas son las siguientes:

- Etapa inicial, es decir, *ImageAcquisition*. Al ser la etapa inicial no recoge los datos de ningún tipo de cola previa, si no que adquiere las imágenes directamente de las cámaras y las introduce en las colas de salida, *ImageQueue*.
- El caso contrario al anterior, *ProfileOutput*. Al ser la etapa final recoge los datos de la cola *ProfileMeasurementQueue*, y los envía a las vistas necesarias.

Un caso especial es también *ProfileGenerator*. Al ser la etapa que genera el perfil completo, deberá coger una imagen de cada una de las colas anteriores, *LaserLineWQueue*. Con cada una de estas colas debe componer el perfil completo del carril y pasarlo a la siguiente cola.

En la Figura 37 se muestra un diagrama de las clases que forman el *pipeline* del medidor.

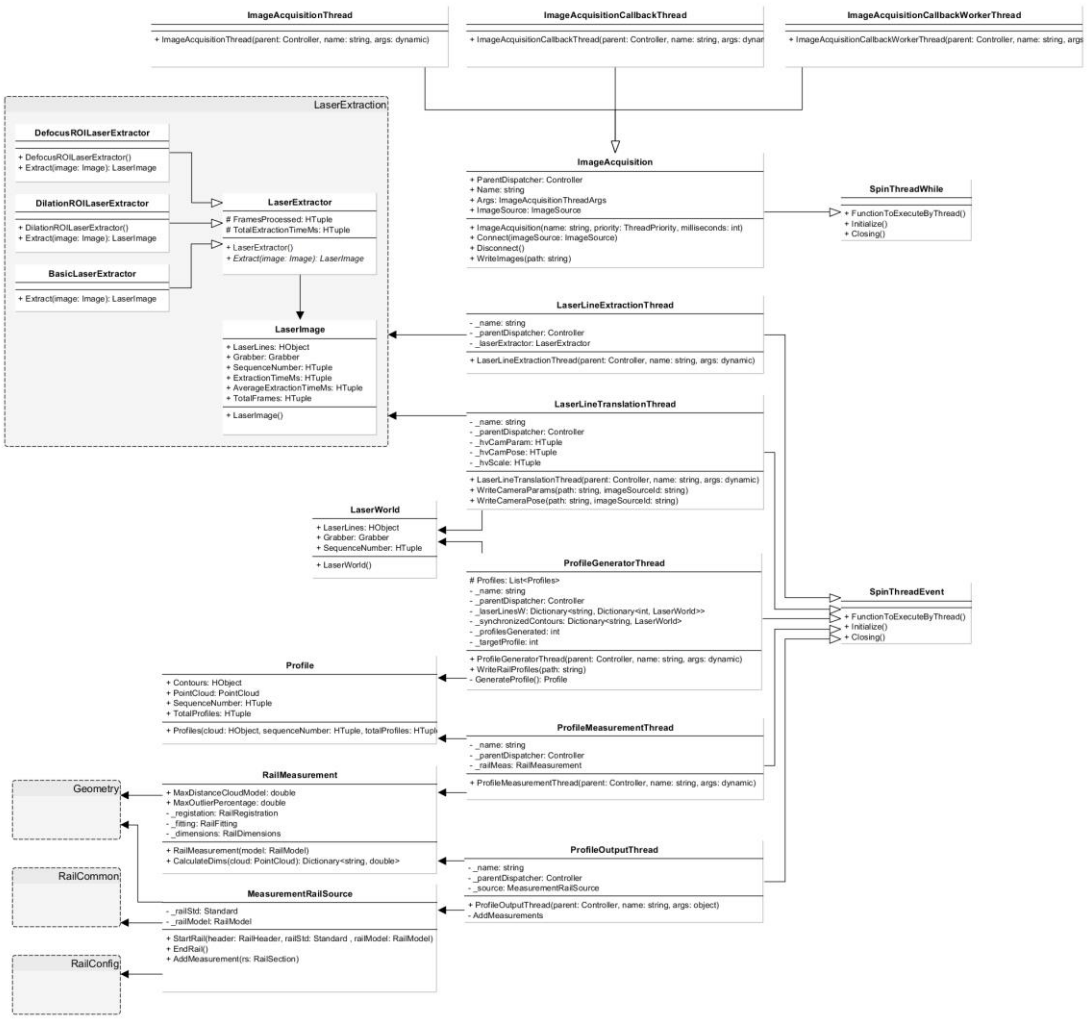


Figura 37: Diagrama de clases del *pipeline* del medidor

Los cometidos de cada una de las clases son como sigue:

- *SpinThreadWhile*: Clase base que proporciona funcionalidad para ejecutar una función en un hilo independiente cada cierto tiempo.
- *SpinThreadEvent*: Clase base que proporciona funcionalidad para ejecutar una función en un hilo aparte cuando se produce un evento.
- *ImageAcquisition*: Clase base para los hilos que se ocupan de la captura de imágenes.
- *ImageAcquisitionThread*: Hilo de captura de imágenes básico.
- *ImageAcquisitionCallbackThread*: Hilo de captura de imágenes que se ejecuta cada vez que se dispara un evento hardware procedente de la cámara.
- *ImageAcqusitionCallbackWorkerThread*: Hilo que se ejecuta cada vez que se dispara un evento hardware procedente de la cámara, y que emplea otro hilo independiente para la captura.
- *LaserLineExtractionThread*: Hilo que toma las imágenes capturadas por *ImageAcquisition* y extrae los contornos láser.
- *LaserExtractor*: Clase base para extraer contornos de láser de una imagen.
- *BasicLaserExtractor*: Clase que extrae todos los contornos de láser presentes en una imagen.
- *DefocusROILaserExtractor*: Clase que extrae contornos de láser en la región contigua al último que se extrajo. La región de búsqueda se genera mediante el desenfoque de la línea encontrada.
- *DilationROILaserExtractor*: Clase que extrae contornos de láser en la región contigua a la última línea que se extrajo. La región de búsqueda se genera mediante la dilatación de la línea encontrada.
- *LaserImage*: Clase que encapsula contornos láser medidos en coordenadas de la imagen.
- *LaserLineTranslationThread*: Hilo que traduce los contornos láser tomados de *LaserLineExtractionThread* a coordenadas del mundo.
- *LaserWorld*: Clase que encapsula contornos láser medidos en coordenadas del mundo.
- *ProfileGeneratorThread*: Hilo que combina los contornos de todas las cámaras procedentes de *LaserLineTranslationThread* y genera un perfil.
- *Profile*: Clase que representa un contorno láser completo.
- *ProfileMeasurementThread*: Hilo que toma un perfil generado por *ProfileGeneratorThread* y mide las dimensiones presentes en él.
- *RailMeasurement*: Clase que contiene los resultados de medición de un carril.
- *ProfileOutputThread*: Hilo que toma los resultados de la medición realizada por *ProfileMeasurementThread* y los muestra, almacena y envía.
- *MeasurementRailSource*: Clase que expone los resultados de la medición a las clases responsables de la visualización.

7.2.2. Lógica de estados

El medidor implementa una lógica de estados, mediante los cuales son controladas las acciones que se realizan en cada momento. Estos estados pueden cambiar dependiendo de la recepción de una serie de mensajes. Estos mensajes pueden ser enviados por cualquiera de las entidades hardware vistas anteriormente, o generadas de forma manual mediante una interfaz de generación de mensajes incluida en el medidor, la cual se verá en detalle más adelante, así como mediante una serie de botones de la interfaz del propio medidor.

La Figura 38 muestra un diagrama que contiene todos los estados posibles en los que se puede encontrar el medidor, así como todas las transiciones entre los mismos.

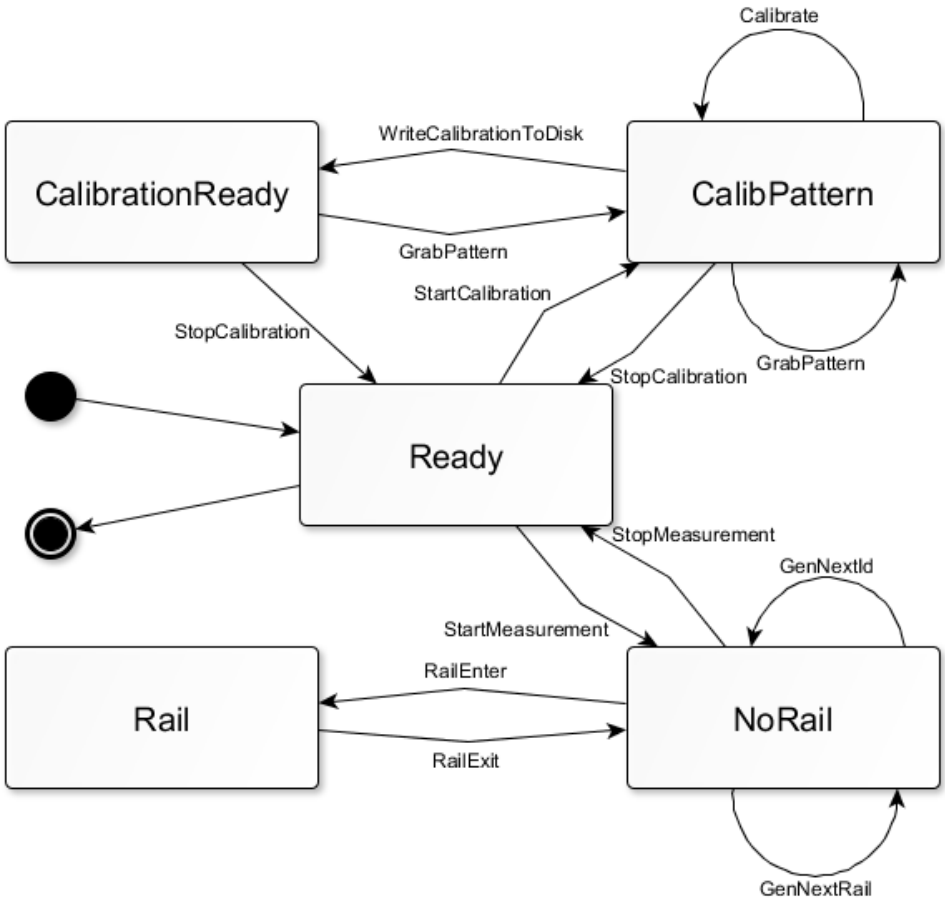


Figura 38: Diagrama de estados del medidor

El estado *Ready* es el estado en el que se encontrará el medidor nada más iniciarse, sirviendo pues de estado inicial. De este estado se puede iniciar el modo medición, pasando entonces al estado *NoRail*. En el estado *Ready* se permite también modificar la configuración global del sistema, así como iniciar el modo calibración, pasando entonces al estado *CalibPattern*.

Si se realiza la acción adecuada para iniciar la medición, es decir, realizar la transición *StartMeasurement* (explicada con más detalle en la sección 4), el medidor pasará al estado *NoRail*, en el cual se pueden realizar las siguientes transiciones:

- *RailEnter*, esta transición se lleva a cabo cuando el medidor recibe un mensaje del PLC o mediante la generación de este mensaje en la ventana de generador de eventos del medidor. Este mensaje indica al sistema que un nuevo carril ha entrado en la zona de medición y debe ser procesado. Esto hace que el sistema cambie al estado *Rail*.
- *GenNextRail*, esta transición se lleva a cabo de forma automática tras recibir el mensaje anterior, RailEnter. El sistema generará la información necesaria (tanto de norma a usar como de tipo de carril) para posteriormente usarla para medir el siguiente carril.
- *GenNextId,* esta transición se lleva a cabo al igual que la anterior, es decir, tras recibir el mensaje *RailEnter*. El sistema generará el identificador de carril para usarlo para asignárselo al siguiente carril a medir.
- *StopMeasurement*, este mensaje devuelve el sistema al estado *Ready*, finalizando así el proceso de medición.

Tras la recepción del mensaje *RailEnter*, el medidor pasa al estado *Rail*, del cual tan solo se puede salir con el mensaje contrario, es decir, *RailExit*. Mientras dure el estado *Rail*, el sistema está procesando, es decir, midiendo el carril que está pasando en ese momento por el sistema.

Si se recibe el mensaje *StopMeasurement* y el medidor está en el estado *Rail*, este cambia de estado al *NoRail*, y posteriormente al estado *Ready*.

Una vez en el estado *Ready* si el medidor recibe el mensaje *StartCalibration*, éste cambia su estado a *CalibPattern*. En este estado el medidor puede llevar a cabo las siguientes transiciones:

- *Calibrate*, que activa todas las tareas de calibración necesarias para el sistema, como se ha explicado previamente en la sección 5.2.
- *GrabPattern,* que esencialmente obtiene una nueva imagen de cada una las cámaras del sistema para poder, posteriormente cuando se reciba el mensaje *Calibrate*, calibrar el sistema con estas.
- *WriteCalibrationToDisk*, que cambia el estado del medidor a *CalibrationReady*, y graba la calibración de cada una de las cuatro cámaras del sistema en disco, para que posteriormente pueda ser usada esta información a la hora de realizar la medición.
- *StopCalibration*, que devuelve el sistema al estado *Ready*, finalizando así el proceso de calibración.

Tras la recepción del mensaje *WriteCalibrationToDisk*, el medidor pasa al estado *CalibrationReady*, en este estado el sistema podrá volver a coger una imagen de cada una de las cámaras y volver al estado anterior, *CalibPattern* para repetir el proceso de calibración. Esto se puede realizar mediante el mensaje *GrabPattern*. También se puede desde el estado *CalibrationReady* terminar el proceso de calibración y pasar al estado *Ready*, mediante la recepción del mensaje *StopCalibration*.

La lógica de estados del medidor esta implementada mediante el uso del patrón de diseño *State*. Este patrón permite una gestión sencilla de la lógica de estados del medidor, haciendo más sencillo la adición de nuevos estados o el cambio entre ellos.

La implementación de este patrón necesita una serie de clases, estas se pueden ver en la Figura 39.

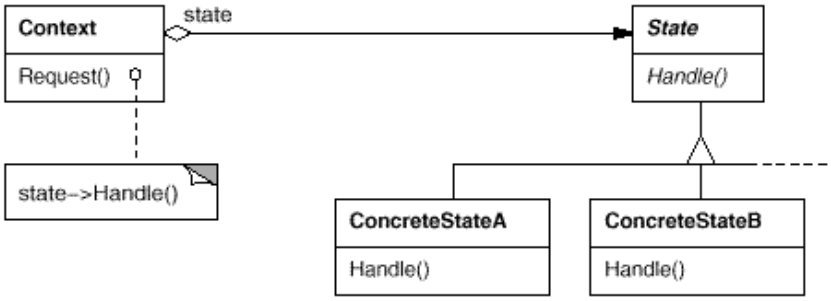


Figura 39: Implementación del patrón State

La clase *Context* define el interfaz que se puede utilizar para llevar a cabo las operaciones de cambios de estado. Además es una instancia del estado actual en el que se encuentra el programa.

La clase abstracta *State* define las transiciones entre estados, que posteriormente tendrán que implementar las clases que hereden de ella.

Las clases *ConcreteStateA* y *ConcreteStateB* implementan cada una de ellas algunas de las transiciones descritas en la clase *State*, y estas son propias de cada estado. Debe existir una clase por cada estado en el que se pueda encontrar el programa, y cada una de estas clases, implementar las transiciones entre ese estado y otro, así como llevar a cabo las acciones necesarias a realizar en estas transiciones.

En la implementación llevada a cabo, la clase *Context* del sistema creado se llama *AppContext*, la clase abstracta *State*, se llama *AppState* y por último las clases que implementan los estados se llaman:

- *ReadyState*, este es el estado inicial del programa, antes de iniciar cualqueriera de los modos medición o calibración.
- *NoRailState*, este es estado inicial del modo medición, cuando se ha iniciado la medición pero aún no hay ningún carril pasando por la zona de inspección.
- *RailState*, este estado indica que hay un carril en la zona de inspección y se está llevando a cabo una medición sobre el mismo.
- *CalibPatternState*, en este estado se podrán adquirir nuevas imágenes y llevar a cabo la calibración con las imágenes adquiridas.
- *CalibrationReadyState*, desde este estado puede pasarse al modo *CalibPatternState* si se desea volver a llevar a cabo el proceso de calibración.

El diagrama mostrado en la Figura 40 muestra el diagrama de clases implementado para llevar a cabo la gestión de estados mediante el patrón State en el medidor.

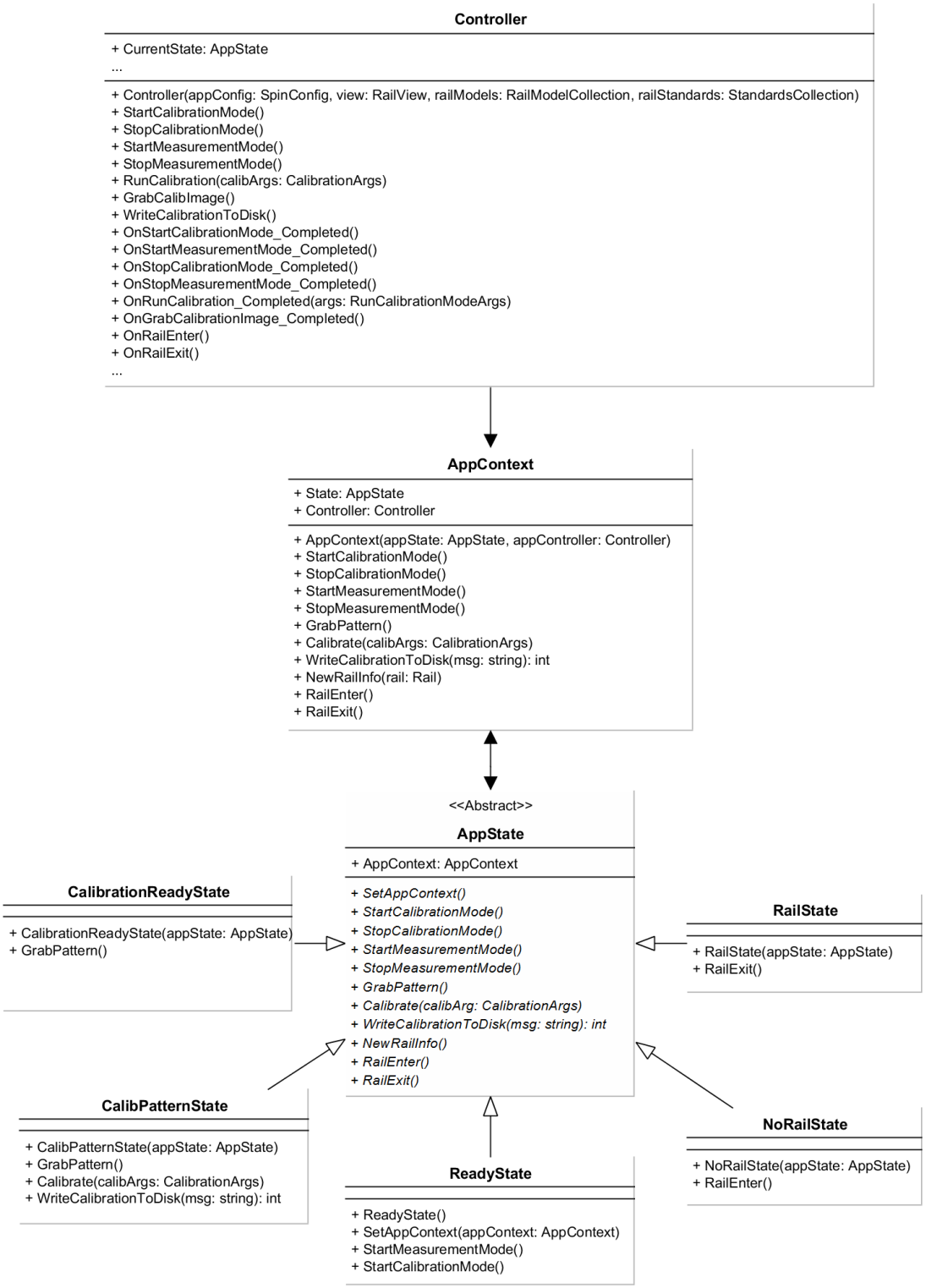


Figura 40: Diagrama de clases del patrón State del medidor

En caso de intentar realizarse una acción no permitida en un estado, como puede ser pasar de un estado a otro, no estando permitido este cambio, se produce un error y este error es registrado automáticamente. Se cancelará también el cambio de estado.

Las transiciones que se pueden realizar entre estados se pueden observar en la Figura 38.

7.3. Interfaz

El componente de interfaz es el encargado, no solo de mostrar formularios al usuario, sino que también es el encargado de interaccionar con el componente de control, y visualizar los datos medidos.

En la Figura 41 se puede observar el diagrama de clases para el componente de interfaz.

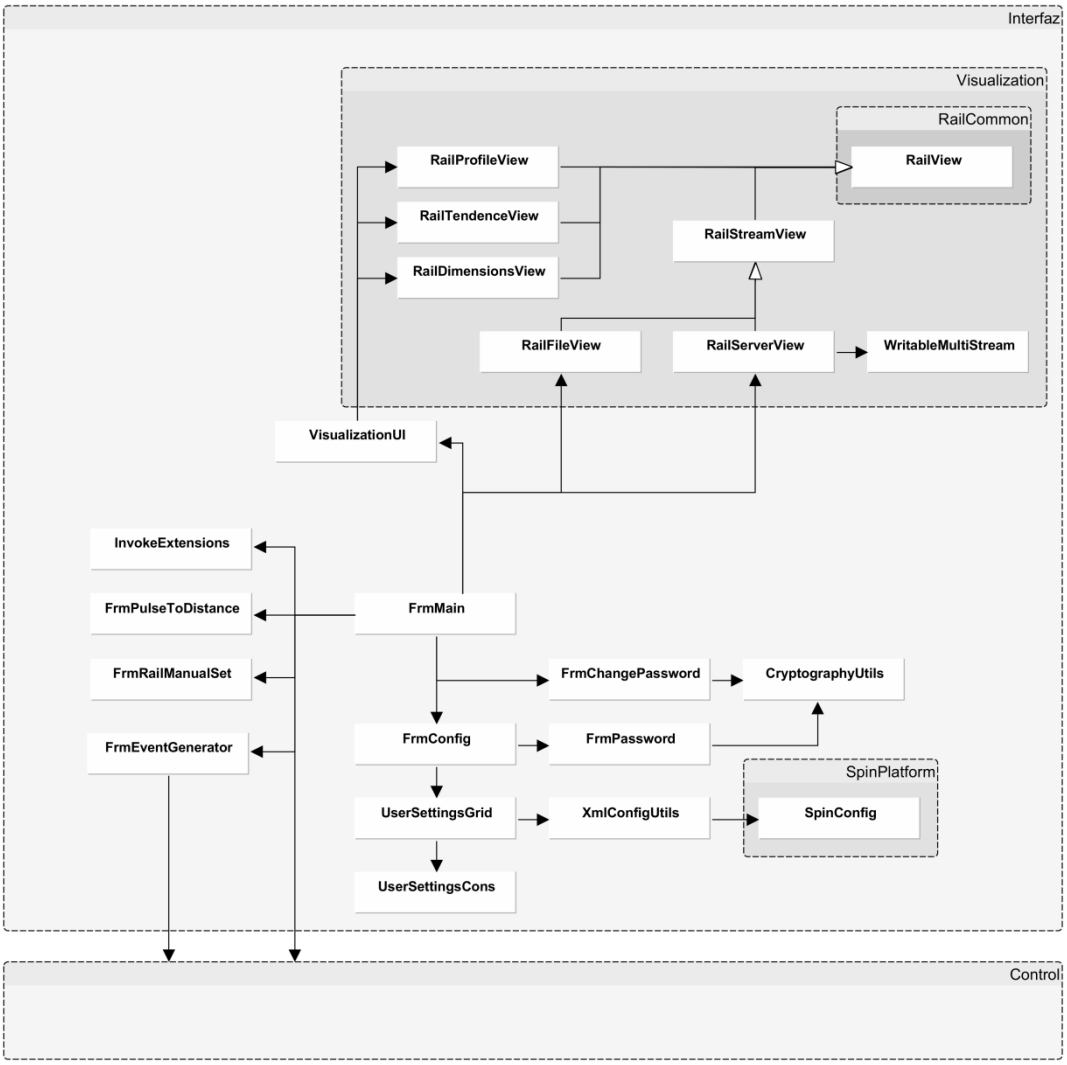


Figura 41: Diagrama de clases del componente de interfaz

Este componente está formado por una serie de formularios, los cuales empiezan con las letras *Frm* (Form), relacionados entre sí.

Un elemento central de esta interfaz es el formulario *FrmMain* que se puede ver en la sección 4.1. Este formulario es cargado al inicio del medidor, mostrando así la ventana principal del mismo.

En esta ventana principal se puede realizar un conjunto de acciones, las cuales permiten al usuario navegar por el medidor.

Otro formulario del medidor es el llamado *FrmPulseToDistance,* el cual se encarga de mostrar al usuario un formulario como el que se ve en la sección 4.6, mediante el cual se permite el cambio de la variable de configuración *PulseToDistanceMm* explicada en la sección 6.1.

El formulario *FrmRailManualSet* permite al usuario introducir el siguiente carril al sistema de forma manual, este formulario se puede ver en la sección 4.2.

Mediante el formulario *FrmEventGenerator*, el usuario puede emular manualmente el comportamiento tanto del ordenador de proceso PA como del PLC (véanse las secciones 3.1 y 3.2 respectivamente). Este formulario tiene comunicación con la parte de control, ya que es esta la encargada de manejar la información sobre el siguiente carril a procesar, de controlar la conexión y desconexión del PLC y del PA, así como de la entrada y salida del carril de la zona de medición. Este formulario se puede observar en la sección 4.5.

El formulario de configuración *FrmConfig* muestra al usuario una interfaz para realizar cambios en la configuración, y se puede ver en la sección 4.4. Estos cambios se realizan sobre un objeto del tipo *UserSettingsGrid*. Este objeto se crea al inicio del formulario, y cuando se cambia alguna de sus propiedades se actualiza la vista de la misma en el propio formulario. La lista de valores posibles del objeto *UserSettingsGrid* son extraídos del objeto *UserSettingsCons*. La inicialización del objeto UserSettingsGrid se lleva a cabo mediante el uso de una clase estática llamada *XmlConfigUtils*, la cual carga el archivo de configuración con la ayuda de la plataforma *Spin* y extrae los valores anteriores de los parámetros de configuración.

Para proteger el formulario de configuración antes comentado se ha creado otro formulario llamado *FrmPassword*, que muestra al usuario un diálogo para la introducción de una contraseña de administración. Este formulario se puede ver en la parte central de la Figura 19. La contraseña introducida se valida mediante el uso de una clase estática llamada *CryptographyUtils*. Esta clase valida la contraseña introducida por el usuario con una que deberá ser extraída de un fichero de texto. Este fichero de texto contiene la contraseña establecida por el usuario mediante un cifrado AES (véase sección 6.2).

Para llevar a cabo el cambio de la contraseña de administrador se ha creado un formulario llamado *FrmChangePassword*, el cual se puede observar en la sección 4.7. En este formulario el usuario deberá introducir la contraseña anterior, la cual se validará de una forma idéntica al caso del formulario anterior. Se deberá introducir también la nueva contraseña en caso de que la contraseña anterior haya sido validada de forma correcta. La nueva contraseña será cifrada mediante el uso de la misma clase estática que para el formulario anterior, es decir, *CryptographyUtils* y también mediante cifrado AES, e introducida en el fichero de texto.

Si el fichero de texto que contiene la clave fuese borrado por el usuario, el sistema, al no poder cargar la clave anterior, no permitiría al usuario acceder a la configuración de administrador hasta que la contraseña no sea reestablecida previamente. Este proceso se explica con mayor detalle en la sección 6.2.

La clase *VisualizationUI* es la encargada de mostrar los resultados procedentes del componente *Medición* (véase sección 7.5). Está compuesta por un conjunto de controles los cuales son los siguientes:

- *RailTendenceView*, esta clase proporciona una vista de los valores de las dimensiones a lo largo de todo el carril, permitiendo además seleccionar una posición concreta para mostrarla en las siguientes vistas.
- *RailDimensionsView*, esta clase proporciona una vista sencilla de los valores de las dimensiones en un momento concreto.
- *RailProfileView*, esta clase proporciona una vista del perfil del carril, a la que se superponen los valores medidos de las dimensiones.

Estas tres clases heredan de *RailView* que es una interfaz la cual define unos métodos y eventos que deben implementar las clases para poder visualizar carriles (lo que se describe en el documento VI - Visualizador).

Además de actualizar la clase *VisualizationUI*, el formulario FrmMain se encarga también de instanciar, mantener y gestionar las clases *RailFileView* y *RailServerView*.

La clase *RailFileView* es la encargada de almacenar las dimensiones medidas a lo largo del carril, para posteriormente, cuando el carril finalice, guardarlo en un archivo de resultados.

La clase *RailServerView* es la encarga de hacer llegar a los visualizadores que se encuentren escuchando a través del protocolo TCP. Para llevar a cabo esta tarea de enviar las mediciones a todos los visualizadores se emplea la clase *WritableMultiStream*. Esta clase escribe en una serie de *streams*, que se han ido añadiendo y a la hora de escribir en ellos, si alguno falla, es eliminado.

RailFileView y *RailServerView,* al igual que las tres anteriores, también implementan el interfaz *RailView*.

Por último para realizar cambios sobre elementos de la interfaz gráfica, se utiliza el patrón *InvokeIfRequired*. Esto es necesario ya que los elementos de la interfaz gráfica solo pueden ser modificados desde el hilo donde se ejecuta. La implementación de este patrón se encuentra en la clase llamada *InvokeExtensions*. Esta clase contiene una función llamada *InvokeIfRequired* a la cual se ejecuta en caso de querer realizar algún cambio de algún control de la interfaz gráfica.

7.4. Adquisición

El componente de adquisición es el encargado de la captura de imágenes y de su preprocesamiento. Está integrado por las clases del *pipeline* *ImageAcquisitionThread*, *LaserLineExtractionThread*, *LaserLineTranslationThread* y *ProfileGeneratorThread*, que se ejecutan en hilos con los mismos nombres. Estas clases se ocupan, respectivamente, de la captura de imágenes, la extracción de líneas de las imágenes, la traducción de las líneas a coordenadas del mundo y su combinación para formar perfiles completos.

El componente de calibración emplea directamente las imágenes obtenidas por *ImageAcquisitionThread*, mientras que el de medición toma los perfiles generados por *ProfileGeneratorThread*. El componente de interfaz, por su parte, muestra los resultados de *ImageAcquisitionThread*, *LaserLineExtractionThread* y *ProfileGeneratorThread*. En los primeros casos, estos resultados se toman de las colas de procesamiento. En el caso de la interfaz, de las de visualización.

7.4.1. Captura

La clase *ImageAcquisitionThread* se encarga de la captura de imágenes. Existe una instancia por cada fuente de imágenes que se utilice.

El componente de calibración emplea directamente los resultados de esta clase, sin pasar por las fases posteriores de la adquisición, y el componente de interfaz los muestra.

Las imágenes capturadas pueden proceder de varios tipos de fuentes distintos:

- Ficheros de imagen: Cada hilo de captura de imágenes utiliza continuamente el mismo fichero preestablecido.
- Ficheros de vídeo: Cada hilo de captura de imágenes va utilizando las imágenes contenidas en un fichero de vídeo.
- Directorios de imágenes: Cada hilo de captura de imágenes va utilizando los ficheros de imagen presentes en un directorio.
- Cámaras: Cada hilo de captura de imágenes emplea la salida de una cámara.

Las tres primeras fuentes se usan principalmente para pruebas, mientras que en el entorno de producción se emplean las cámaras. La fuente utilizada está entre los parámetros de configuración que se especifican en la sección 6.1.

Además, existen cuatro modos diferentes de adquisición:

- Sync: Adquisición síncrona.
- Async: Adquisición asíncrona.
- Async + Callback: Adquisición asíncrona gestionada mediante la ejecución de un *callback* iniciado por el driver de la cámara.
- Async + Callback + Worker: Adquisición asíncrona gestionada mediante la ejecución de un *callback* iniciado por el driver de la cámara. Este *callback* se ejecuta por un hilo independiente para desacoplar la tarea de adquisición.

El modo que se utiliza con mayor frecuencia es Async + Callback dado que proporciona un mejor balance entre rendimiento y uso de recursos del sistema.

Las imágenes obtenidas de las fuentes se encapsulan en objetos *Image*, que incluyen el contenido de la imagen en forma de *HObject* de HALCON, la fuente que la capturó y su número de secuencia, y se almacenan en las colas de imágenes.

7.4.2. Extracción

La clase *LaserLineExtractionThread* toma las imágenes (*Image*) capturadas por el hilo anterior y extrae líneas de ellas. El componente de interfaz muestra directamente los resultados obtenidos.

Para la extracción de líneas se emplean procedimientos de HALCON. Pueden utilizarse tres estrategias distintas:

- Básica: Se emplea el procedimiento *lines_gauss* para extraer líneas de toda la imagen.
- Desenfoque: Para incrementar el rendimiento, se buscan líneas en una región delimitada alrededor de las líneas detectadas en la imagen anterior procesada por el sistema. La región de búsqueda se determina mediante la aplicación de un filtro de desenfoque a la última línea encontrada.
- Dilatación: Para incrementar el rendimiento, se buscan líneas en una región delimitada alrededor de las líneas detectadas en la imagen anterior procesada por el sistema. La región de búsqueda se determina mediante la dilatación de la última línea encontrada.

Típicamente se emplea la estrategia de dilatación, puesto que esta ofrece un rendimiento superior con un consumo de recursos contenido.

Los resultados se almacenan en objetos *LaserImage*, que incluyen las líneas extraídas en forma de *HObject* de HALCON, la fuente, el número de secuencia, etc., y se encolan.

7.4.3. Traducción

La clase *LaserLineTranslationThread* toma las líneas extraídas de las imágenes y las traduce a coordenadas del mundo.

Para llevar a cabo esta traducción han de emplearse los parámetros de calibración generados por el componente de calibración, que se describe en la sección 7.6.

Los resultados se almacenan en objetos *LaserWorld*, que son similares a los *LaserImage* salvo que almacenan las líneas en coordenadas del mundo, y se encolan.

7.4.4. Generación de perfil

La clase *ProfileGenerationThread* toma las líneas obtenidas por cada fuente y puestas en el mismo sistema de coordenadas y las combina para generar una única nube de puntos.

La salida de esta clase la emplean tanto el componente de interfaz como el de medición.

El resultado se almacena en un objeto *Profile*, que contiene la nube de puntos en forma de *PointCloud* de *Geometry*.

7.5. Medición

El componente de medición es el encargado de procesar la salida del componente de adquisición para obtener un conjunto de valores de las dimensiones.

La salida del componente adquisición como se comenta en la sección 7.4 es un conjunto de puntos, o nube de puntos, que forman un contorno de un perfil de carril. Esta nube de puntos debe ser transformada en un conjunto de primitivas que formen un modelo de carril, en el cual se podrán realizar las mediciones.

Por lo tanto, este componente puede ser dividido a su vez en tres partes más pequeñas:

- *Alineamiento*, proceso por el cual cada punto de la nube de puntos recibida del componente adquisición pasa a estar alineado de acuerdo al modelo sobe el cual se realiza la medición. Esto se explica con mayor detalle en la sección 7.5.1.
- *Fitting* o ajuste, proceso por el cual la nube de puntos se ajusta a un conjunto de primitivas (arcos y segmentos) extraídas del modelo, convirtiéndose en primitivas ajustadas. Esto se explica con mayor detalle en la sección 7.5.2.
- *Cálculo dimensional*, proceso por el cual una vez ajustadas todas las primitivas del modelo del carril, se calculan las dimensiones del mismo. Esto se explica con mayor detalle en la sección 7.5.3.

En la Figura 42 se puede observar el diagrama de clases donde se divide el componente de medición en las tres partes antes comentadas.

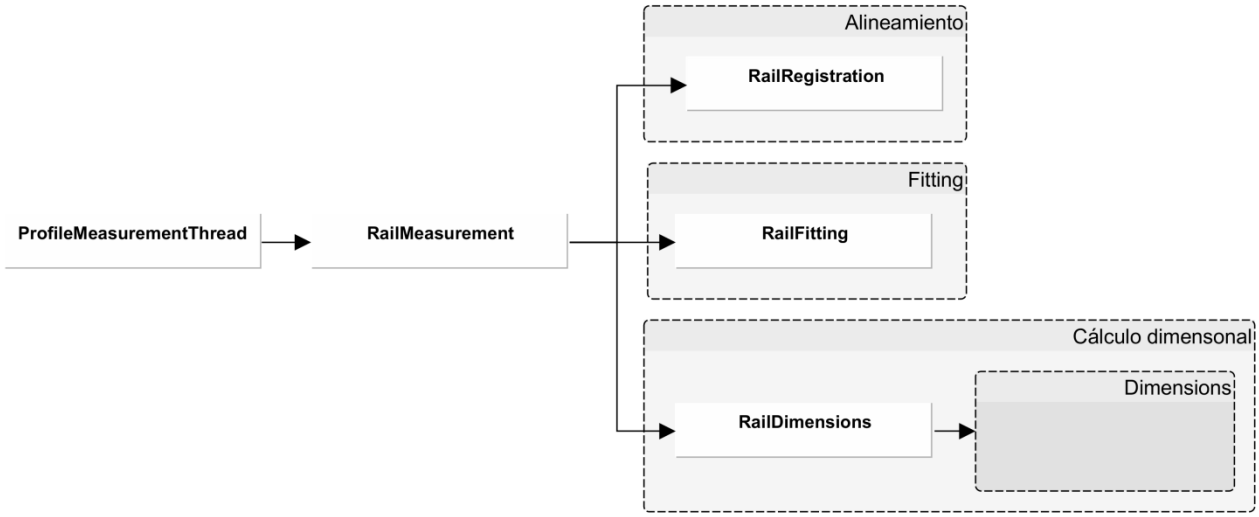


Figura 42: Diagrama de clases del componente de medición

Al constructor de la clase *RailMeasurement* se le ha de pasar un modelo de carril, este modelo será el que se va a utilizar para llevar a cabo el alineamiento y el fitting sobre él.

Esta clase *RailMeasurement* implementa un método llamada *CalculateDimensions*. A esta función se le ha de pasar como parámetro una nube de puntos que contenga el contorno del carril al cual se van a calcular las dimensiones. A esta nube de puntos se le realizará un alineamiento y un fitting como se explicará en las siguientes secciones. Una vez realizado esto

se obtendrá un modelo a partir de la nube de puntos que contendrá las primitivas del carril aproximadas, a las cuales se les puede calcular todas las dimensiones, obteniendo así el resultado para ese perfil.

7.5.1. Alineamiento

Mediante esta técnica la nube de puntos que conforma el contorno del perfil del carril pasa a estar en la misma posición que el modelo del carril que se utiliza para llevar a cabo la medición. Para ello se ha de llevar a cabo una transformación geométrica.

Una transformación geométrica está compuesta por dos partes:

- Translación, es decir, movimientos directos de todos los puntos sin cambios de orientación. Esta mantiene la forma y el tamaño de la nube de puntos.
- Rotación, es decir, hacer que todos los puntos de la nube de puntos roten alrededor de un punto central, donde este punto no tiene por qué ser de la propia nube.
- Escalado, esta parte de la transformación no se utiliza en el sistema. Esta parte de la transformación aumentaría o disminuiría la distancia entre puntos.

La Figura 43 muestra una transformación realizada sobre una nube de puntos extraída del modelo de carril.

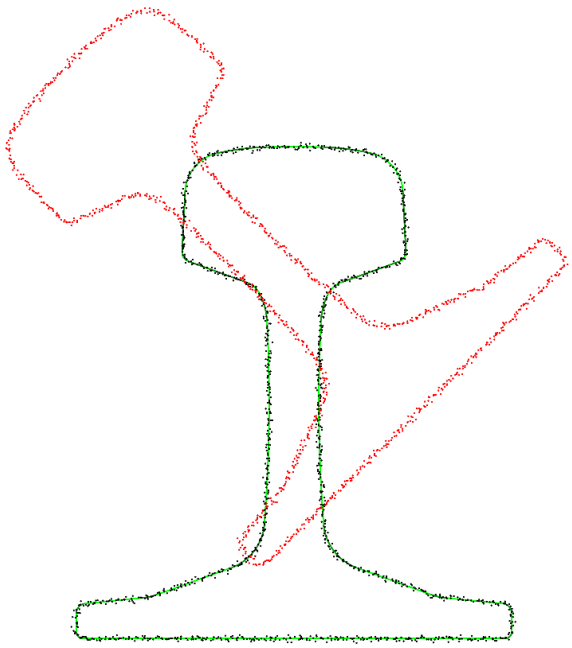


Figura 43: Alineamiento de una nube de puntos a un modelo

En este caso la nube de puntos obtenida del proceso de adquisición de imagen se muestra en color rojo, mientras que la nube de puntos obtenida tras aplicar la transformación se muestra en color negro. En color verde se muestra el modelo de carril sobre el que se realiza la medición, es decir, el modelo de carril al que se ha ajustado la nube de puntos obtenida.

Se ha creado, con la finalidad de realizar transformaciones geométricas sobre nubes de puntos, la clase *Transformation*. Esta clase tiene los siguientes métodos:

- *AddTranslation*, *AddRotation* y *AddScale* (no se utiliza). Añade la translación, rotación o escalado a la clase matriz de transformación de la clase *Transformation*.
- *Transform*, a esta función se le ha de pasar un punto al cual se le ha de aplicar dicha transformación geométrica.
- *EstimateRigidTransform*, a esta función se le ha de pasar la nube de puntos del modelo, así como la nube de puntos transformada (que se ha obtenido mediante el proceso de adquisición explicado en la sección 7.4). Esta función devuelve una instancia de clase *Transform*, que contiene la transformación que hay que realizar a la nube de puntos del modelo para obtener la nube de puntos modificada.

Para llevar a cabo el proceso de alineamiento se utiliza una técnica llamada *ICP* o *Iterative Closest Point*. Mediante esta técnica se consigue la transformación que aplicada sobre la nube de puntos medida proporciona esta nube alineada al modelo.

Esta nube de puntos alineada sirve de entrada a la siguiente fase del componente de medición.

7.5.2. Fitting o ajuste

Este es el proceso por el cual una nube de puntos determinada se ajusta a una primitiva o conjunto de primitivas determinada y se ha de realizar sobre el resultado obtenido en el proceso anterior de alineamiento de la sección 7.5.1.

Para llevar a cabo la aproximación de una nube de puntos a una primitiva, primeramente hay que seleccionar que puntos de la nube general del contorno del carril pertenecen a una primitiva o a otra. Para ello se crea alrededor de la primitiva una región en la cual se admitirán los puntos que estén en ella como pertenecientes a esa primitiva en concreto.

Esta región se conoce como *Envelope*. La figura Figura 44 muestra el uso del *Envelope* en un arco.

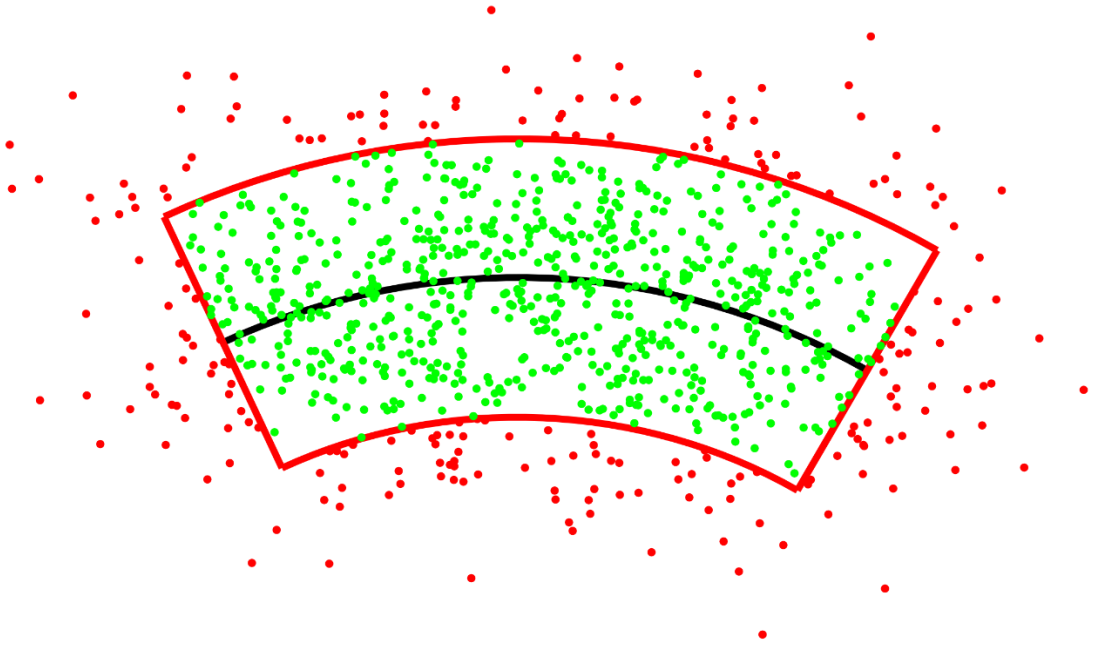


Figura 44: Uso del *Envelope* para el caso de un arco

Los puntos mostrados en verde de la Figura 44 serán los que posteriormente se utilizarán para aproximar a la primitiva. Por otro lado los puntos en rojo, es decir, los que están fuera del *Envelope* del arco, no serán utilizados para realizar el *fitting* o aproximación

Este proceso de comprobación del *Envelope* se ha de realizar para cada una de las primitivas del carril. Una vez se han determinado que puntos pertenecen a cada primitiva se procede a la realización de la aproximación.

Con la intención de minimizar el error, las primitivas a las que se aproximará no serán arcos y segmentos, si no círculos y líneas, minimizando así el error ya que no se definen extremos.

La Figura 45 muestra un *fitting* realizado a una nube de puntos y que se ha aproximado por un círculo en la parte izquierda, y un fitting realizado a una nube de puntos que se ha aproximado por una línea en la parte derecha.

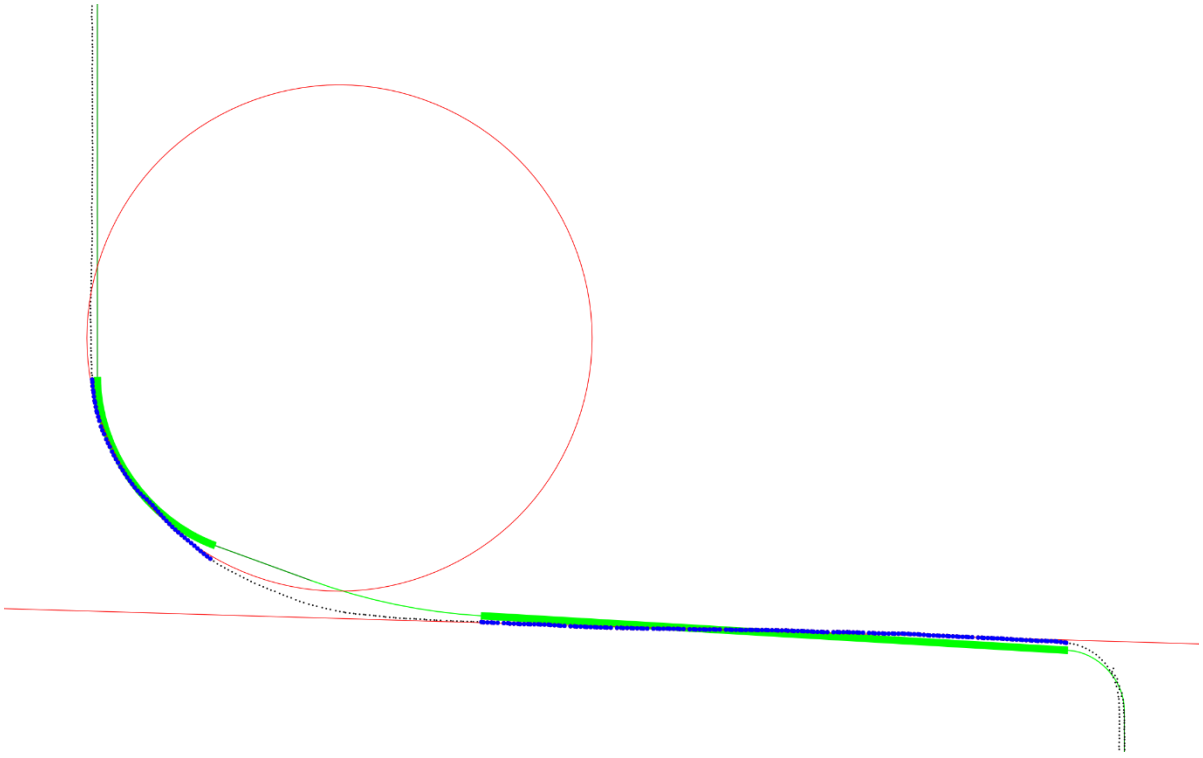


Figura 45: Fitting de una nube de puntos

En la Figura 45 los elementos aproximados se muestran de color rojo, mientras que las primitivas del modelo se muestran en color verde, así como los puntos que están dentro del *Envelope* de las primitivas se muestran en color azul.

Para llevar a cabo todo este proceso se ha implementado la clase *RailFitting* esta clase lleva a cabo el proceso de *fitting* de una nube de puntos al modelo de carril, lo cual se realiza mediante la función *Fit*, a la cual se le debe pasar como parámetro una nube de puntos, ya que el modelo deberá haber sido pasado como parámetro en el constructor de la propia clase.

Esta clase utiliza además una variable *EnvelopeOffset* para definir el tamaño del *Envelope* a crear en cada primitiva.

La función *Fit* antes comentada, devuelve un modelo de carril que contendrá las primitivas del carril aproximadas, y que servirán de entrada al proceso que se explica en la sección 7.5.3.

7.5.3. Cálculo dimensional

Este proceso tiene como entrada un modelo de carril que ha sido aproximado y es la salida del proceso descrito en la sección 7.5.2.

Para llevar a cabo el cálculo dimensional se ha creado una clase llamada *RailDimensions*. Esta clase contiene básicamente una lista de objetos *Dimension*, como los que se puede ver en la Figura 46. Esta clase tiene también un método llamado *Calculate* a la cual se le pasa por referencia un modelo de carril (la salida del proceso anterior), y que devuelve un diccionario con los nombres de las dimensiones y sus valores para esa sección del carril.

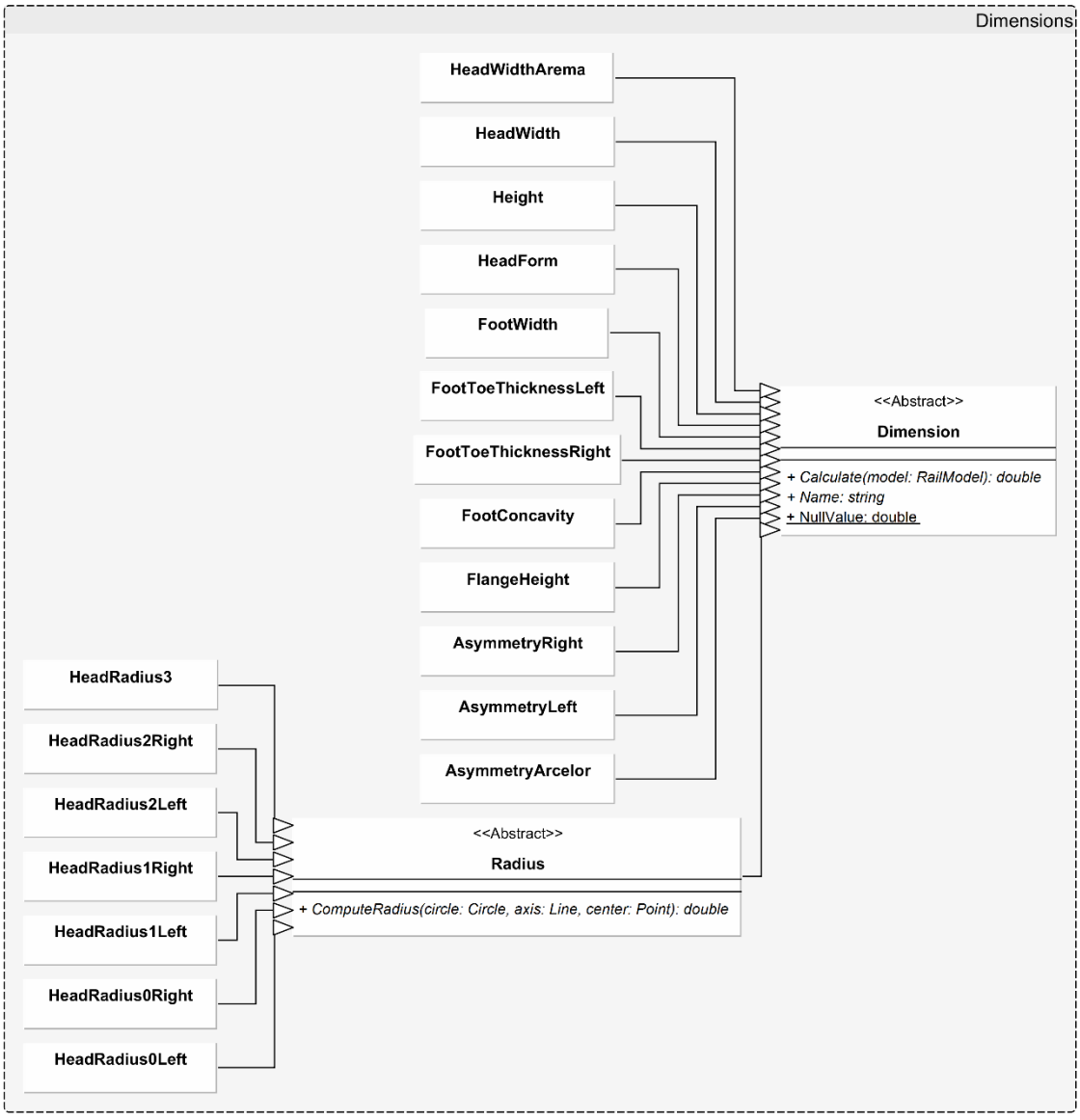


Figura 46: Diagrama de clases de dimensiones del proyecto Geometry

La clase abstracta *Dimension*, contiene un método *Calculate* que toma un modelo de carril y devuelve un *double.* Este método deberá ser implementado por todas y cada una de las clases que heredan de *Dimension*.

Esta clase abstracta contiene además la propiedad *Name* de tipo *string,* la cual contendrá en cada una de las clases que hereden de *Dimension*, el nombre específico de esa dimensión.

Por último esta clase abstracta contiene un campo estático llamado *NullValue*, el cual será asignado a las dimensiones que no puedan ser calculadas.

El algoritmo seguido para llevar a cabo el cálculo de todas y cada una de las dimensiones queda fuera del ámbito de este TFM.

7.6. Calibración

El algoritmo de calibración, que se describe en el documento V - Calibración, se implementó en primer lugar en HDevelop (véase la sección 7.8.1.2) y se exportó a C#.

El código C# generado, formado por una serie de métodos estáticos, uno por cada procedimiento definido en HDevelop, se refactorizó para facilitar su comprensión y mantenimiento.

Puesto que los métodos estáticos generados tomaban un gran número de parámetros, se añadieron varias propiedades correspondientes a los parámetros de configuración de la calibración (algoritmo empleado, número de puntos, etc.) y se crearon métodos de instancia correspondientes a los estáticos pero que tomaban menos parámetros y utilizaban los valores de las propiedades en su lugar.

El resultado final es una clase *Calibration* cuyas instancias representan objetos de calibración con determinados parámetros, y que contienen un método *CalibrateCamera* que toma una imagen, una lista de coordenadas y radios de los cilindros de la plantilla, un ángulo de inclinación aproximado y una lista de parámetros intrínsecos de la cámara y devuelve los parámetros extrínsecos calculados. También incluye métodos para el cálculo del error de calibración.

Al realizar la calibración, el método *CalibrateCamera* se llama una vez por cada una de las cuatro cámaras:

- Los parámetros de configuración toman el valor que se especifique en la configuración del sistema.
- La imagen utilizada es la última capturada (*Adq. imagen*) para la cámara de la que se trate.
- Las coordenadas de los cilindros presentes en la plantilla de calibración y sus radios se especifican en un fichero de configuración (por defecto denominado *CircularCalTarget.txt*), que incluye una línea por cada cilindro, con el siguiente formato:

  X Y Radio

  Donde la X y la Y son relativas al centro de la plantilla. Por ejemplo:

  15.7264 -149.9195 11.0293

  71.7691 -112.9950 11.0332

- El ángulo de inclinación depende de la cámara de la que se trate y va en incrementos de 90º: 45º, 135º, -45º, -135º.
- Los parámetros intrínsecos de cada cámara han de estar calculados de antemano y especificarse en la configuración del sistema. A diferencia de los parámetros extrínsecos, estos no varían con el tiempo si no cambian la cámara ni su lente.

7.7. Ejecución

El punto de entrada al programa es el método *Main*. Éste carga la configuración del sistema y a continuación muestra el formulario principal (*FrmMain*). El formulario principal espera entonces hasta que el usuario escoge una opción.

Si el usuario elige modificar la configuración o la contraseña de administrador, desde el formulario principal se abre el diálogo correspondiente. Las modificaciones que se hagan en esta configuración se almacenan en un fichero tan pronto como se cierran los formularios correspondientes.

Cuando el usuario activa el modo de medición, se construye el controlador. Este se encarga de activar las comunicaciones con el PLC y el ordenador de proceso (a través de un proceso RailCommunications, que se crea automáticamente en este momento si así se establece en la configuración), y de construir el *pipeline* descrito en la sección 7.2.1. Una vez activado el modo de medición, se inicia también un hilo responsable de atender conexiones TCP y servir los resultados de las mediciones, al que se conecta el programa visualizador (parte de la interfaz).

La construcción del *pipeline* consiste en la creación de los hilos y las colas que lo conforman. Los hilos se construyen, inician y luego destruyen en orden, desde el final del *pipeline* hasta el comienzo. Una vez construido, arranca la ejecución de todos los hilos. Los hilos ocupados directamente de la captura de imágenes (*ImageAcquisition*) quedan bloqueados esperando un evento procedente del controlador. Cada uno de los restantes hilos espera un evento procedente del hilo anterior. De este modo, el Controller puede establecer cuándo se adquieren imágenes.

Una vez activado el modo de medición, no hay actividad hasta que se recibe del PLC un evento *RailEnter* o el usuario lo genera manualmente. Este evento marca el comienzo de un carril. En el momento en que se recibe, el controlador envía un evento a los hilos de adquisición de imágenes.

Cada hilo de adquisición captura las imágenes procedentes de una cámara y las inserta en dos colas, una para el procesamiento y otra para la visualización de las imágenes. Al insertarlas, también disparan eventos que provocan la activación de los hilos siguientes, que son los encargados de extraer líneas de las imágenes. Cada uno de estos hilos obtiene las imágenes procedentes de la cola de procesamiento de imágenes de una de las cámaras, extrae las líneas que aparecen en esas imágenes y las inserta en otras dos colas, una para el procesamiento de las líneas y otra para su visualización. Del mismo modo se activan entonces los hilos que toman las líneas en coordenadas de la imagen, las traducen a coordenadas del mundo y las introducen en otras colas, una para cada cámara. Cada uno de estos dispara un evento al introducir un elemento en una de las colas. Entonces, un único hilo comprueba si las colas de todas las cámaras están llenas y si las líneas que contienen proceden de imágenes tomadas en el mismo momento. Si es así, extrae las líneas procedentes de las cuatro cámaras y las combina en un único perfil, que pasa a una cola de procesamiento y a otra de visualización.

El siguiente hilo del *pipeline*, *ProfileMeasurement*, toma los perfiles ya completos y mide los valores de sus dimensiones. Los resultados se encolan y pasan a otro hilo, *ProfileOutput*, que los comunica a la interfaz.

Mientras tanto, en el hilo principal, el formulario principal (clase *FrmMain* del componente de interfaz) toma los resultados intermedios, obtenidos de las colas de imágenes, líneas y perfiles, y los resultados finales de la medición, obtenidos a través de *ProfileOutput*, y los muestra al usuario. Los resultados finales se sirven también a los clientes de visualización que estén conectados.

Cuando llega (del PLC o generado por el usuario) el evento *RailExit*, el controlador resetea el evento que controla los hilos de captura de imágenes, de modo que se van vaciando las colas. Una vez vacías las colas, *ProfileOutput* marca el carril como terminado, con lo que el componente de interfaz almacena los resultados en un fichero y comunica el final del carril a los clientes de visualización.

En el caso del modo de calibración, el controlador y el *pipeline* se crean del mismo modo, pero es el usuario el que debe solicitar la adquisición de imágenes. Cuando lo hace, la interfaz se lo comunica al componente de control, que a su vez activa el hilo de adquisición, que toma una única imagen. Es el controlador el que llama al componente de calibración pasándole estas imágenes y los valores de configuración necesarios. El componente de calibración extrae los parámetros extrínsecos de las cámaras y las devuelve al componente de control. Una vez confirmados por el usuario, se almacenan.

7.8. Bibliotecas

La solución del programa medidor está compuesta por una serie de proyectos que le aportan distintas funcionalidades.

La Figura 47 muestra la relación de todos los proyectos que conforman la solución del medidor.

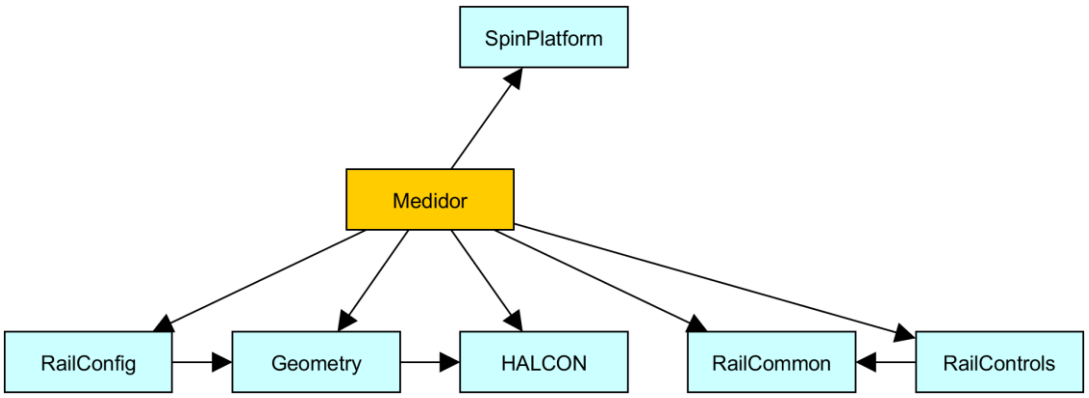


Figura 47: Relación de los proyectos de la solución RailMeter

El proyecto de librería *RailConfig* contiene todas las clases y elementos necesarios para cargar la configuración que se ha creado en el configurador sobre normas y perfiles. Esta configuración va a ser utilizada, como se ha comentado previamente, para comparar los resultados obtenidos en la medición con los modelos esperados, y comprobar así el cumplimiento de la calidad. Esta biblioteca queda fuera del ámbito de este TFM.

El proyecto de librería *Geometry* actúa como soporte para todas las operaciones de geometría que se realizan en el medidor. *Geometry* proporciona también una serie de estructuras de datos utilizadas para almacenar los perfiles de los carriles, con sus segmentos, arcos nubes de puntos, etc.

El proyecto HALCON está formado por una serie de librerías creadas por la empresa MVTec. HALCON es un sistema de visión artificial, el cual ayuda a la extracción de las líneas láser de las imágenes, así como de llevar a cabo diferentes tipos de tareas en la calibración, etc.

La biblioteca *RailCommon* contiene la estructura básica de la funcionalidad de visualización, el modelo y las interfaces que tienen que implementar las vistas y los controladores.

La biblioteca *RailControls* contiene los componentes gráficos utilizados para la visualización. Depende de *RailCommon*.

El proyecto *Spin* contiene una plataforma desarrollada por el departamento homónimo de *ArcelorMittal*. Esta plataforma está formada por un conjunto de DLLs que buscan homogeneizar todo los proyectos que se realicen en dicho departamento. Este proyecto cuenta con una serie de funcionalidades amplias, que abarcan desde la creación de hilos, compartición de memoria, etc.

7.8.1. Bibliotecas externas

7.8.1.1. Plataforma SPIN de ArcelorMittal

Para llevar a cabo la implementación del sistema, se va a utilizar una plataforma que ha sido creada y está suficientemente probada y es robusta, que se ha implementado en ArcelorMittal y que hace que las tareas de creación de hilos, compartición de memoria y gestión de recursos, entre otras, sean más sencillas de llevar a cabo.

SPIN consta de una serie de bibliotecas enlazadas dinámicamente (DLL), para realizar tareas automatizadas, cuenta con una serie de funcionalidades que abarcan la creación de hilos, creación y establecimiento de comunicaciones entre procesos y a través de la red, gestión y uso de sensores, etc.

Estas funcionalidades están divididas en varias DLL, de entre las cuales se van a emplear las siguientes en este proyecto:

- **SpinPlatform_common:** Biblioteca principal en la que se incorpora el *SpinDispatcher*, la parte que gestiona todos los recursos e hilos de Spin¸ por lo que deberá estar presente en todos los proyectos en los que se use esta plataforma, aquí se incluye además el módulo de configuración, comunicaciones, errores y todo el conjunto de estructura de datos.
- **SpinPlatform_IO:** Aquí se encuentran los módulos de comunicación con bases de datos y de generación de consultas, un módulo para la creación de logs, así como un módulo para envío de emails y otro para comunicaciones FTP.

7.8.1.2. HALCON

HALCON es una plataforma de visión artificial que permite realizar un conjunto de operaciones con imágenes de forma precisa y eficiente. También puede interactuar directamente con una amplia gama de dispositivos de captura de imágenes.

HALCON se divide en dos componentes:

- Un entorno de desarrollo, denominado HDevelop, que puede emplearse para ejecutar procedimientos de HALCON y obtener resultados en tiempo real y de forma interactiva. Emplea un lenguaje de programación propio, llamado simplemente “lenguaje de HDevelop”, pero permite traducir automáticamente código de este lenguaje a otros, incluyendo C#, C++ y Visual Basic, y exportarlo. Este entorno se empleó para desarrollar la funcionalidad de calibración y realizar algunas pruebas.
- Una DLL independiente que contiene implementaciones de los procedimientos que forman parte de la plataforma y permite emplearlos desde otros lenguajes y plataformas. El programa medidor depende de esta DLL para su ejecución.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

DOCUMENTO IV

COMUNICACIONES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Introducción al sistema de comunicaciones

El medidor no puede funcionar correctamente de forma aislada, sino que necesita conocer su entorno y, en menor medida, modificarlo: debe conocer cuándo entran y salen carriles y de qué tipos son; debe ser capaz de desplegar y replegar la plantilla de calibración, etc.

Para gestionar la entrada y la salida, y al mismo tiempo aislar al medidor de su entorno, se implementa un sistema de comunicaciones independiente. Este sistema es el responsable de gestionar toda la comunicación entre el medidor y otros equipos: el ordenador de proceso (PA) y el PLC que indica dónde comienzan y terminan los carriles.

Más específicamente, el sistema de comunicaciones establece conexiones con dichos equipos externos, transmite al medidor sus peticiones y remite los mensajes del medidor al equipo que corresponda, dependiendo del tipo de mensaje.

De este modo, el medidor ya no es responsable de comunicarse con varios equipos externos diferentes, sino simplemente con el sistema de comunicaciones creado específicamente para esa tarea, y que puede modificarse en caso de cambiar los equipos externos, sin tener que modificar el código del medidor en sí.

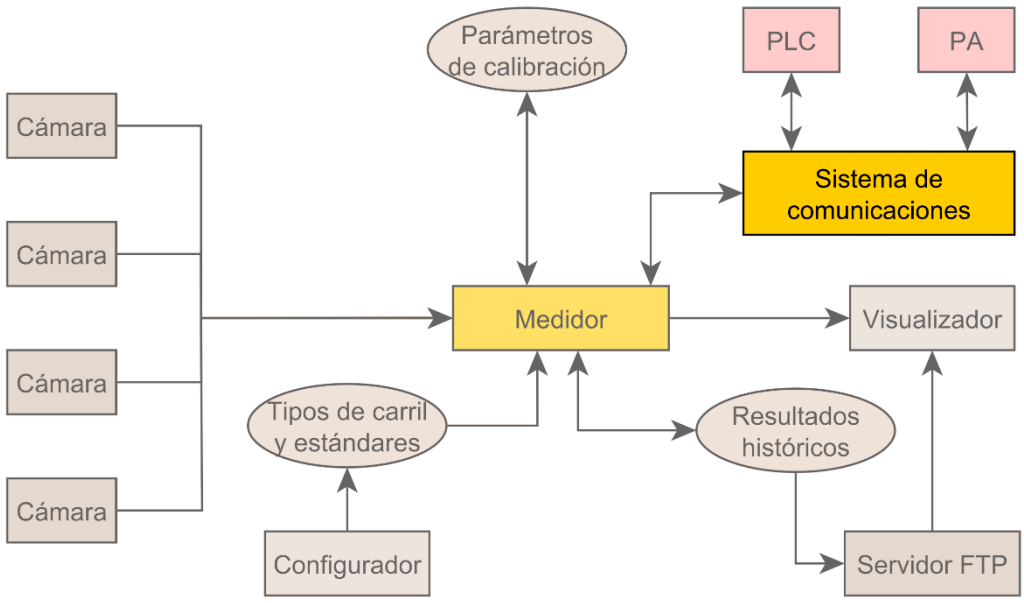


Figura 1: Diagrama del sistema, mostrando solamente los elementos relacionados con el sistema de comunicaciones

La Figura 1 y la Figura 2 muestran la relación del sistema de comunicaciones con los otros elementos.

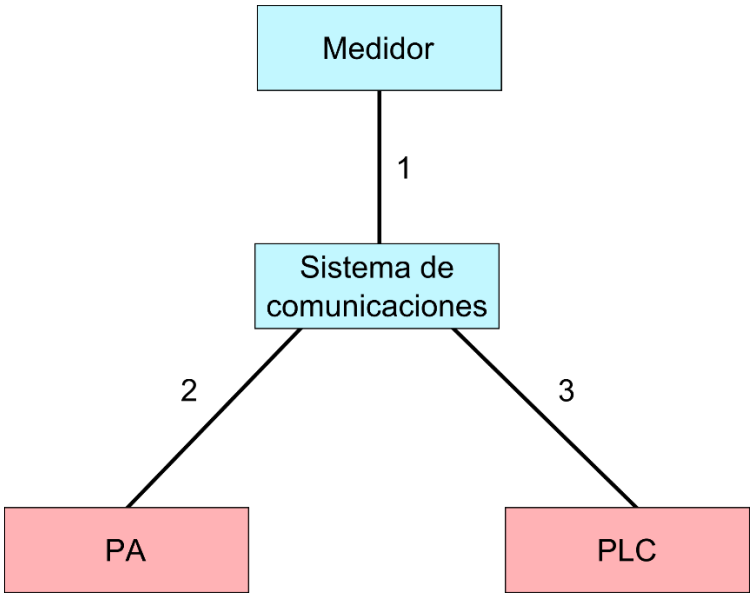


Figura 2: Estructura de las comunicaciones (en rojo los equipos externos; en azul los pertenecientes al equipo de medición)

Para poder llevar a cabo pruebas con las comunicaciones en el laboratorio se han implementado dos emuladores que se encargan de generar los mensajes tal y como lo harían los elementos reales de los que no se dispone en el laboratorio, el PA y el PLC. Estos emuladores deben seguir estrictamente los protocolos de comunicaciones definidos a priori para establecer la secuenciación de mensajes. Dichos protocolos serán los usados en un futuro cuando el sistema se ponga en producción.

1.1. Tecnologías de comunicación

1.1.1. TCP/IP en .NET

En la biblioteca de clases de .NET (.NET Framework Class Library), la responsable en última instancia de las comunicaciones de red es la clase Socket (System.Net.Sockets.Socket). Esta clase, como su nombre sugiere, ofrece una interfaz a bajo nivel para el trabajo con sockets de red.

Por ello, existen también otras clases de más alto nivel que ofrecen una interfaz más amigable a las operaciones con sockets. Una de ellas es NetworkStream (System.Net.Sockets.NetworkStream), que trata las comunicaciones de red como un flujo de datos.

Puesto que NetworkStream hereda de la clase Stream (System.IO.Stream), los flujos con los que trabaja pueden tratarse igual que los procedentes de ficheros (FileStream) o de memoria (MemoryStream).

La mayor parte de los métodos de ambas clases para la realización de operaciones síncronas (Read, Write, Receive, etc.) cuentan también con versiones asíncronas[1] (BeginRead y EndRead,

[1] En .NET 4.5 se añadieron métodos con nombres terminados en *Async* pensados para utilizarse con la interfaz *async*/*await*. Sin embargo, no pueden emplearse en este proyecto ya que ha de ser compatible con la versión 4.0.

etc.) que permiten especificar una función a la que se llamará tras completar la operación correspondiente. Esto evita tener que gestionar manualmente los hilos, ya que en este caso es .NET quien gestiona los que sean necesarios.

1.1.2. WCF

Windows Communication Foundation, WCF, es una API para la creación de aplicaciones orientadas a servicios. Forma parte del framework .NET.

Se trata de un mecanismo de llamada a procedimientos remotos (RPC), y su funcionamiento es similar al de RMI en Java, por ejemplo. La idea general es declarar una interfaz en el servidor, que contiene los métodos que se ofrecen, e implementarla; y en el cliente obtener una referencia al servicio (que puede "autodescubrirse") y utilizar directamente la interfaz para consumir la funcionalidad que el servidor proporciona.

Además de esta comunicación en una dirección, WCF también ofrece posibilidades de comunicación "dúplex". En la comunicación dúplex, además de la interfaz del servidor ofrecida por el servidor, puede definirse una segunda interfaz que el cliente debe implementar, de modo que el servidor pueda emplearla una vez establecida la conexión.

WCF se encarga de gestionar y abstraer los aspectos de bajo nivel de la comunicación entre el cliente y el servidor, que puede emplear distintos enlaces o "bindings". Entre los "bindings" disponibles por defecto existen varios basados en HTTP (con SOAP o REST), y otros en protocolos específicos de .NET.

2. Protocolos de comunicaciones

2.1. Comunicación con el PA

El sistema de comunicaciones ha de conectarse al PA ("Process Automation", el ordenador de proceso) para recibir la información identificativa de los carriles que se van a medir y para enviar los resultados de las mediciones para que pueda dárseles uso.

Esta comunicación se realiza a través de un protocolo propio sobre TCP. La especificación completa de este protocolo, incluyendo el formato general de los mensajes y el contenido de cada uno, se describe en el anexo D - Comunicación con el PA.

Estos mensajes están compuestos por cadenas de longitud variable para la comunicación entre ambos extremos. El PA actúa como servidor y el sistema de comunicaciones como cliente.

Los mensajes empleados en este protocolo son los siguientes:

- **Life Message**, mensaje que envía periódicamente el PA al sistema de comunicaciones. El sistema de comunicaciones espera recibirlo cada 120 segundos como máximo.
- **Confirmation Life Message**, mensaje de respuesta al mensaje anterior, del sistema de comunicaciones al *PA*.
- **New rail data**, información del siguiente carril a medir, del PA al sistema de comunicaciones.
- **Confirmation of new rail data received**, confirmación del mensaje anterior, del sistema de comunicaciones al PA.
- **Last rail measurement results**, conjunto de las mediciones realizadas al último carril medido, del sistema de comunicaciones al PA.
- **Confirmation of reception of last rail measurement results**, confirmación del mensaje anterior, del PA al sistema de comunicaciones.
- **Request one rail measurement results**, petición de las mediciones de un carril que ha sido medido previamente, del PA al sistema de comunicaciones.
- **One rail measurement results,** respuesta a la petición anterior con las mediciones del carril solicitado, de sistema de comunicaciones al PA.
- **MV Status**, estado actual en que se encuentra el medidor. Enviado por el sistema de comunicaciones al PA.

2.2. Comunicación con el PLC

El sistema de comunicaciones ha de conectarse al PLC para poder controlar cuándo se despliega o se repliega la plantilla de calibración, y para conocer cuándo comienzan y terminan los carriles.

Los mensajes que se pueden enviar entre el PLC y el sistema de comunicaciones han sido definidos previamente en un protocolo, que se describe en el anexo C - Comunicación con el PLC. Como en el caso del protocolo para la comunicación con el PA, este se utiliza directamente sobre TCP.

Cada uno de los mensajes, en ambos sentidos, tiene un tamaño de 1 byte. El PLC opera como servidor mientras que el sistema de comunicaciones es el cliente.

Este protocolo constará de una serie de 4 mensajes, los cuales serán:

- Mensaje **Hello**: mensaje de inicio de sesión, en ambos sentidos.
- Mensaje **Enter**: entra un nuevo carril en la zona de medición, del PLC al sistema de comunicaciones.
- Mensaje **Exit**: el carril actualmente en la zona de medición la abandona, del PLC al sistema de comunicaciones.
- Mensaje **Deploy_CalibrationTemplate**: solicitud de que se despliegue la plantilla de calibración, del sistema de comunicaciones al PLC.
- Mensaje **Retract_CalibrationTemplate**: solicitud de que se recoja la plantilla de calibración, del sistema de comunicaciones al PLC.

2.3. Comunicación con el medidor

Entre el sistema de comunicaciones y el medidor se emplea una interfaz basada en la tecnología Windows Communication Foundation (WCF).

No existe ningún requisito de seguridad, dadas las circunstancias (el entorno en el que se ejecuta y el hecho de que el sistema de comunicaciones es meramente un adaptador, una "fachada" que de todas formas ha de conectarse de forma insegura a otros sistemas), lo que simplifica la configuración notablemente.

Sin embargo, sí es necesario emplear una comunicación en ambos sentidos: por ejemplo, el sistema de comunicaciones debe comunicar al medidor cuándo comienza y termina un carril, mientras que el medidor ha de señalarle cuándo debe desplegarse y retirarse la plantilla de calibración. Esto requiere que ambos puedan enviar mensajes en cualquier momento. Puesto que HTTP no permite esto por sí mismo (el servidor se limita a responder a las peticiones del cliente), no puede emplearse como canal de la comunicación ("enlace" o "binding" en la terminología de WCF). Debe emplearse un tipo de enlace "dúplex".

Entre los enlaces dúplex que proporciona WCF existe uno basado también en HTTP, pero que emplea una conexión distinta (cliente y servidor) en cada sentido. El uso de este enlace resulta impráctico.

Por otra parte, existe un enlace alternativo, denominado net.tcp, que consiste en el uso de un protocolo desarrollado especialmente para WCF. Además de proporcionar mayor rendimiento (lo cual no es particularmente importante en este caso), permite la comunicación en ambos sentidos sin necesidad de modificar la configuración por defecto. Por tanto, es este el que se utiliza.

La interfaz finalmente empleada es la siguiente:

Los métodos que expone el medidor (que actúa como servidor):

- `bool SetRailHeader(RailHeader header);`

Permite especificar la cabecera (identificador, tipo y norma empleada) del siguiente carril que se va a medir. Devuelve true si el medidor reconoce el tipo y la norma, y false si no reconoce alguno de los dos.

- `string GetRailMeasurement(string id);`
  Permite obtener el resultado de la medición de un carril concreto, especificando su identificador. Si existe un carril con ese identificador, devuelve el contenido del fichero de medición correspondiente. Si no existe, devuelve una cadena vacía.

- `void RailEnter();`
  Señala la entrada de un nuevo carril.

- `void RailExit();`
  Señala la salida de un carril.

- `void ConnectionStatusChanged(bool paConnected, bool plcConnected);`
  Informa de un cambio en la disponibilidad del ordenador de proceso (*paConnected*) o del PLC (*plcConnected*). Para cada uno de los dos, el valor true indica que está conectado mientras que false indica que no lo está.

- `void Register();`
  Registra el cliente y permite que reciba las llamadas correspondientes de la otra interfaz.

Los métodos que expone el sistema de comunicaciones (que actúa como cliente; se trata de una interfaz que se denomina de “callback”):

- `void StartCalibration();`
  Despliega la plantilla de calibración.

- `void EndCalibration();`
  Repliega la plantilla de calibración.

- `void SendLastRailMeasurement(string id, string data);`
  Recibe los resultados de la última medición realizada (llamado por el medidor al completar la medición de un carril).

- `void ServerClosed();`
  Notifica que el medidor se cerrará inmediatamente.

3. Implementación

3.1. Emulador del PA - *PAServer*

El emulador del ordenador de proceso, llamado *PAServer*, es un programa que actúa como servidor en la comunicación con *RailCommunication*, permitiendo emplearlo fuera del entorno de producción en el que está presente el ordenador de proceso. Este emulador incluye una interfaz gráfica para elegir de forma interactiva los mensajes que se envían al sistema de comunicaciones.

Para su implementación se ha empleado el patrón State: se han definido varios estados independientes con distintos comportamientos, que se utilizan en distintas etapas de la comunicación. Esto simplifica la adición de nuevos estados que puedan ser necesarios en el futuro.

En la Figura 3 se muestra el diagrama de clases de *PAServer*, en el que puede observarse claramente la implementación de dicho patrón.

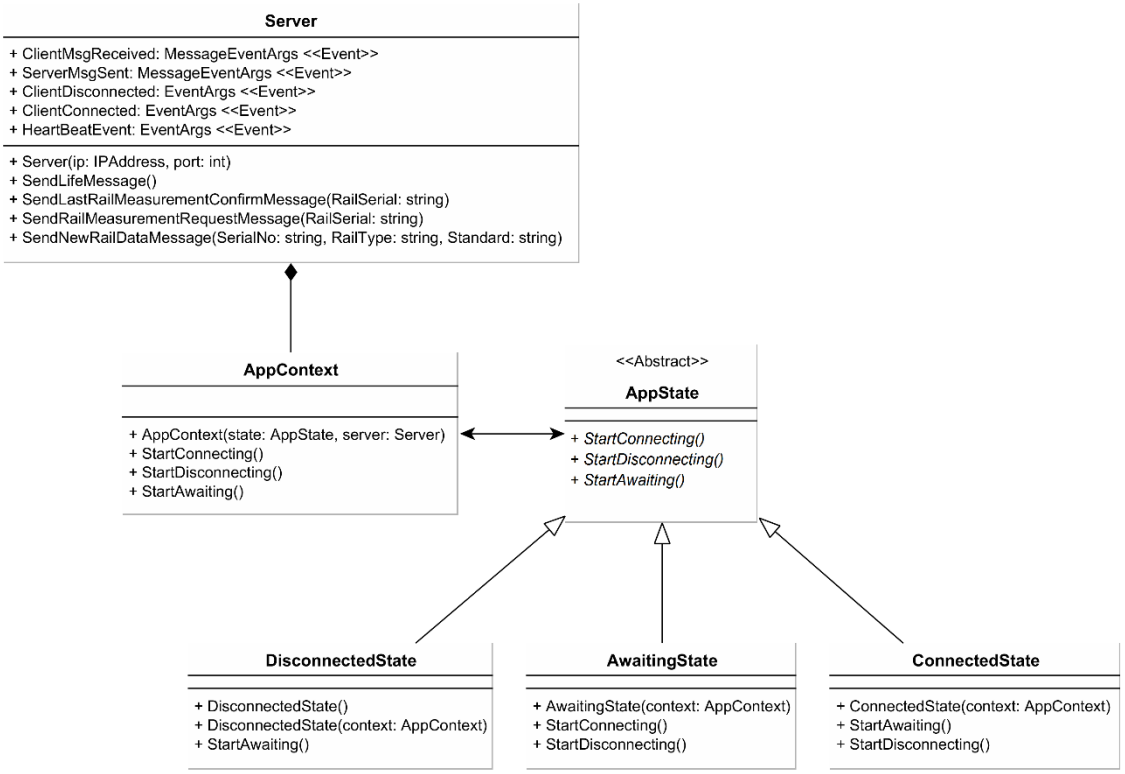


Figura 3: Diagrama de clases para el PAServer

El estado inicial del servidor es el estado *Disconnected*. Este es un estado meramente de transición desde el que se pasa al estado *Awaiting*, en el que el servidor espera conexiones entrantes. Una vez establecida una conexión, el servidor pasa al estado *Connected*, en el que se mantiene mientras dure esta. Cuando la conexión se corta, vuelve al estado *Awaiting*. Cuando el servidor ha de cerrarse, pasa de cualquiera de esos dos estados al estado *Disconnected*. Esta estructura puede verse en la Figura 4.

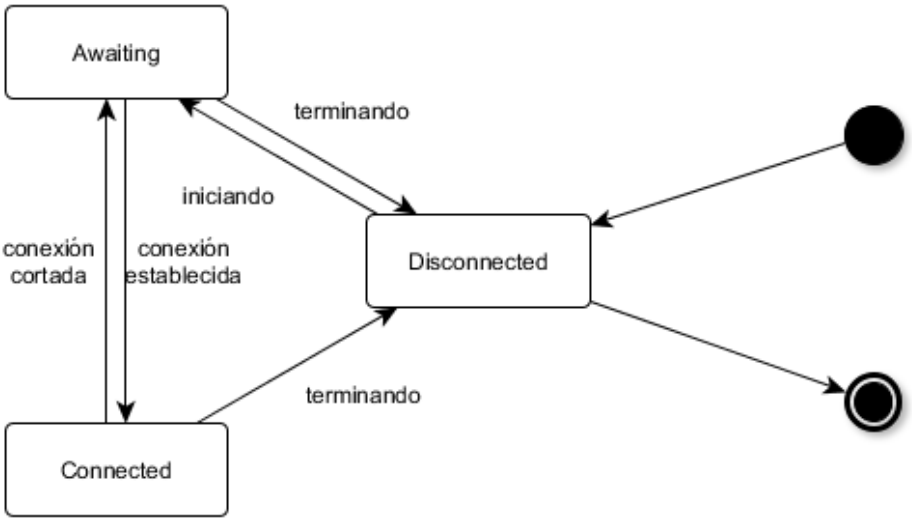


Figura 4: Diagrama de estados de *PAServer*

Todas las comunicaciones, así como el establecimiento de la conexión, se realizan mediante operaciones asíncronas (véase la sección 1.1.1).

3.1.1. Implementación de los mensajes

Para cada uno de los mensajes definidos en la especificación se implementa una clase responsable de serializarlo y deserializarlo. El diagrama de clases se muestra en la Figura 5.

Para la serialización, la clase base, *Message*, declara un método abstracto protegido *GetContent*, que debe implementarse en cada mensaje. Tomando el contenido del mensaje, Message genera una cabecera con los valores correctos y lo combina todo en una única cadena.

La deserialización es más compleja. Para implementarla, una posibilidad sería detectar el tipo de mensaje en la clase *Message* y llamar desde allí a un método de la clase adecuada. Sin embargo, para eso *Message* necesitaría conocer todos los mensajes existentes, dificultando una posible ampliación.

En su lugar, el método *Read* de *Message* devuelve un *BinMessage*, que es meramente un contenedor; y cada una de las demás clases hijas define una conversión explícita desde *BinMessage*.

De este modo, cada clase de mensaje es responsable tanto de su lectura como de su escritura. Esto no afecta apenas al código que haga uso de ellas, que simplemente ha de llevar a cabo una conversión a la clase correcta en función del tipo; lo cual habría de hacer, aún con la primera solución señalada antes, para poder acceder a los campos específicos de cada mensaje.

Esta misma implementación de los mensajes se emplea en el sistema de comunicaciones. La Figura 5 es un diagrama UML que muestra de forma simplificada las relaciones entre los mensajes.

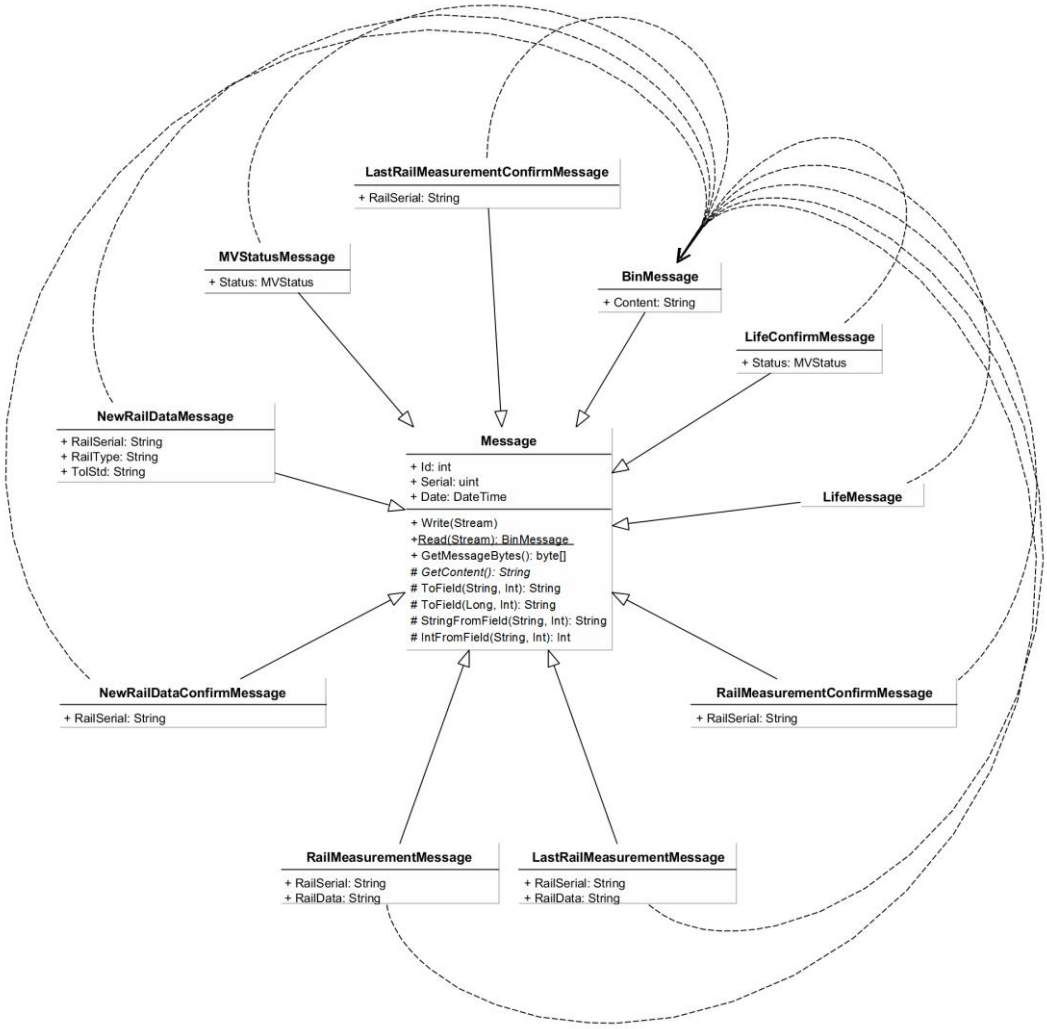


Figura 5: Diagrama de las clases empleadas en el procesamiento de mensajes

3.1.2. Interfaz gráfica

Para la implementación de este emulador se ha optado por una solución con una interfaz de usuario gráfica realizada mediante la tecnología de Windows Forms.

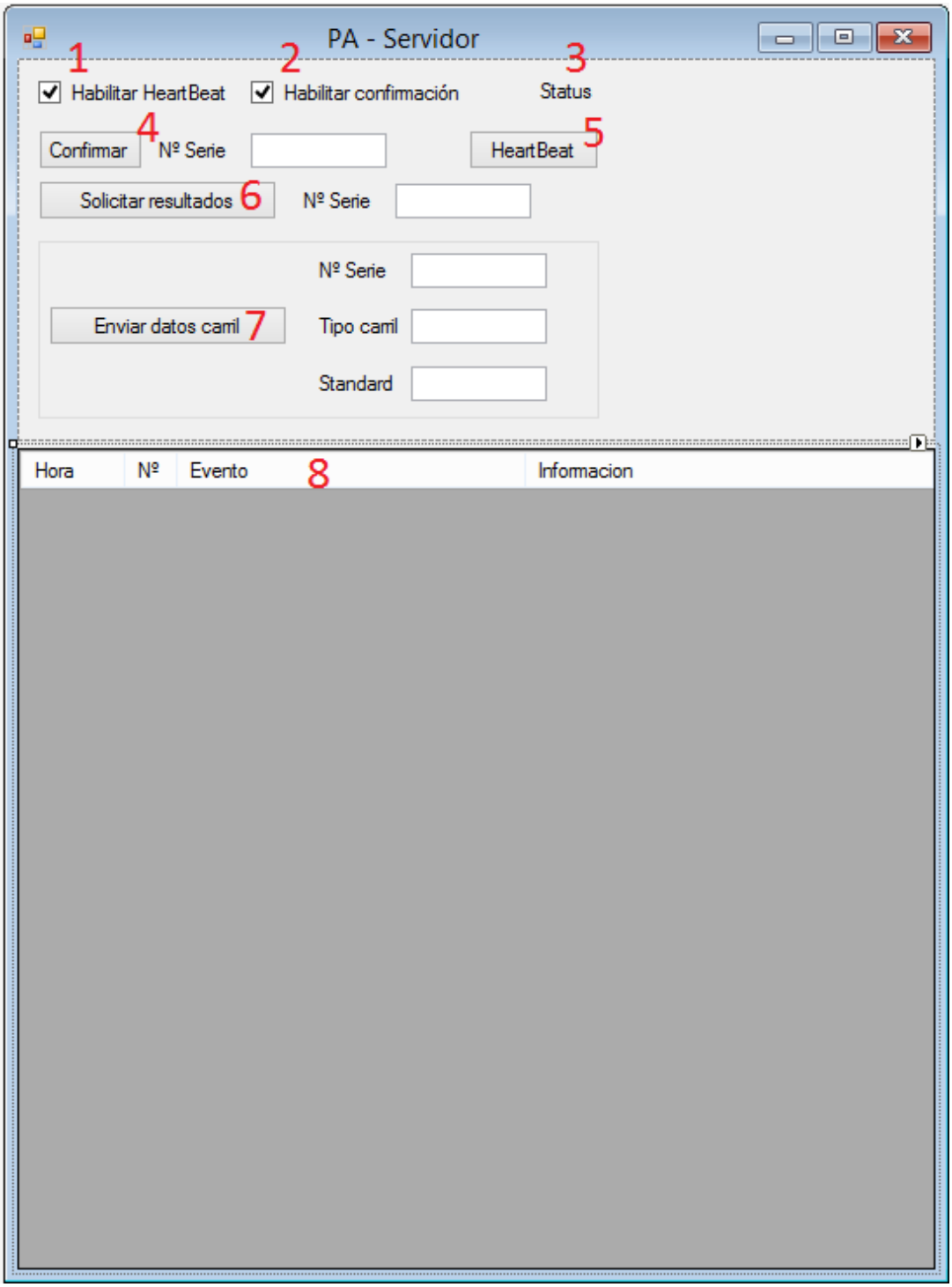


Figura 6: Interfaz gráfica del *PAServer*

La interfaz gráfica cuenta con una serie de controles (señalados en la Figura 6) que el usuario puede utilizar para emular el comportamiento del PA y elegir los mensajes que se han de generar. Estos controles son los siguientes:

1. **Habilitar heartbeat:** Este checkbox permite al usuario habilitar o deshabilitar el envío automático de *LifeMessage*, que se realiza cada 60 segundos, si este checkbox está marcado.
2. **Habilitar confirmación:** Mediante este checkbox, el usuario puede habilitar o deshabilitar el envío automático de mensajes *LastRailMeasurementConfirmMessage* tras recibir de *RailCommunications* mediciones nuevas.
3. **Status:** Este control muestra el estado actual del servidor:
    a. En color verde si el servidor está en el estado *Connected* (cliente conectado).
    b. En color naranja si el servidor está en el estado *Awaiting* (esperando conexión).
    c. En color rojo si el servidor está en el estado *Disconnected* (sin conexión).
4. **Confirmar:** Este botón permite confirmar manualmente la recepción de las mediciones (enviar inmediatamente un mensaje *LastRailMeasurementConfirmMessage*) por parte del *PAServer*. El carril cuya medición se confirma es el que tiene el número de serie que se introduzca en el cuadro de texto que se encuentra a la derecha del botón. Este número de serie sigue un formato determinado especificado por ArcelorMittal, si bien ni el servidor ni ningún programa del sistema validan el texto que se introduce.
5. **Heartbeat:** Este botón permite enviar manualmente un *LifeMessage*.
6. **Solicitar resultados:** Este botón permite enviar un mensaje *SendRailMeasurementRequestMessage*, mediante el cual el PA puede pedir que se le envíen los resultados de la medición de un carril con un número de serie igual que el introducido en el cuadro de texto que se encuentra a la derecha del botón.
7. **Enviar datos carril:** Este botón permite el envío de un mensaje *NewRailDataMessage* con los datos que se introduzcan en los cuadros de texto que están a su derecha; es decir: el número de serie, el tipo de carril y la norma con la que se va a medir este carril.
8. **Tabla de información:** En este campo se mostrarán todos los eventos producidos en el *PAServer*, ya sean de recepción, de envío de mensajes o de conexión o desconexión del cliente.

3.2. Emulador del PLC – *PLCServer*

Al igual que en el caso del emulador del PA, el *PAServer*, el *PLCServer* actúa como servidor en la comunicación con *RailCommunication*, en este caso reemplazando fuera del entorno de producción al PLC que coordina la medición. Este emulador incluye una interfaz gráfica similar a la del *PLCServer*.

Se implementa también un patrón State, con los mismos estados que el *PAServer* y las mismas transiciones entre ellos.

El diagrama de clases del PLCServer se puede observar en la Figura 7.

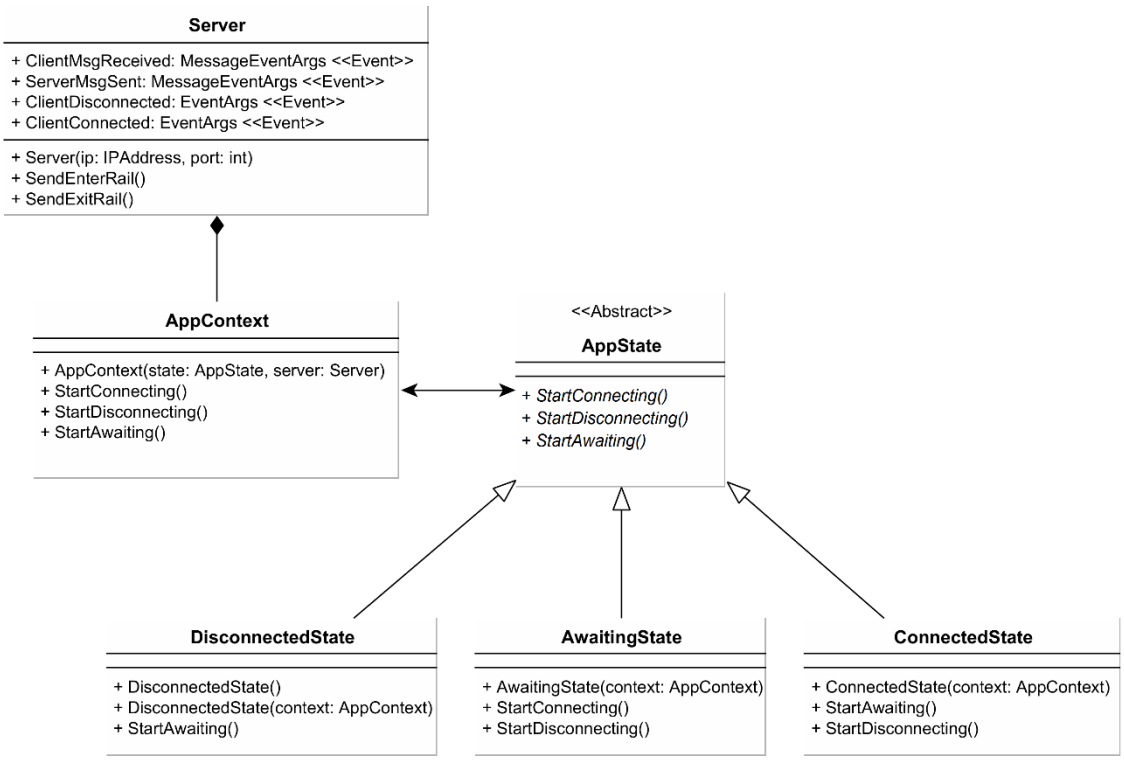


Figura 7: Diagrama de clases para el *PLCServer*

Los mensajes que van a ser enviados por el emulador del PLC, *PLCServer*, son mucho más sencillos que los empleados en el caso del PA. En lugar de una jerarquía de clases de mensaje, se trabaja simplemente con bits individuales. Por lo demás, la lógica es muy similar.

3.2.1. Interfaz gráfica

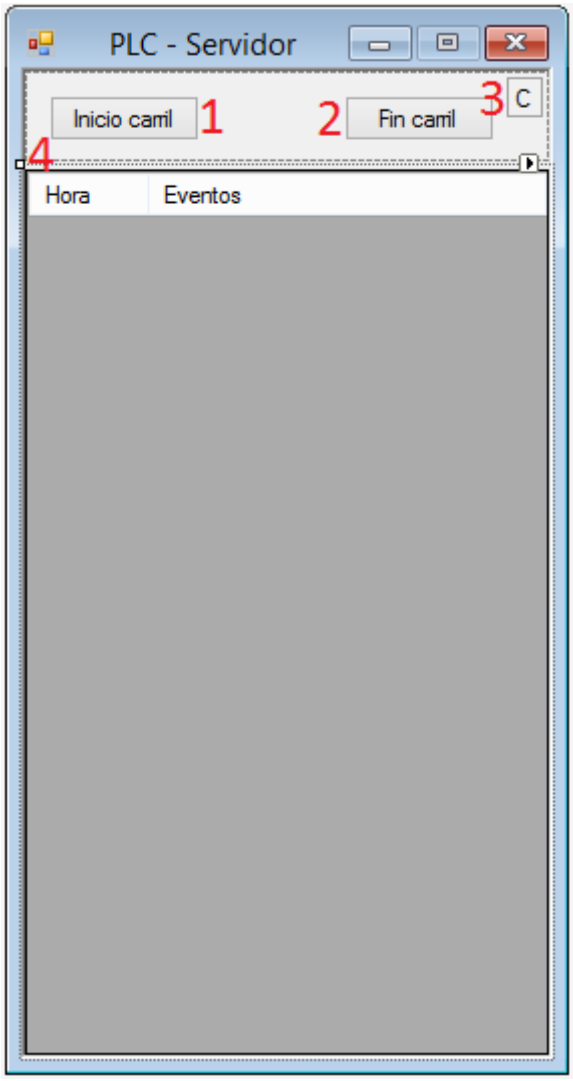


Figura 8: Interfaz gráfica del *PLCServer*

El *PLCServer* cuenta con una serie de controles (visibles en la Figura 8) que el usuario puede utilizar para emular el comportamiento del PLC y enviar los mensajes que podría generar. Estos controles son los siguientes:

1. **Inicio carril:** Este botón permite enviar de forma inmediata un mensaje de inicio de carril.
2. **Fin carril:** Este botón permite enviar de forma inmediata un mensaje de fin de carril.
3. **Estado:** Este control muestra el estado actual del servidor:
    a. En color verde si el cliente está conectado (estado *Connected*).
    b. En color naranja si el servidor está esperando conexiones del cliente (estado *Awaiting*).
    c. En color rojo si no hay conexión (estado *Disconnected*).
4. **Tabla de información:** En este campo se mostrarán todos los eventos producidos en el emulador del PLC: recepción y envío de mensajes, conexión y desconexión del cliente.

3.3. Medidor - *RailMeter*

La finalidad última del sistema de comunicaciones es servir de puente entre el medidor (*RailMeter*), que se describe en el documento III - Medidor, y otros equipos.

En el medidor se implementa (clase MeasurementService) un servidor WCF con la interfaz que se describe en la sección 2.3. Esta implementación se limita a lanzar los eventos internos correspondientes a cada posible llamada WCF. Al ocuparse WCF de todos los detalles de bajo nivel, esto tiene escasa complejidad técnica.

En cuanto al callback que debe implementarse en el cliente para que pueda utilizarse desde el medidor, este se expone al código del medidor a través de una propiedad (MeasurementService.Callback). De este modo, si no hay un cliente conectado, la llamada llega a una instancia de una "implementación" de la interfaz, llamada DummyMeasurementCallback, que se limita a registrarla como un error de forma transparente para el código que llama. Si hay un cliente conectado, la llamada pasa por una instancia de MeasurementCallbackWrapper, que se encarga de llamar al método WCF correspondiente, y en su caso de registrar las excepciones que se produzcan.

En la Figura 9 se muestra un diagrama de las relaciones entre las clases explicadas en los anteriores párrafos.

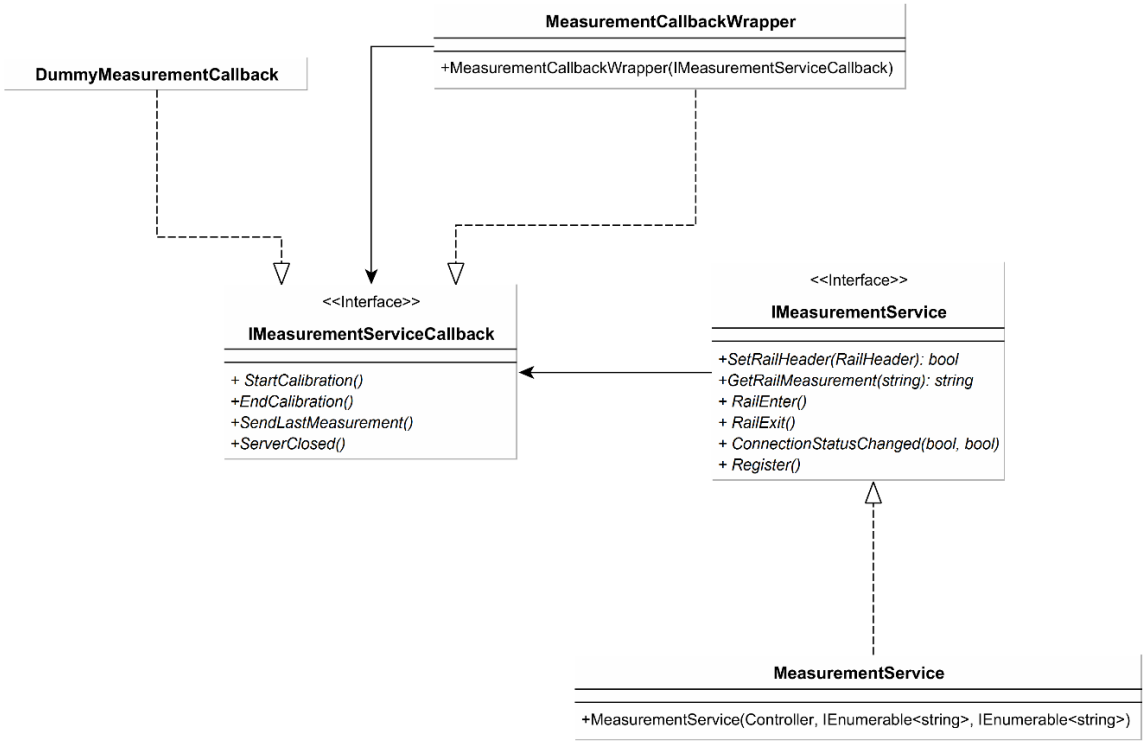


Figura 9: Diagrama de clases para la comunicación entre *RailMeter* y *RailCommunication*

3.4. Sistema de comunicaciones - *RailCommunications*

En este apartado se aborda la implementación del sistema de comunicaciones en sí. En el sistema de comunicaciones, la comunicación con cada equipo es responsabidad de una clase distinta.

En primer lugar, para el registro de mensajes de error e informativos se implementa una clase *Loggable*. Esta clase contiene eventos que se disparan cada vez que se registra un mensaje.

De la clase *Loggable* hereda la clase *Client*, que proporciona una interfaz común para iniciar y parar los clientes, y para que estos expongan el estado de la conexión.

Las clases *PAClient* y *PLCClient*, que heredan de *Client*, implementan los clientes para la comunicación con el ordenador de proceso y el PLC, respectivamente. Cada una de ellas proporciona métodos para enviar los mensajes que son competencia del cliente y eventos que informan de la llegada de mensajes recibidos del servidor.

La clase *MVCommunicationHandler*, que hereda directamente de *Loggable* (pues no tiene sentido que implemente algunos de los métodos de *Client*), consume los eventos producidos por *PAClient* y *PLCClient* y los traduce en llamadas a la interfaz WCF, y en el otro sentido recibe llamadas a través del callback WCF y las remite a los métodos correspondientes de *PAClient* y *PLCClient*. La estructura puede verse en la Figura 10.

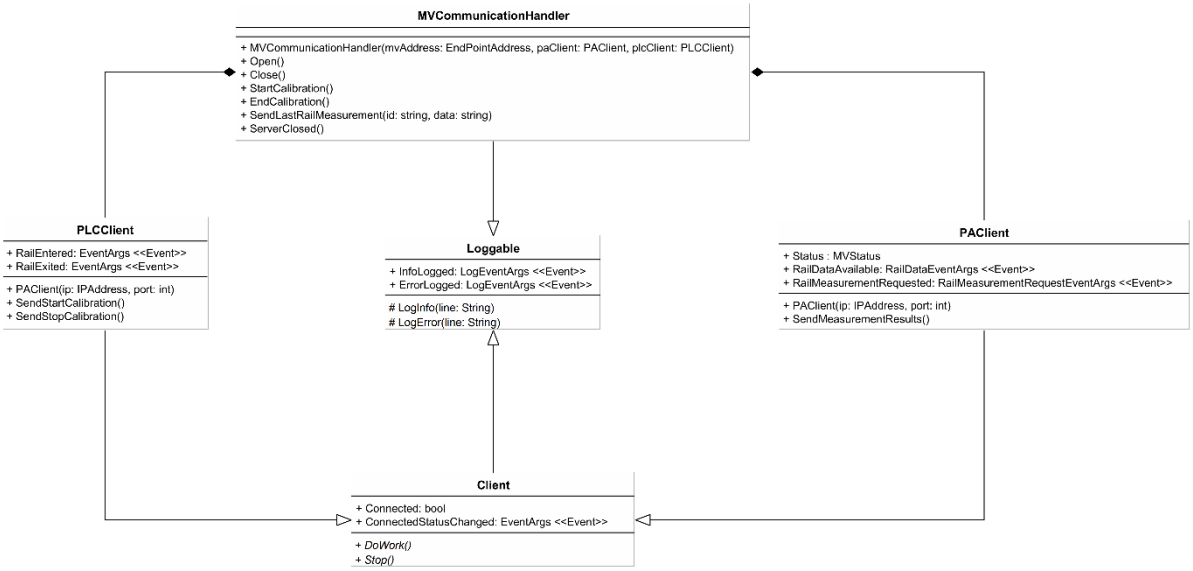


Figura 10: Diagrama de clases de RailCommunications

Para el tratamiento de la comunicación con el PA se emplean las mismas clases de mensajes que en la sección 2.1, y se aprovecha la implementación descrita en la sección 3.1.1.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

DOCUMENTO V

CALIBRACIÓN

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Introducción

Para combinar entre sí correctamente los patrones láser extraídos de las imágenes obtenidas por las cuatro cámaras, y generar así un perfil completo, es necesario conocer el modo de traducir las coordenadas de los puntos visibles en las imágenes a puntos del plano láser. Esto depende, como es obvio, de la posición de cada una de las cámaras y su orientación, pero también de algunas propiedades ópticas, como la distancia focal.

Esta información ha de obtenerse mediante un proceso de calibración. El proceso de calibración se divide, típicamente, en dos etapas:

- **La calibración intrínseca**, que permite obtener los valores de determinados parámetros que dependen exclusivamente de las características de las propias cámaras, tales como su distancia focal y las distorsiones de la óptica. La calibración intrínseca ha de realizarse una única vez por cada cámara, salvo que se cambie o manipule la lente.
- **La calibración extrínseca**, que calcula otros parámetros: el vector de traslación y la rotación de la cámara con respecto a un marco de referencia conocido. Esta calibración debe repetirse cada vez que se modifica la posición de alguna de las cámaras en relación con el plano láser, y periódicamente para compensar los movimientos mínimos de la estructura.

Dado que la calibración extrínseca debe realizarse periódicamente, así como siempre que cambian de posición las cámaras, el sistema desarrollado ha de incluir funcionalidad para poder llevarla a cabo en el mismo entorno en el que tiene lugar la medición.

La calibración extrínseca se realiza con ayuda de una plantilla específica, que puede verse en la Figura 1. Esta plantilla está formada por una placa metálica de la que sobresalen 13 cilindros de los que se conoce con exactitud sus radios y las distancias relativas entre sus centros. En los laterales de estos cilindros se refleja la luz de los láseres cuando la plantilla se coloca en paralelo al plano láser, a una distancia determinada, para realizar el procedimiento de calibración.

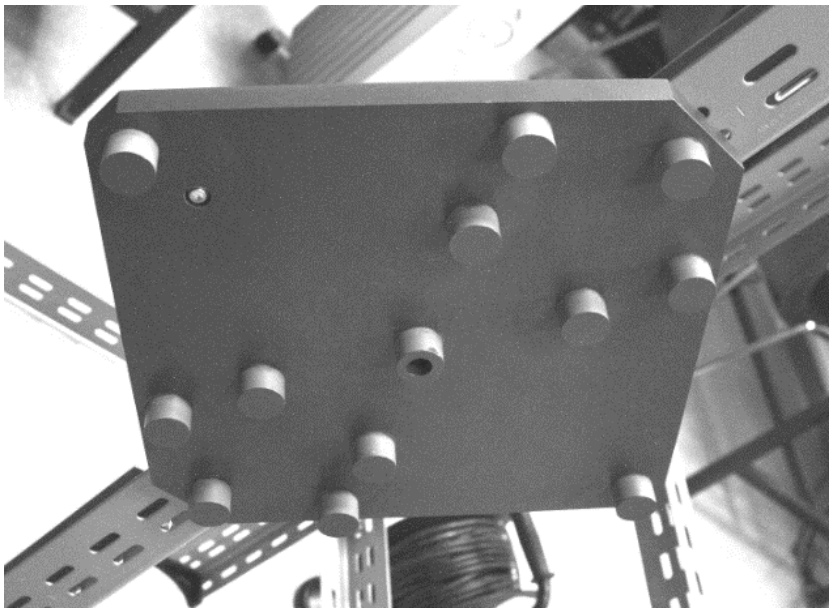


Figura 1: Plantilla de calibración, vista desde una de las cámaras utilizadas por el sistema

Se trata de detectar los cilindros en las imágenes adquiridas, calcular sus posiciones relativas y obtener los parámetros que permitan realizar la traducción deseada de la imagen al plano, utilizando para ello la información de la posición y el radio reales de cada uno de los cilindros.

De este modo, desde cada una de las cámaras es visible parte del contorno de algunos de los cilindros. Conociendo el ángulo aproximado de cada cámara con respecto a la plantilla, así como el patrón que siguen los cilindros en la plantilla, es posible asociar cada contorno a un cilindro. Comparando las posiciones relativas de los contornos y el patrón de los cilindros se puede hallar la posición de cada cámara en relación con el plano láser y su ángulo de rotación.

En las siguientes páginas se describen los algoritmos empleados para hacer frente a las particularidades de la plantilla descrita y el entorno en el que se utiliza, mientras que el modo en que se realiza el cálculo de los parámetros extrínsecos (es decir, la calibración propiamente dicha), que es preexistente y de uso general, queda fuera del ámbito de este documento.

2. Problemas iniciales

HALCON proporciona métodos para la detección de segmentos curvilíneos en la imagen (la proyección del láser sobre los cilindros de la plantilla de calibración) y para el cálculo de los parámetros de la cámara dadas dos listas de puntos[1], pero antes deben resolverse dos problemas para los que HALCON no ofrece soluciones inmediatas. Estos problemas son:

- La selección de los contornos que cada cámara capta con mejor calidad (es decir, mayor exactitud).
- La correspondencia entre los contornos seleccionados y los cilindros de la plantilla de calibración.

2.1. Selección de contornos

En primer lugar, han de hallarse los centros de los cilindros partiendo de las líneas dibujadas por los láseres. Desde las cámaras, estas líneas pueden verse como contornos de elipses cuyos centros son cercanos a los de los cilindros. Las líneas que han de detectarse pueden verse en la Figura 2.

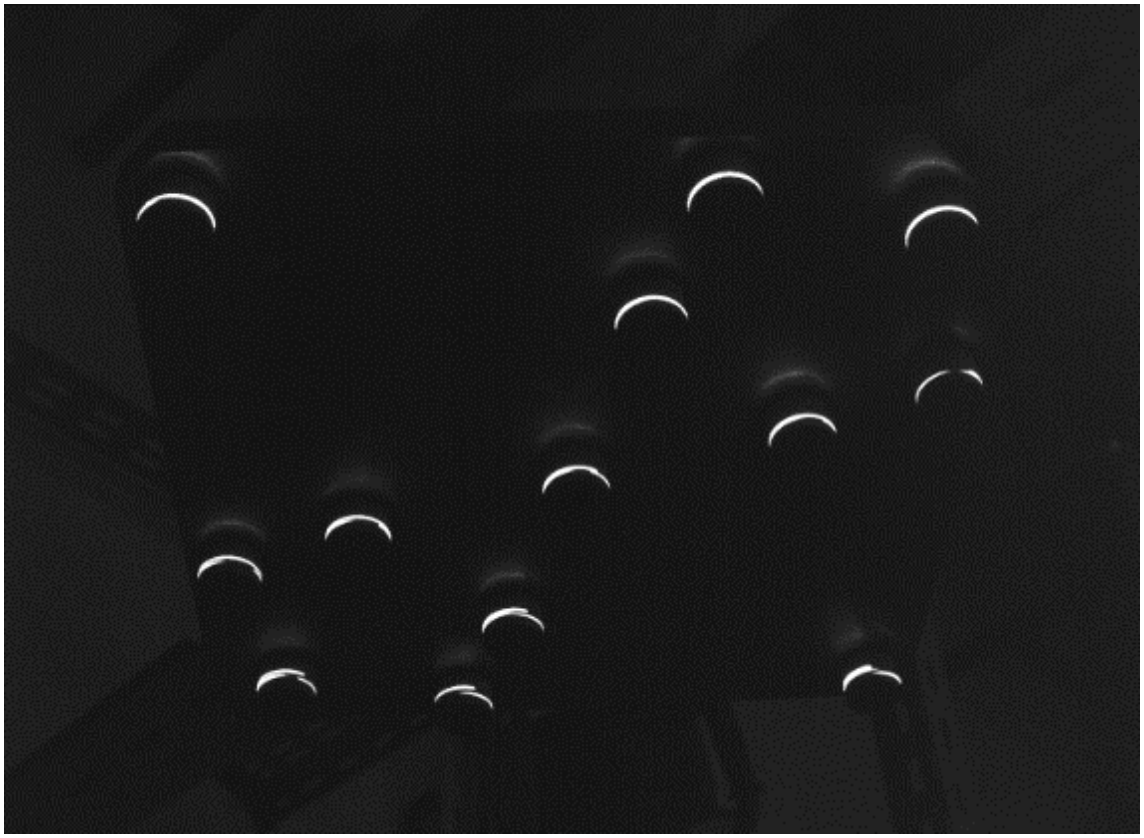


Figura 2: Líneas dibujadas por los láseres en la plantilla de calibración

El operador *lines_gauss* permite detectar líneas en la imagen y extraerlas. En este caso, las líneas que se obtienen son, en su mayoría, las marcadas por los láseres sobre los cilindros de la

[1] Es decir, tomando una lista de puntos del mundo y otra de puntos de la imagen correspondientes a ellos, así como los parámetros de la cámara empleada para tomar la imagen, computa los parámetros extrínsecos.

plantilla. Sin embargo, también podrían aparecer otras líneas; por ejemplo, reflejos en la placa o en la estructura que soporta la plantilla.

También es posible que de un único contorno se extraigan erróneamente varias líneas, especialmente si los láseres no están correctamente alineados. Pueden verse varios ejemplos de esto en la figura: los cilindros de la esquina inferior izquierda y el que se encuentra más a la derecha pueden ser problemáticos. Aun cuando de un contorno se obtiene una única línea, es posible que la mala alineación de los láseres le dé una forma anómala.

Sobre las líneas extraídas se emplea el operador de HALCON *fit_ellipse_contour_xld,* que trata de ajustarlas a elipses (es decir, completarlas de modo que formen elipses).

Un ejemplo del resultado de aplicar este operador puede verse en la Figura 3. En este ejemplo, las líneas detectadas con *lines_gauss* aparecen en rojo, mientras que las elipses creadas por *fit_ellipse_contour_xld* se muestran de color verde y sus centros se marcan con cruces amarillas.

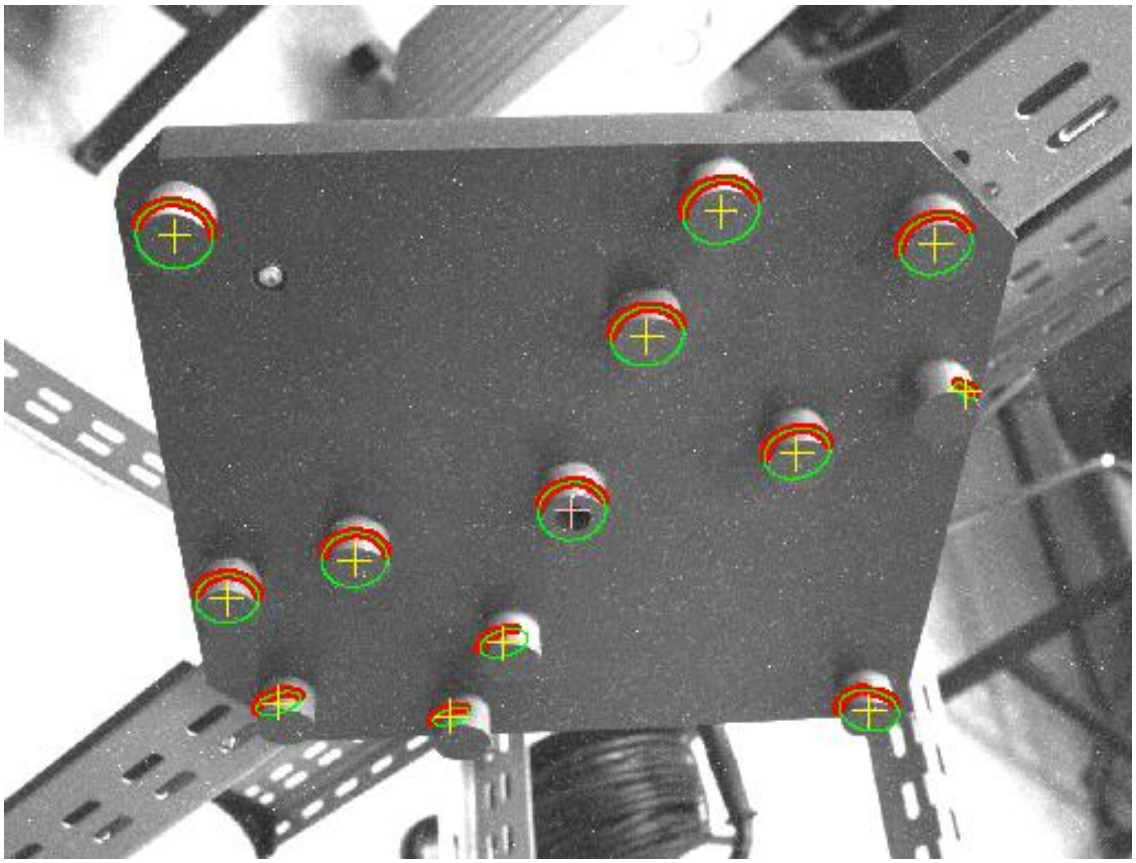


Figura 3: Líneas, elipses y sus centros

En los casos en los que las líneas no se extrajeron de forma correcta, el ajuste puede ser bastante inexacto, dando lugar a elipses de tamaños muy distintos a los esperados y, lo que es más importante, con centros muy alejados de los centros de los cilindros correspondientes. Un ejemplo de ajuste erróneo puede verse en la Figura 4.

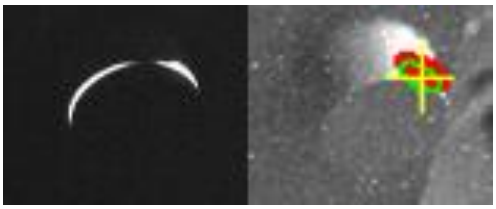


Figura 4: Ajuste erróneo de una elipse

El uso de estos centros reduciría la calidad de la calibración, por lo que resulta preciso detectar las elipses erróneas y descartarlas, seleccionando solamente las que puedan ser válidas (positivos).

Se han probado distintos criterios para conseguir este objetivo:

- **Longitud de las líneas:** Se mide la longitud de las líneas identificadas y se eliminan los valores atípicos. Este criterio no es adecuado porque la longitud de las líneas vistas por la cámara varía sensiblemente con su posición en la imagen, obteniendo en consecuencia una gran cantidad de falsos positivos y negativos.
- **Radio de las elipses:** Se descartan las elipses cuyo radio mayor tiene un valor atípico. Este criterio sufre el mismo problema que el anterior.
- **Diferencia de altura entre los extremos:** Partiendo de que, desde el punto de vista de la cámara, las curvas dibujadas por los láseres tienen forma de U invertida, se calcula la diferencia entre la altura de los extremos de cada curva y se divide entre su altura total. El valor se compara con una proporción máxima, como 0,75. Este criterio ofrece mejores resultados que los anteriores, pero existe un riesgo de falsos positivos, dado que aún en algunas curvas de buena calidad uno de los extremos puede alargarse más que el otro. Además, no tiene en cuenta el tamaño de la elipse.
- **Ángulo entre el radio por los extremos y la horizontal:** Se calcula el ángulo entre la horizontal y la recta que une el centro de la elipse con cada extremo de la curva, y se suma el valor absoluto para cada extremo. Este resultado se compara con un ángulo máximo, como 35 grados, descartándose si supera el valor de este. Hay falsos negativos porque se penaliza que un extremo de la curva se encuentre por debajo de la horizontal que pasa por el centro.
- **Ángulo entre el radio por los extremos por encima de la horizontal y esta:** Como en el caso anterior, pero si el extremo se encuentra por debajo de la horizontal que pasa por el centro se toma el valor 0. Si bien no tiene en cuenta el tamaño de la elipse, en la práctica esto carece de importancia porque las elipses anormalmente pequeñas suelen ser el producto de líneas incompletas, descartadas si se utiliza este criterio.

Este último criterio es el que finalmente se escogió. Para aportar una mayor robustez a la comprobación, se añade un criterio adicional basado en la proporción entre el radio mayor y el menor, que no puede ser mayor que 2. De este modo se descartan algunas elipses inválidas, principalmente producidas por reflejos de los láseres sobre la placa.

2.2. Correspondencia entre centros

Una vez escogidos los puntos de la imagen que se utilizarán, estos han de asociarse con los correspondientes de la plantilla, para poder llevar a cabo la calibración.

Los centros de los cilindros de la plantilla están dispuestos como se muestra en la Figura 5.

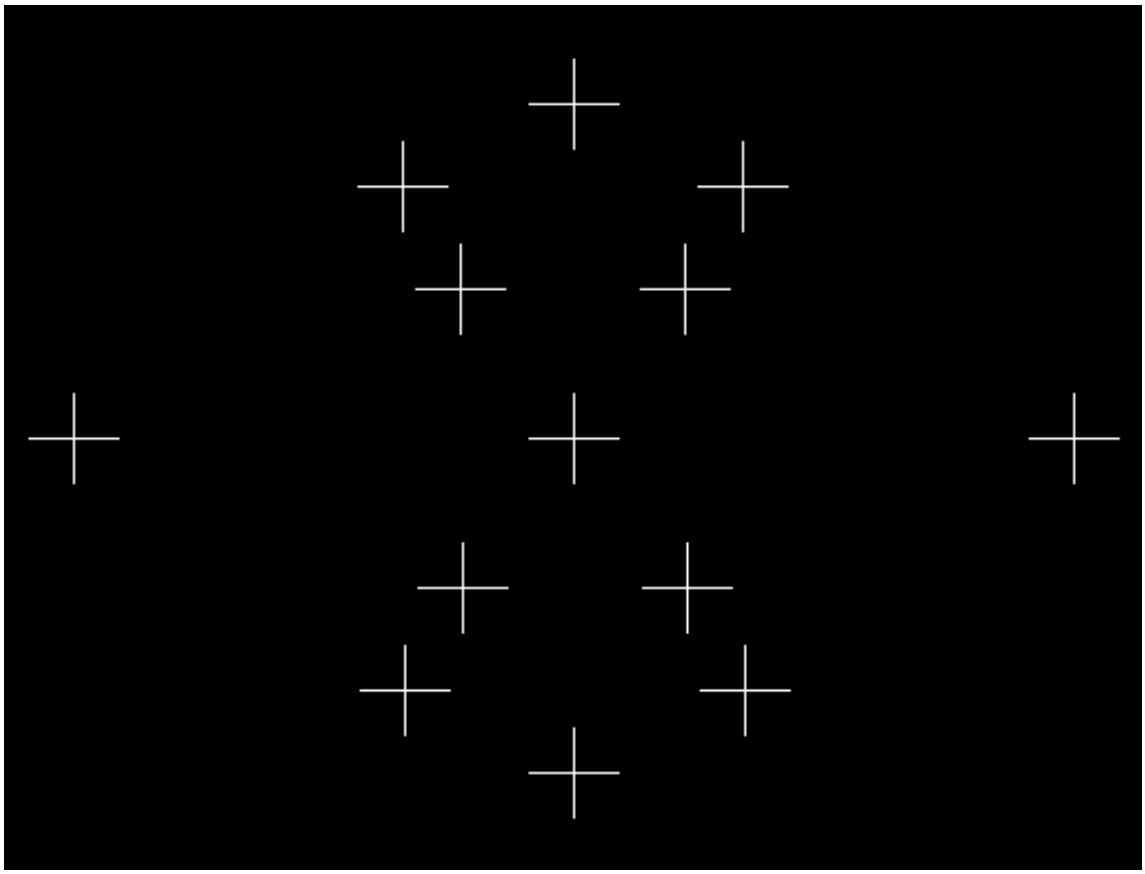


Figura 5: Centros de los cilindros de la plantilla

En primer lugar, se calculan los puntos centrales tanto de la imagen como de la plantilla real. Para esto se calcula el punto medio de ambas y se toma el más cercano en distancia euclídea.

Tomando este punto como referencia, se obtienen las distancias a los demás puntos y los ángulos con respecto a la horizontal.

Los puntos de referencia descubiertos son como se muestran en la Figura 6.

Para poder comparar las distancias de la imagen con las de la plantilla, se corrigen del siguiente modo:

- Se identifican las cuatro esquinas, los puntos aproximadamente a -180, -90, 0 y 90 grados más alejados del punto central, utilizando un algoritmo que se describirá más adelante.
- Se toman la distancia a las esquinas vertical y horizontal más alejadas, y se calcula la proporción entre ellas.
- Se calcula la distancia de cada punto al central, dando distinto peso a cada componente para corregir la disparidad calculada anteriormente.
- Se divide la distancia resultante entre la distancia media, dando lugar a una distancia relativa.

Los ángulos se limitan siempre al intervalo [-180, 180) grados, para facilitar las comparaciones.

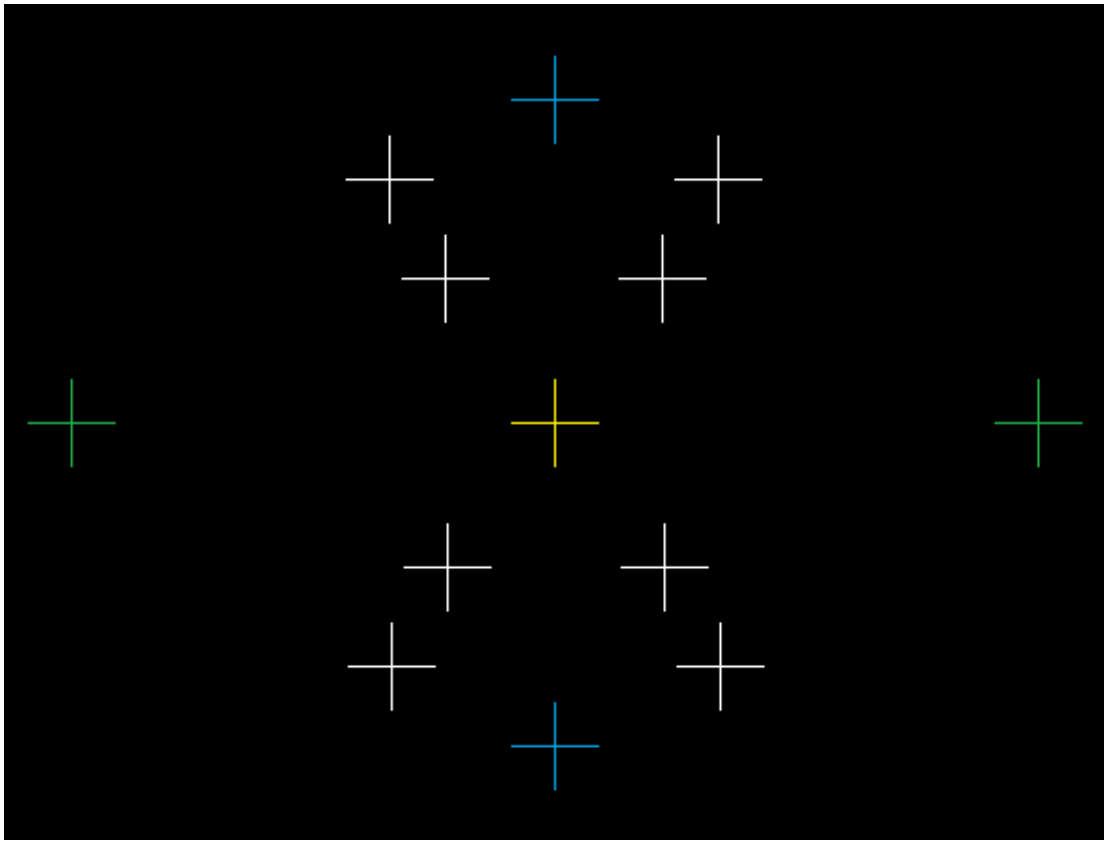


Figura 6: Puntos de referencia: central (en amarillo), esquinas verticales (en azul), esquinas horizontales (en verde)

Una vez calculados los ángulos y las distancias, para cada uno de los puntos de la plantilla, ordenados de menor a mayor distancia al centro, se calcula el punto de la imagen que mejor se ajusta.

En primer lugar, se toman los dos puntos que tienen un ángulo con respecto a la horizontal más similar al del punto para el que se busca una correspondencia. Si la diferencia entre alguno de los ángulos encontrados y el buscado es superior a un valor determinado, el punto se descarta. Finalmente, los puntos se ordenan según la diferencia entre sus distancias al centro y la distancia del punto buscado al centro.

Tomando esta lista ordenada, si el primer punto no se ha asociado aún a ningún punto real, se asocia al actual. Si ya ha sido asociado a algún punto real (más cercano al centro), se asocia al actual (eliminando la asociación al anterior) solamente si la diferencia entre las distancias es inferior a la que había en el caso anterior y la diferencia entre los ángulos es inferior al menos en una cantidad determinada. Si no se ha asociado el punto al actual, se comprueba el otro punto de la lista, si lo hay. Si finalmente ningún punto puede asociarse al punto actual, se deja sin asociar y se descarta.

Entre los centros de la imagen sobre los que se realiza esta operación se incluyen los que se han descartado anteriormente, salvo que sean muy cercanos a otros centros (como ocurre cuando una línea aparece partida y se detectan dos contornos donde debería haber uno solo)

o sus radios sean muy reducidos. Sin embargo, una vez terminado el mapeo se eliminan las asociaciones entre centros del cilindro y centros de la imagen descartados anteriormente. Esto se hace así porque de otro modo los centros de la imagen no descartados rodeados de otros que sí se descartaron podrían asociarse erróneamente a los centros de la plantilla correspondientes a alguno de estos últimos. Al tener en cuenta también los descartados, y permitir que estos "compitan" por estos centros, aunque luego no se empleen para la calibración propiamente dicha, se evita este problema.

3. Obtención de los parámetros extrínsecos

Una vez seleccionados los puntos que pueden emplearse para la calibración, esta se lleva a cabo del siguiente modo:

Para empezar, se ejecuta una primera iteración del procedimiento de calibración (que HALCON proporciona directamente con el nombre de *vector_to_pose*).

Luego, los parámetros obtenidos en esta primera iteración se emplean para pasar los contornos obtenidos previamente a coordenadas del mundo, donde han de ser aproximadamente circulares. Cada uno de los contornos se ajusta a un círculo, y se almacenan el centro de este (que ha de estar cerca del centro de la elipse correspondiente, pero no coincide exactamente) y su radio.

Después, se compara el radio de cada uno de los círculos con el del cilindro correspondiente, y se toman los ocho cilindros para los que el error (calculado como se describe en la sección 4.3) es menor. Sobre los centros y cilindros seleccionados se aplica por segunda vez *vector_to_pose*, y los parámetros resultantes son los que finalmente se emplean.

4. Medida del error

Estimar el error de la calibración resulta útil por tres motivos:

- Permite comparar distintas versiones del algoritmo de calibración durante el desarrollo.
- Permite conocer la calidad de la calibración realizada, permitiendo repetirla con una nueva imagen si hubo algún problema.
- En el caso particular del algoritmo empleado finalmente (descrito en la sección 3), con dos iteraciones de la calibración, permite descartar los puntos con peor calidad.

Se consideran varios métodos para estimar el error de la calibración.

4.1. Distancia entre los puntos de la circunferencia y el centro real

Para cada contorno visible en la imagen que se corresponda con un cilindro de la plantilla de calibración, se toman los puntos que lo forman, pasados a coordenadas del mundo, y para cada uno de esos puntos se calcula la diferencia en valor absoluto entre su distancia al centro real del cilindro correspondiente y el radio real del cilindro. Esta diferencia en valor absoluto se emplea como medida del error.

Esta medida del error puede calcularse por separado para cada cilindro, pudiendo conocer así el error de calibración para distintas regiones de la imagen.

4.2. Distancia entre los centros de la imagen y los de la plantilla

Para cada centro de la imagen (en coordenadas del mundo) asociado a algún centro de la plantilla se calcula la distancia euclídea entre ambos centros. Esta distancia se emplea como medida del error.

4.3. Diferencia entre los radios

Cada contorno visible en la imagen que se corresponda con un cilindro de la plantilla de calibración se ajusta a una circunferencia. Se emplea como medida del error la diferencia en valor absoluto entre el radio de esta circunferencia y el radio real del cilindro.

4.4. Comparación

La Tabla 1, la Tabla 2 y la Tabla 3 muestran los errores calculados con cada uno de los métodos descritos para determinadas imágenes (captadas simultáneamente con cada una de las cámaras). Cada columna tras las tres primeras representa uno de los cilindros de la plantilla.

Cuando para la imagen de una cámara determinada no se puede identificar algún cilindro, la celda correspondiente aparece en blanco.

Puede verse que los resultados el primer algoritmo descrito y el tercero son en la mayor parte de los casos cercanos entre sí, mientras que los del segundo algoritmo no parecen guardar relación con ellos.

Puesto que no parece haber motivos para considerar más exactos los resultados del primer algoritmo que los del tercero, se escoge este último, al ser más sencillo de implementar.

Cámara	Error medio	Error máximo	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0,193	0,322	0,298	0,210	0,151		0,078	0,111	0,106	0,218	0,322	0,162	0,160	0,239	0,260
2	0,121	0,217	0,091	0,147		0,061	0,133		0,217	0,094		0,094	0,166	0,082	0,124
3	0,198	0,378	0,244	0,141	0,151	0,095	0,065	0,223	0,378		0,227	0,148	0,206	0,271	0,224
4	0,062	0,132			0,132	0,057	0,082	0,036	0,026	0,053	0,102	0,041	0,038	0,052	0,067

Tabla 1: Distancia entre los puntos de la circunferencia y el centro real (mm)

Cámara	Error medio	Error máximo	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0,381	0,900	0,848	0,232	0,900		0,210	0,128	0,228	0,520	0,333	0,108	0,405	0,587	0,073
2	0,472	0,792	0,738	0,566		0,330	0,370		0,360	0,232		0,291	0,716	0,792	0,326
3	0,613	1,680	0,706	0,076	0,128	0,151	0,627	1,680	1,429		0,722	0,337	0,323	0,778	0,404
4	0,248	0,563			0,267	0,062	0,150	0,122	0,175	0,313	0,563	0,211	0,056	0,554	0,256

Tabla 2: Distancia entre los centros de la imagen y los de la plantilla (mm)

Cámara	Error medio	Error máximo	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0,195	0,287	0,264	0,272	0,213		0,133	0,122	0,128	0,287	0,232	0,162	0,134	0,200	0,195
2	0,109	0,202	0,024	0,101		0,147	0,107		0,202	0,019		0,102	0,143	0,108	0,133
3	0,206	0,316	0,289	0,190	0,152	0,132	0,073	0,242	0,187		0,316	0,128	0,219	0,310	0,230
4	0,044	0,134			0,070	0,004	0,004	0,027	0,029	0,077	0,134	0,060	0,010	0,030	0,036

Tabla 3: Diferencia entre los radios (mm)

5. Parámetros de calibración

El algoritmo de calibración precisa que se especifiquen varios parámetros para su funcionamiento. Estos parámetros son los siguientes:

- La imagen de la plantilla que se empleará para la calibración.
- Los parámetros intrínsecos de las cámaras.
- El ángulo de rotación aproximado de cada cámara con respecto a la plantilla.
- La disposición de los cilindros en la plantilla de calibración.
- Determinados parámetros de configuración que controlan de forma general el comportamiento del algoritmo, y que son los siguientes:
    - El nombre del algoritmo empleado para el ajuste de contornos a elipses en la calibración. Los algoritmos disponibles son “fitzgibbon”, “fhuber”, “ftukey”, “geometric”, “geohuber”, “geotukey”, “focpoints”, “fphuber”, “fptukey” y “voss”.
    - La intensidad del desenfoque gaussiano aplicado (llamada *sigma*).
    - El ángulo máximo (en grados) bajo la horizontal para el que no se descartan los contornos.
    - El ángulo máximo de diferencia entre el cilindro de la plantilla de la calibración y un círculo de la imagen para que este pueda considerarse equivalente.
    - El máximo número de puntos que se usarán en la calibración.

5.1. Selección de los parámetros de configuración empleados

Para seleccionar los parámetros de configuración de la calibración que ofrecen los mejores resultados, se actuó del siguiente modo:

En primer lugar, se eligió un conjunto de valores “razonables” para cada uno de los parámetros de configuración:

- Para el algoritmo de calibración, se consideraron los valores “fitzgibbon”, “fhuber”, “ftukey”, “geometric”, “geohuber” y “geotukey”. Se descartaron los restantes algoritmos posibles después de comprobar que sus resultados eran peores de forma consistente.
- Para la intensidad del desenfoque gaussiano, que en HALCON toma generalmente valores entre 1 y 5, se escogieron los valores entre 2 y 3 con incrementos de 0,25: 2; 2,25; 2,5; 2,75; 3.
- Para el ángulo máximo, 20, 35 y 50 grados.
- Para el máximo número de puntos, cuyo valor ideal se estimaba que estaba en torno a 10, los valores de 6, 8, 10 y 12.

Además de los parámetros de configuración de la calibración, se trató de determinar el mejor tiempo de exposición posible, entre 1 y 100 milisegundos.

Se llevó a cabo la calibración de cada una de las cámaras para cada una de las 36000 posibles combinaciones de parámetros[2], y se almacenó, para cada una, la calidad del resultado: error medio y máximo y número de cilindros utilizados. Teniendo en cuenta estos resultados, se tomó la combinación que minimizaba el error medio para las cuatro cámaras. Esta empleaba el algoritmo "fitzgibbon", con *sigma* de 2,75, un ángulo máximo de 35 grados, un máximo de 8 puntos y un tiempo de exposición de 75 ms.

[2] 6 x 5 x 3 x 4 x 100 = 36000.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

DOCUMENTO VI

VISUALIZADOR

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Introducción

El cometido de la funcionalidad de visualización es presentar los resultados de la medición, la evolución en el tiempo de los valores de las dimensiones, de modo que puedan ser interpretados fácilmente por humanos.

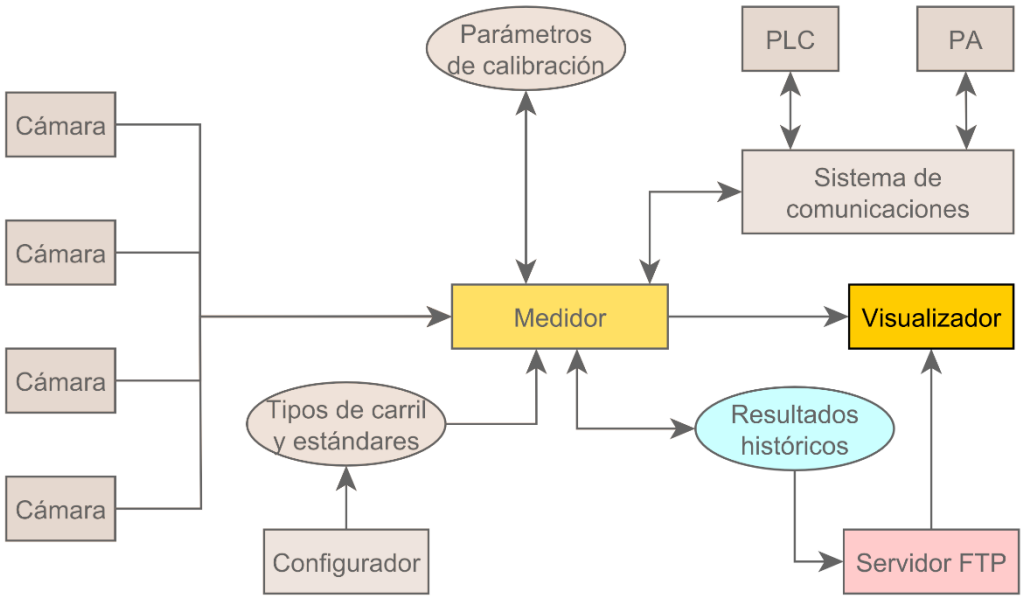


Figura 1: Diagrama del sistema, mostrando solamente los elementos relacionados con el visualizador

La funcionalidad de visualización se presenta en dos formas: una vista dentro del programa medidor para mostrar los resultados de la medición que se esté realizando en ese momento y un programa visualizador independiente, pensado para la visualización remota, que permite visualizar tanto los resultados de un carril que se esté midiendo en un momento concreto como los resultados históricos que se encuentren archivados.

La situación del visualizador dentro del sistema se indica en la Figura 1.

En este documento se presentan tanto la funcionalidad común como el programa independiente, mientras que la pestaña integrada en el medidor se describe en el documento dedicado a este.

2. Objetivos

1. Funcionalidad de visualización

1.1. Se mostrará, para cada dimensión:
  1.1.1. Su nombre.
  1.1.2. El valor esperado.
  1.1.3. El valor medido realmente en la posición seleccionada.
  1.1.4. Si este último está dentro de las tolerancias de la clase seleccionada.

1.2. Los valores medidos en la posición seleccionada se mostrarán como cotas sobre una representación del perfil.

1.3. Los valores de las dimensiones a lo largo de todo el carril se mostrarán en forma de gráfica.

1.4. Se indicará en el gráfico la situación relativa al valor esperado.

1.5. En el gráfico se mostrarán las tolerancias para la clase seleccionada.

1.6. Podrá seleccionarse una posición en la gráfica, que será la que se muestre en las demás vistas.

1.7. Para cada carril se indicarán:
  1.7.1. Su identificador.
  1.7.2. Su tipo.
  1.7.3. El estándar seguido.

1.8. En cualquier momento podrá seleccionarse una clase de tolerancia de entre las disponibles en la norma. La selección de una clase de tolerancia distinta afectará inmediatamente a todas las vistas que dependan de la clase de tolerancia seleccionada.

2. Programa visualizador

2.1. Modo en línea
  2.1.1. Se establecerá una conexión con el programa medidor.
  2.1.2. Al establecer la conexión se descargará inmediatamente la información ya medida hasta ese momento.
  2.1.3. El visualizador recibirá actualizaciones del servidor con cada medición, y las añadirá a los datos ya visibles.
  2.1.4. Cuando se inicie la medición de un carril nuevo, se sustituirán los datos del que se muestre hasta entonces por los del nuevo carril. Las mediciones antiguas se ocultarán.

2.2. Modo fuera de línea remoto
  2.2.1. Se establecerá una conexión FTP con el servidor que albergue los resultados de mediciones anteriores.
  2.2.2. El usuario podrá elegir un carril ya medido para mostrar, seleccionándolo en un árbol de directorios.
  2.2.3. El carril escogido se mostrará.

2.3. Modo fuera de línea local
  2.3.1. El usuario elegirá un fichero de mediciones en el sistema local.
  2.3.2. El carril contenido en el fichero se mostrará.

2.4. Configuración

2.4.1. El usuario podrá modificar el orden en el que se muestran las dimensiones en las distintas vistas.

2.4.2. El usuario podrá ocultar determinadas dimensiones.

2.4.3. El usuario podrá modificar los datos para la conexión a los servidores.

2.4.4. La configuración será persistente entre ejecuciones del programa visualizador.

2.5. Tratamiento de errores

2.5.1. Los fallos al intentar acceder a los ficheros de mediciones o al cargarlos se señalarán al usuario, y se le permitirá elegir otro fichero u otro modo de visualización.

3. Programa visualizador

El programa visualizador permite acceder de forma remota a los resultados de un carril que se esté midiendo en un momento dado, así como los resultados históricos que se encuentren archivados.

Este programa tiene dos modos de funcionamiento diferenciados: un modo en línea, en el que se conectará al medidor y mostrará los valores en tiempo real según se midan, y otro modo de funcionamiento fuera de línea, que permitirá consultar los resultados de mediciones anteriores. Este último puede dividirse en otros dos, uno para la visualización de ficheros locales y otro para los ficheros remotos.

En esta sección se describe la interfaz gráfica del programa visualizador y la funcionalidad que presenta para los usuarios.

3.1. Ventana principal

La ventana principal del programa visualizador permite seleccionar carriles y conocer los resultados de su medición.

La barra de menús superior incluye opciones para seleccionar el modo de visualización, exportar los resultados y modificar la configuración. Dependiendo del modo de funcionamiento, puede mostrarse una barra a la izquierda para seleccionar carriles concretos. El resto del espacio de la ventana está reservado para la visualización. Todo esto puede verse en la Figura 2.

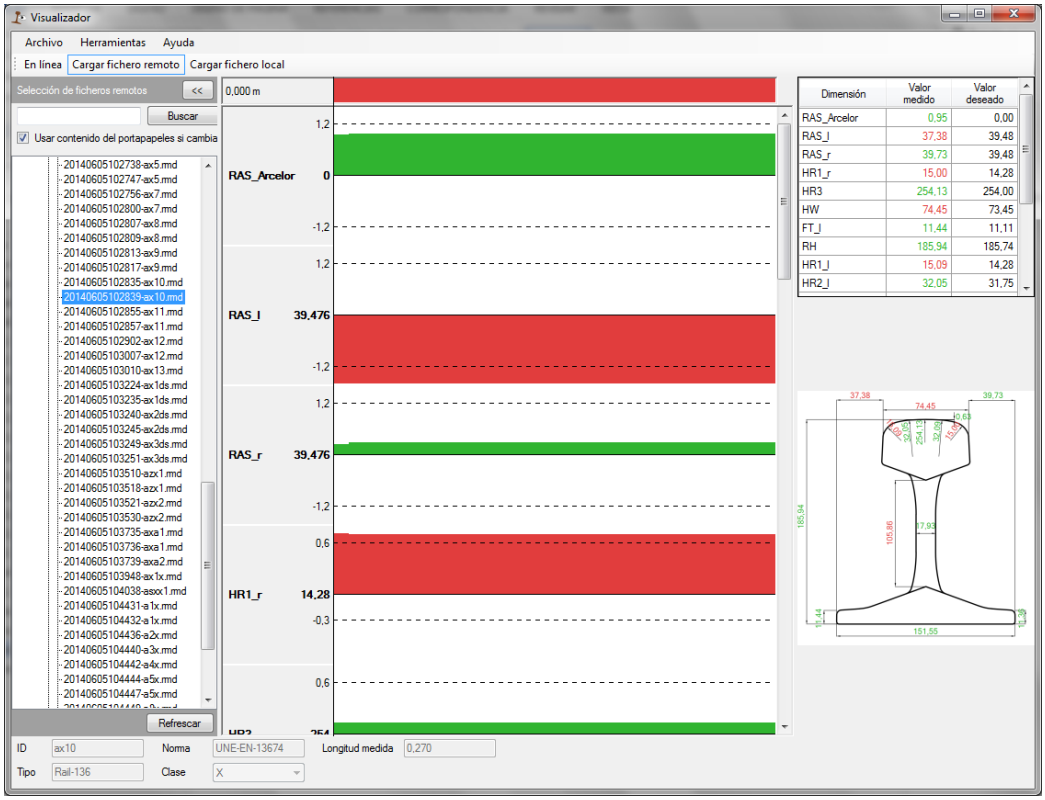


Figura 2: Programa visualizador en funcionamiento, en modo *Cargar fichero remoto*

Los resultados de la medición se muestran de tres formas distintas:

- En la zona central se muestra una gráfica para cada dimensión.
  En el centro de cada gráfica aparece una línea horizontal gruesa y sólida que representa el valor esperado. Por encima y por debajo de esta línea se muestran líneas discontinuas que representan las tolerancias correspondientes, que dependen de la dimensión, la norma y la clase seleccionada. Por cada sección de carril medida puede verse una línea vertical que aparece en verde si la diferencia entre el valor medido y el esperado está dentro de las tolerancias y en rojo si no es así.
  Al pulsar sobre una sección de carril esta se selecciona y pasa a mostrarse en las otras vistas. Esto se representa sobre la gráfica con una línea vertical de color negro.
  Por encima de las gráficas se muestra una barra más delgada que indica, para cada sección, si todas sus dimensiones toman valores tolerables o no.
- Arriba a la derecha se muestra una lista de dimensiones junto con los valores esperados y los valores medidos en la sección de carril seleccionada. Los valores medidos se muestran en rojo o en verde dependiendo de si están fuera o dentro de las tolerancias.
- Abajo a la derecha se muestra una representación de una sección de carril, sobre la que aparecen acotados los valores de las dimensiones.

En la parte inferior de la ventana pueden verse el identificador del carril medido, su tipo, la norma que sigue, la clase dentro de la norma y la longitud de carril medida. Además, si la norma dispone de varias clases es posible modificar la clase seleccionada. Al cambiar la clase, se actualizan las vistas del carril para mostrar las tolerancias correspondientes a la nueva clase. El funcionamiento de estas vistas se describe con mayor detalle en la sección 4.1.2.

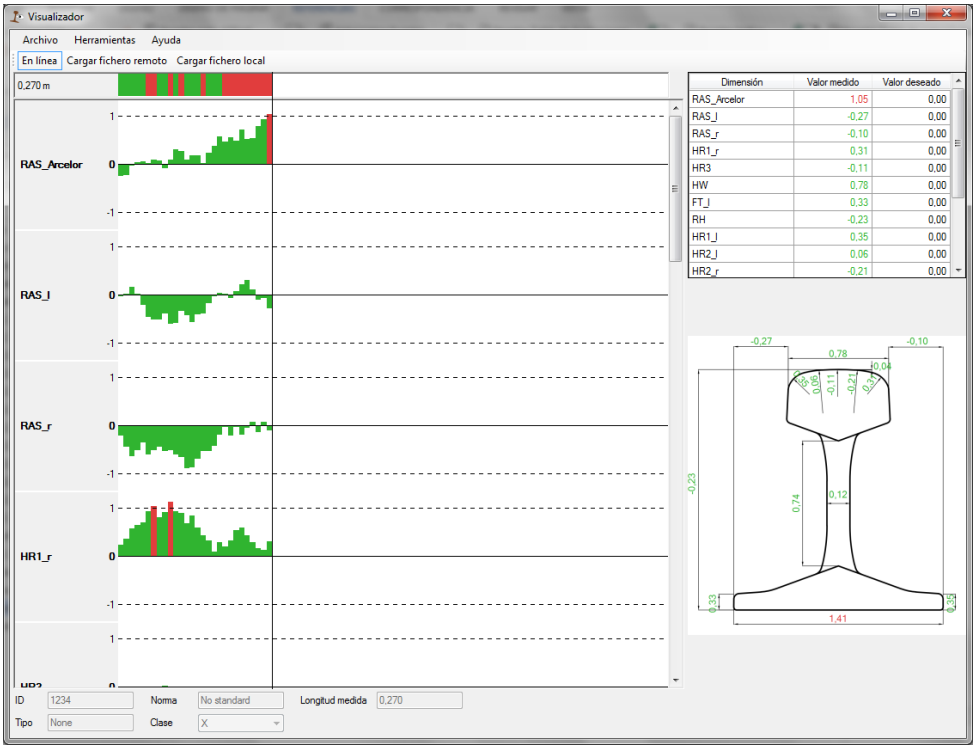


Figura 3: Programa visualizador en funcionamiento, en el modo *En línea*

En la zona superior aparecen tres botones, cada uno de los cuales está asociado a un modo de funcionamiento.

En el modo *En línea* (Figura 3), el visualizador se conecta al medidor y trata de obtener de él los resultados en tiempo real.

En el modo *Cargar fichero remoto* (Figura 2), el visualizador se conecta a un servidor FTP, muestra en un panel lateral el contenido de un directorio raíz preestablecido y permite que el usuario seleccione un fichero de resultados. Una vez seleccionado, lo descarga y trata de mostrarlo. Si el usuario selecciona entonces un fichero distinto, se repite el proceso con dicho fichero. El panel lateral permite además la búsqueda por identificador, e incluye una funcionalidad para emplear automáticamente el contenido del portapapeles.

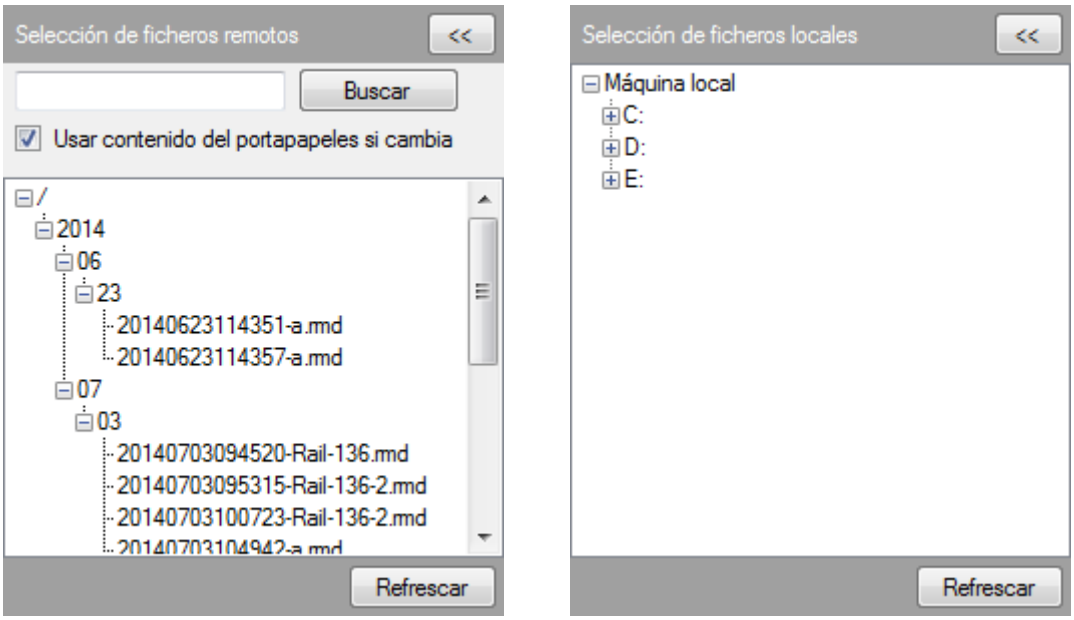


Figura 4: Barras laterales de selección de ficheros

En el modo *Cargar fichero local*, el visualizador muestra los resultados de medición almacenados en el equipo y permite seleccionarlos y mostrarlos del mismo modo que en el caso anterior. Se muestra un panel lateral muy similar al de *Cargar fichero remoto*, que puede verse en la Figura 4.

Al pulsar sobre un botón desmarcado, se marca este y se desmarcan los demás. Al pulsar sobre un botón marcado, este se desmarca. Desmarcar el botón no borra los resultados de la medición, pero sí oculta la barra lateral en los modos *Cargar fichero remoto* y *Cargar fichero local* y deja de mostrar carriles nuevos (aunque sigue añadiendo nuevas mediciones al actual hasta que se completa) en el modo *En línea*.

En cuanto a la barra de menús, esta incluye las siguientes opciones:

- Archivo
    - Cargar: Permite seleccionar un modo de funcionamiento y un carril, del mismo modo que los botones que aparecen debajo.

- Exportar: Permite guardar los gráficos de las dimensiones visibles en formato PNG. Puede guardar cada gráfico por separado o todos en el mismo fichero.
- Cerrar: Termina la visualización del carril actual.
- Salir: Cierra el programa visualizador.

- Herramientas
  - Configuración: Abre la ventana de configuración que se describe en la sección siguiente.

3.2. Ventana de configuración

La ventana de configuración se divide en dos pestañas, que permiten especificar varios aspectos del comportamiento del programa visualizador. La configuración se almacena de forma persistente, y los cambios que se realicen se aplican automáticamente al cerrar la ventana.

La pestaña de dimensiones (Figura 5) permite especificar cuáles de estas han de ser visibles y en qué orden. Para ello muestra una lista de las dimensiones seleccionadas y otra de las disponibles, permitiendo mover dimensiones de una lista a la otra, y en vertical dentro de la misma lista. Las dimensiones que aparecen en esta pestaña son las que figuran en el fichero de configuración (sección 4.2.1.1).

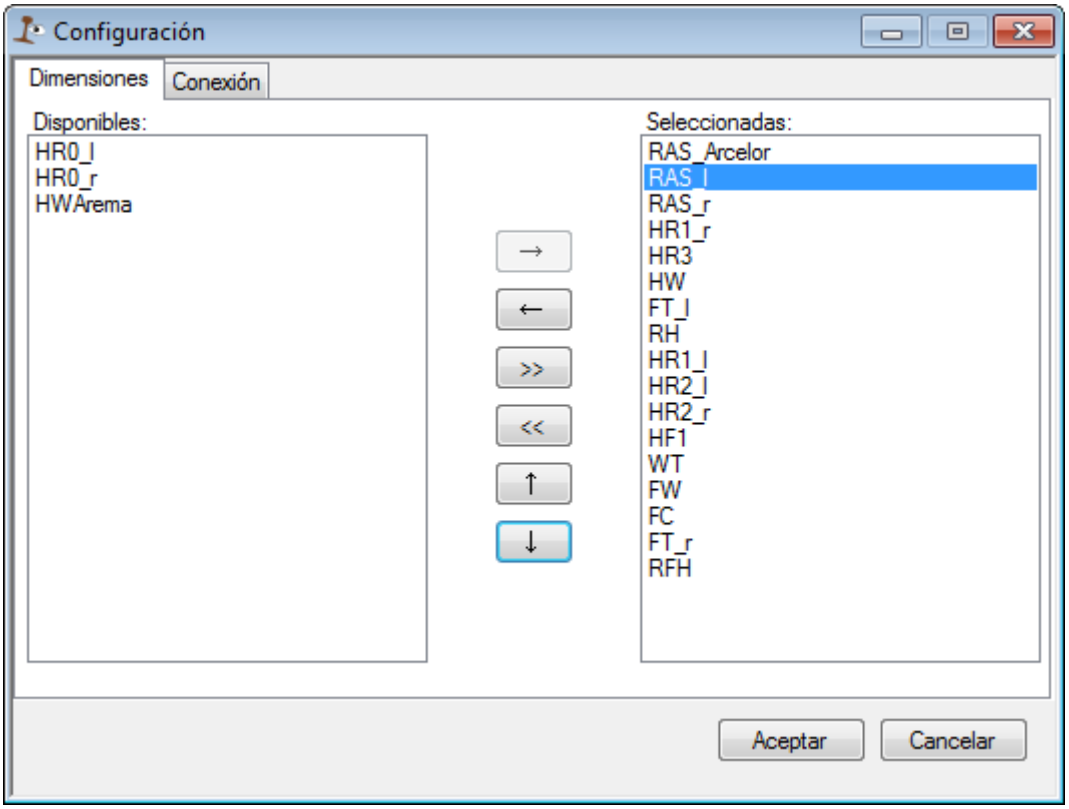


Figura 5: Pestaña de dimensiones de la ventana de configuración

La pestaña de conexión (Figura 6) permite especificar los parámetros de la comunicación con el servidor. Se especifican por separado los parámetros empleados en el modo *En línea* y los del modo *Cargar fichero remoto*, aunque la dirección del servidor será típicamente la misma en ambos.

Los parámetros que pueden modificarse en esta pestaña son:

- Para el modo *En línea* (servidor de medición):
  - Servidor: dirección IP del servidor.
  - Puerto: puerto TCP en el que escucha el servidor de medición. Por defecto se utiliza el puerto 6071.
- Para el modo *Cargar fichero remoto* (servidor FTP):

- URI base: URI del servidor FTP, incluyendo la ruta hasta el directorio raíz de las mediciones.
- Puerto: puerto TCP en el que escucha el servidor FTP. Por defecto se utiliza el puerto 21.
- Timeout: Tiempo, en milisegundos, que se esperará para que el servidor responda una solicitud.
- Usuario: Identificación del usuario en el servidor FTP.
- Contraseña: Contraseña del usuario en el servidor FTP.

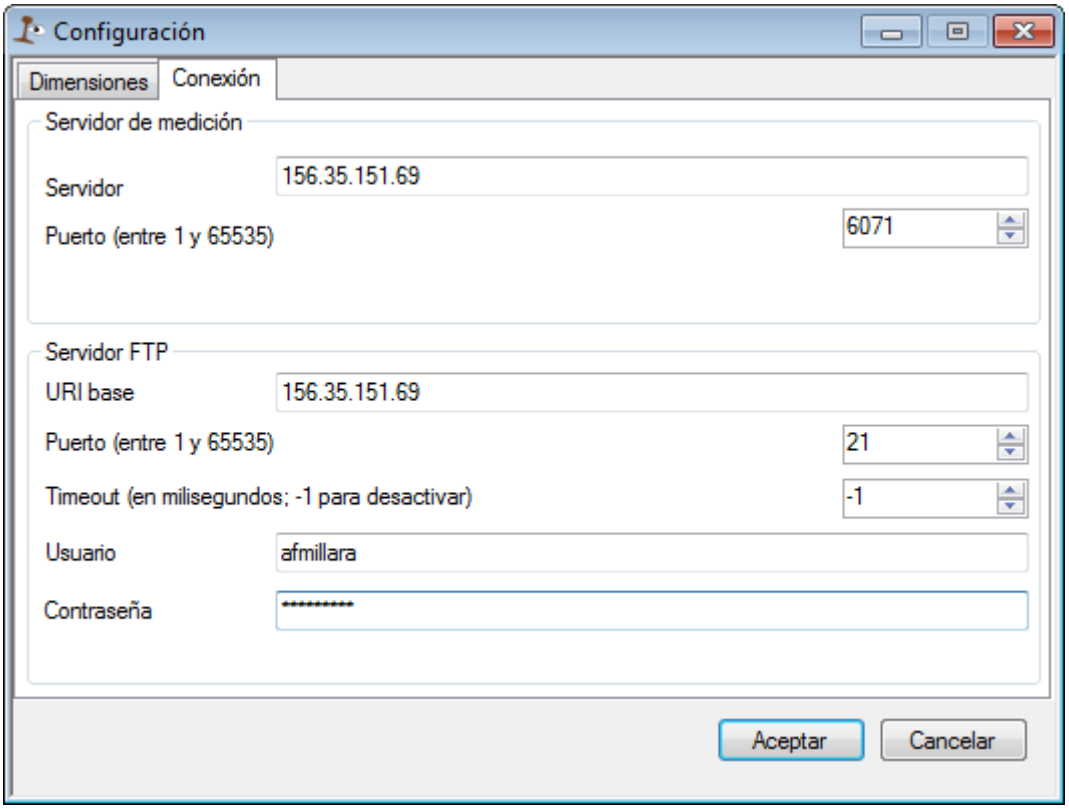


Figura 6: Pestaña de conexión de la ventana de configuración

3.3. Modos de funcionamiento

Como se ha establecido, el programa visualizador puede conectarse al medidor y mostrar la medición que se esté llevando a cabo (modo *En línea*) o acceder a ficheros de un servidor remoto (modo *Cargar fichero remoto*) o ficheros locales (modo *Cargar fichero local*).

3.3.1. Modo *En línea*

En el modo *En línea*, el programa visualizador se conecta al medidor, empleando los datos de conexión que se especificaron en la configuración, y muestra los resultados del carril que se esté midiendo en ese momento. Si hay una medición ya comenzada en el momento en el que el visualizador se conecta, los resultados que ya estén disponibles se muestran inmediatamente. Si la medición concluye mientras el programa visualizador está conectado, este se queda esperando hasta que comienza la medición de un nuevo carril, y entonces pasa a mostrar ese nuevo carril en lugar del anterior. El visualizador continúa mostrando en todo momento lo que se esté midiendo hasta que se desactiva este modo de medición. Este modo emplea una fuente *ServerStreamRailSource* (sección 4.1.3.2).

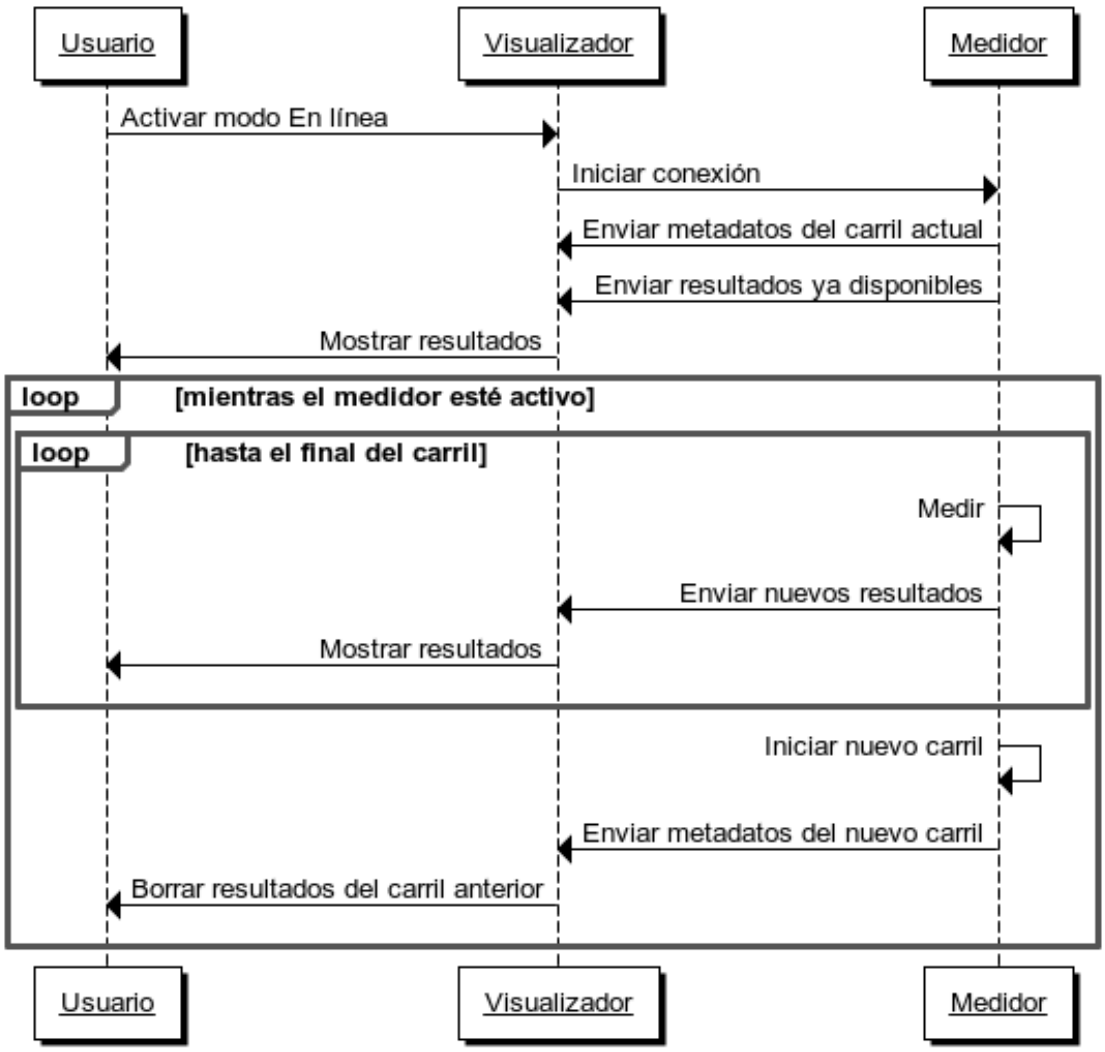


Figura 7: Diagrama de secuencia del modo *En línea*

3.3.2. Modo Cargar fichero remoto

En este modo, el programa visualizador se conecta a un servidor a través del protocolo FTP, muestra una estructura de directorios y muestra los carriles contenidos en los ficheros de resultados contenidos en esa estructura que el usuario seleccione.

Típicamente, el servidor FTP tomará los ficheros de resultados generados por el medidor, conservando la estructura de directorios creada por este. Este modo emplea una fuente *StreamRailSource* (sección 4.1.3.1).

3.3.3. Modo Cargar fichero local

Este modo permite que el usuario seleccione ficheros de mediciones que estén disponibles en el equipo local, y los muestra de la misma forma que en el caso del modo anterior. Este modo emplea una fuente *StreamRailSource* (sección 4.1.3.1).

4. Funcionalidad de visualización

La funcionalidad de visualización ha de proporcionar varias vistas distintas de los resultados de la medición (metadatos, evolución en el tiempo, estado en un momento concreto, etc.), que a su vez ha de obtener de diferentes fuentes (los hilos de medición dentro del mismo programa, el medidor a través de la red, ficheros…).

4.1. Modelo-Vista-Controlador

Resulta particularmente adecuado el uso del patrón *Modelo-Vista-Controlador* en la arquitectura de esta funcionalidad. Este patrón permite lograr una separación clara entre el modelo, las distintas fuentes o controladores y las vistas, permitiendo combinar las fuentes y las vistas sin necesidad de escribir código específico para cada combinación.

En la Figura 8 se muestra la estructura general, sobre la que se profundiza en las siguientes páginas, y que a grandes rasgos es la siguiente:

- *Rail*: Representa un carril completo. Contiene referencias a dimensiones, metadatos y resultados de medición. Es el *Modelo*.
- *Dimension*: Representa una de las dimensiones del perfil que se miden.
- *Tolerance*: Almacena el rango de valores tolerables para una dimensión y una clase de tolerancia concretas.
- *RailHeader*: Almacena metadatos del carril.
- *RailSection*: Representa una sección de carril medida.
- *RailSource*: Genera resultados para que las vistas los consuman. *Es el Controlador.*
- *RailView*: Muestra los resultados contenidos en un carril. Es la *Vista*.

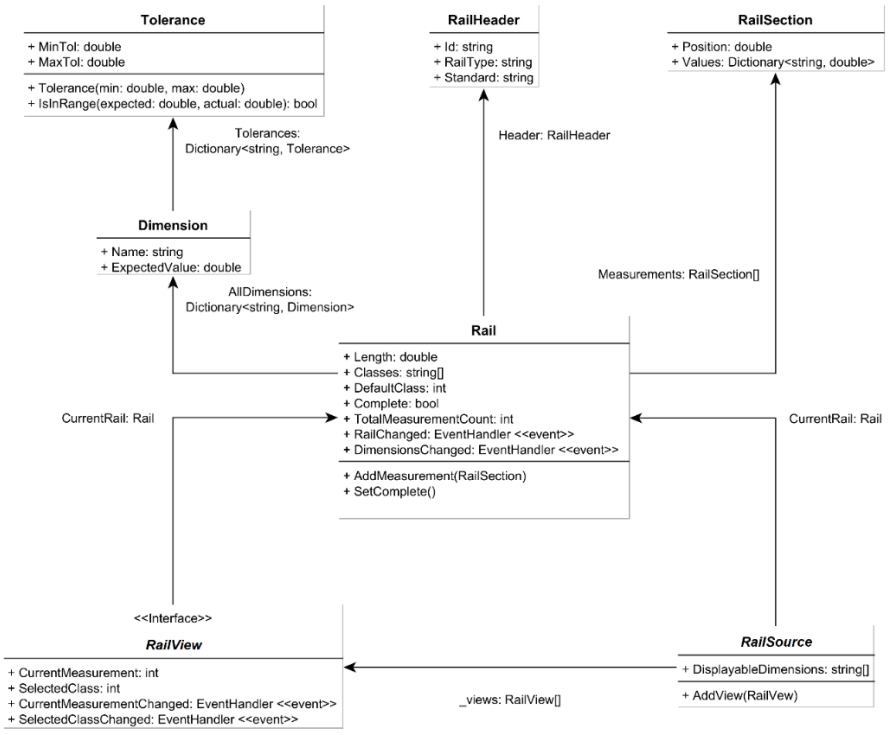


Figura 8: Diagrama de las clases del patrón Modelo-Vista-Controlador empleado

4.1.1. Modelo

Como puede verse en el diagrama de la Figura 8, la clase Rail almacena, directamente o indirectamente a través de otras clases, toda la información referente a un carril concreto que es relevante para la visualización.

Esta información es la siguiente:

- **Metadatos:** Identificador del carril, tipo y nombre de la norma empleada (clase RailHeader); longitud total del carril, número de secciones esperado y si la medición ha concluido ya.
  El visualizador no emplea el identificador, el tipo ni el nombre de la norma, pero sí los muestra al usuario.
  El número de secciones esperado se emplea para calcular el ancho que ocupa cada sección en el gráfico.
- **Norma:** dimensiones relevantes del carril (clase Dimension) y tolerancias de cada dimensión (clase Tolerance).
  Cada dimensión tiene un nombre, que se muestra al usuario, un valor esperado y una o varias tolerancias, que expresan la distancia máxima al valor esperado, tanto por encima como por debajo, para que una sección de carril se considere satisfactoria. Si existen varias tolerancias, cada una de ellas se corresponde con una "clase de tolerancia" dentro de la norma.
- **Contenido:** secciones de carril medidas (clase RailSection).
  Para cada sección de carril se almacena su posición relativa al comienzo del carril, así como el valor que en ella toma cada dimensión.

La clase Rail proporciona un método AddMeasurement, para añadir los resultados de una sección, y otro método SetComplete, para señalar que el carril ha terminado. Además, expone dos eventos: RailChanged, que se dispara cuando se añaden resultados a la medición, y DimensionsChanged, que se dispara cuando se modifica qué dimensiones son visibles.

4.1.2. Vista

La interfaz *RailView* incluye los eventos y las propiedades que deben implementar las clases que visualizan carriles. Estos son:

- La propiedad **CurrentRail**, el carril que se está visualizando.
- La propiedad **CurrentMeasurement**, entero correspondiente a la medición que se está visualizando en un momento determinado.
- La propiedad **SelectedClass**, entero correspondiente al índice de la clase de tolerancia que se emplea.
- Los eventos **CurrentMeasurementChanged** y **SelectedClassChanged**, que se disparan cuando cambian las dos anteriores, respectivamente.

CurrentMeasurement y **SelectedClass** se establecen en -1 si no tienen sentido para la vista de la que se trata o no están fijados en un momento determinado.

Existen múltiples implementaciones de *RailView* para mostrar carriles al usuario de distintas formas, almacenarlos en ficheros, enviarlos por red, etc.

4.1.2.1. RailDimensionsView

La clase *RailDimensionsView* proporciona una vista sencilla de los valores de las dimensiones en un momento concreto. Se trata de un *DataGridView* que muestra en formato tabular los nombres de las dimensiones ("Dimensión"), los valores medidos para cada una ("Valor medido") y los valores esperados ("Valor deseado"). Los valores medidos se muestran de color rojo si se encuentran fuera de las tolerancias correspondientes y de color verde si están dentro, como puede apreciarse en la Figura 9.

Dimensión	Valor medido	Valor deseado
RAS_Arcelor	0,00	70,00
RAS_r	16,74	16,00
RH	143,09	159,00
RAS_l	0,00	140,00
HR0_l	66,86	80,00
HWArema	34,29	0,00

Figura 9: Vista de valores de las dimensiones

4.1.2.2. RailProfileView

La clase *RailProfileView* proporciona una vista de perfil del carril, a la que se superponen los valores medidos de las dimensiones. Estos valores medidos pueden mostrarse de color rojo o verde dependiendo de si están fuera o dentro de las tolerancias.

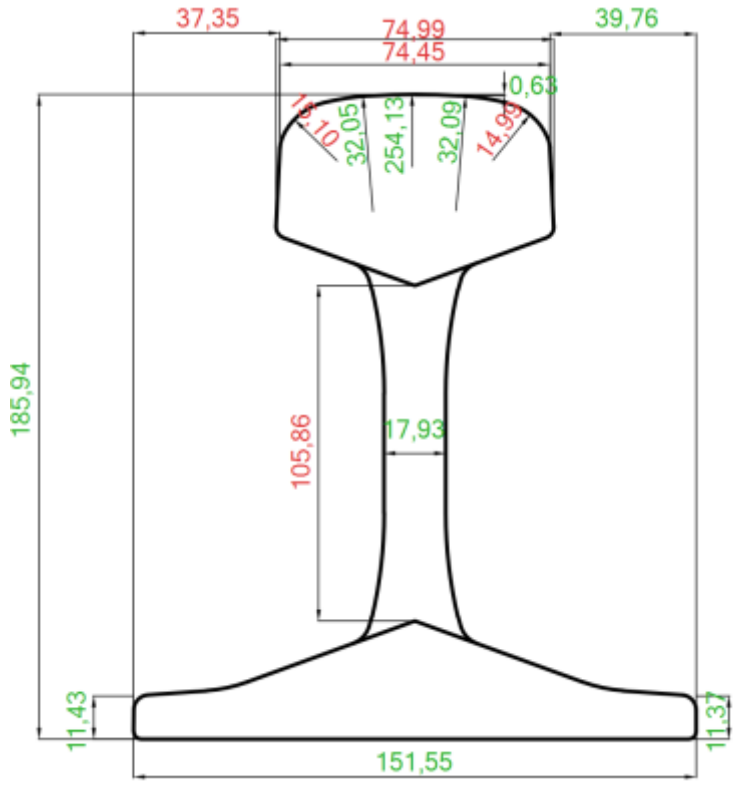


Figura 10: Silueta empleada para los carriles simétricos

No todas las dimensiones pueden mostrarse en esta vista. Por ejemplo, la concavidad del pie (FC) no puede incluirse porque no hay ningún lugar en las imágenes donde puedan aparecer las cotas correspondientes, ya que en un carril ideal el pie no es cóncavo y por tanto esta dimensión tiene el valor 0.

La vista de perfil que se muestra no se corresponde realmente con el perfil del carril medido. Si bien el medidor dispone de información suficiente para dibujar la silueta, y de hecho incluye la funcionalidad para hacerlo; el programa visualizador independiente no tiene acceso a los datos necesarios, ya que emplea solamente los resultados de medición en el formato que se describe en el anexo E - Formato de fichero de resultados. Transmitirle también al programa visualizador la información que necesitaría para mostrar la silueta supondría un coste demasiado alto para las ventajas que supone esa funcionalidad.

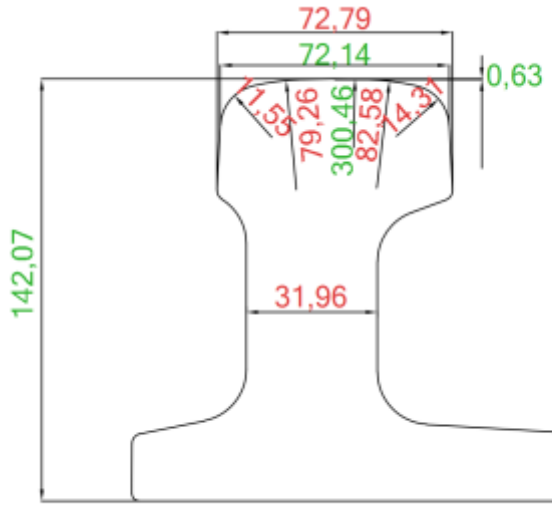


Figura 11: Silueta empleada para los carriles asimétricos

En lugar de eso, se muestra solamente una de entre dos imágenes generadas con anterioridad, dependiendo de si el carril medido es simétrico o asimétrico. Estas imágenes pueden verse en la Figura 10 y en la Figura 11. Las imágenes mostradas, así como las reglas que se emplean para elegir cuál mostrar, son configurables del modo que se describe en la sección 4.2.

4.1.2.3. RailTendenceView

La clase *RailTendenceView* proporciona una vista de los valores de las dimensiones a lo largo de todo el carril. Además, permite seleccionar una posición concreta para mostrarla en otras vistas.

La vista de tendencias, que se muestra en la Figura 12, está conformada por varias partes diferenciadas.

Figura 12: Gráfico de tendencias de un carril completo

En la esquina superior izquierda aparece un indicador de la posición seleccionada, en metros. A su derecha se muestra una barra que indica, para cada sección de carril medida, si todas las dimensiones tienen valores dentro de las respectivas tolerancias, mostrando una línea de color verde si es así y rojo en caso contrario.

Debajo de la barra superior se muestra una gráfica para cada una de las dimensiones visibles. Esta representa mediante líneas verticales el valor de la dimensión en cada sección medida, en relación con el valor esperado (línea horizontal continua). Las líneas se muestran verdes si están dentro de las tolerancias (líneas horizontales discontinuas) y rojas si no lo están.

A la izquierda de cada gráfica se ven el nombre de la dimensión que representa, el valor esperado y las tolerancias superior e inferior, que pueden ser diferentes entre sí.

La línea vertical negra señala la posición seleccionada actualmente. La línea vertical azul sigue al puntero del ratón y puede emplearse para seleccionar una posición distinta.

Cuando el carril que se está mostrando no está marcado como completo (propiedad *Complete* de la clase *Rail*, véase la sección 4.1.1), se asume un tamaño determinado y se deja en blanco la región que todavía no se ha medido, como puede verse en la Figura 13.

Si este tamaño no es suficiente, al llenarse el gráfico se reduce automáticamente el grosor de las líneas para dejar sitio a las mediciones restantes, de modo que en ningún caso sea necesario hacer scroll horizontal. Al completarse la medición, se ajusta nuevamente el tamaño de las líneas de forma que ocupen el espacio disponible en su totalidad.

En caso de que la posición seleccionada sea la última, al añadir nuevas mediciones se desplaza de modo que la seleccionada siga siendo la última en todo momento, hasta que se seleccione manualmente otra posición distinta.

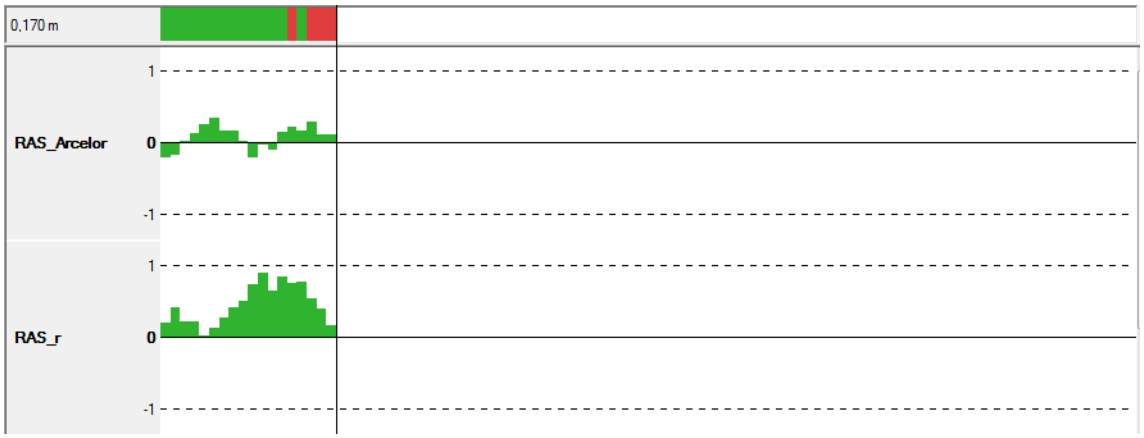


Figura 13: Gráfico de tendencias de un carril incompleto

4.1.2.4. RailParametersView

La clase *RailParametersView* proporciona una vista de los parámetros del carril, que puede verse en la Figura 14.

Los parámetros que se muestran son:

- **ID:** Identificador del carril
- **Tipo:** Tipo de carril
- **Norma:** Norma que se sigue para el control de la calidad
- **Clase:** Clase de tolerancia de entre las disponibles en la norma. Puede modificarse del mismo modo que *RailClassView* (4.1.2.5).
- **Longitud medida:** Longitud de carril medida, en milímetros. Si la medición no ha concluido, va actualizándose según se añaden nuevas mediciones.

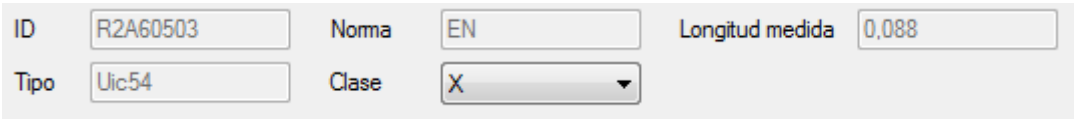


Figura 14: Vista de parámetros del carril

4.1.2.5. RailClassView

RailClassView es una vista de las clases de tolerancia disponibles, que permite seleccionar una clase distinta en cualquier momento. Al seleccionar una clase distinta se vuelven a generar todas las vistas para tener en cuenta los valores de las nuevas tolerancias.

Las clases disponibles son las que se especifican en la norma que sigue el carril.

Esta vista se muestra en la Figura 15.

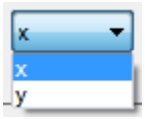


Figura 15: Selector de clases de tolerancia

4.1.2.6. RailStreamView

RailStreamView es una clase base para la generación de resultados de medición en el formato que se describe en el anexo E - Formato de fichero de resultados.

4.1.2.7. RailServerView

RailServerView hereda de *RailStreamView* y se encarga de enviar los resultados de medición a los clientes que se encuentren conectados, según se reciban. Admite múltiples clientes al mismo tiempo, y almacena los resultados que ya ha recibido para transmitírselos a los nuevos clientes que se conecten, inmediatamente después de establecerse la conexión.

4.1.2.8. RailFileView

RailFileView hereda de *RailStreamView* y se ocupa de almacenar los resultados de medición en ficheros. Los resultados se guardan en el momento en el que concluye la medición del carril, en la ruta <Raíz>\yyyy\mm\dd\yyyymmddHHMMSS-<Id>.rmd, donde <Raíz> es el lugar establecido para almacenar resultados, <Id> es el identificador del carril e yyyy, mm, dd, HH, MM y SS representan el año, el mes, el día del mes, la hora, el minuto y el segundo en que finalizó la medición, cada uno con dos dígitos salvo el año, con 4.

4.1.2.9. RailViewSet

La clase *RailViewSet* representa un grupo de vistas asociadas entre sí. Siguiendo un patrón *Composite*, el *RailViewSet* transmite los cambios en sus propiedades a las vistas subyacentes, y los que se produzcan en una vista a las demás. Por ejemplo, si se selecciona una clase de tolerancia en un *RailClassView* que forme parte de un *RailViewSet*, los cambios se propagarán al resto de las vistas asociadas, que pasarán a utilizar esa misma clase de tolerancia.

4.1.3. Controlador

La clase abstracta *RailSource* incluye las propiedades y los métodos que han de implementar las distintas fuentes de carriles, o controladores. Estos son los siguientes:

- La propiedad **CurrentRail**, el carril actual.
- La propiedad **DisplayableDimensions**, una lista de las dimensiones que pueden mostrarse.

- El método **AddView**, que registra una nueva vista y la asocia a los carriles que muestre la fuente.

Existen múltiples controladores distintos que proporcionan carriles obtenidos de diferentes formas: leídos de fichero o del servidor de medición, etc.

4.1.3.1. StreamRailSource

La fuente *StreamRailSource* lee resultados de medición de un flujo en el formato que se describe en el anexo E - Formato de fichero de resultados y los va añadiendo al modelo. Se emplea para la lectura de mediciones obtenidas de ficheros.

4.1.3.2. ServerStreamRailSource

La fuente *ServerStreamRailSource*, basada en *StreamRailSource*, lee resultados obtenidos del servidor de mediciones, y los va añadiendo al modelo según los recibe. Se emplea en el modo en línea del programa medidor.

4.1.3.3. MeasurementRailSource

La fuente *MeasurementRailSource* emite los valores de las dimensiones según se calculan. Se trata de un adaptador entre el sistema de medición y el de visualización. Se emplea en la pestaña de visualización del medidor, que se describe en el documento III - Medidor.

4.1.3.4. RandomRailSource

La fuente *RandomRailSource* emite valores al azar para una cantidad determinada de mediciones. Se emplea para probar el programa visualizador cuando el medidor no está disponible.

4.1.3.5. EmptyRailSource

La fuente *EmptyRailSource* emite un carril vacío. Se emplea cuando no hay nada que la vista deba mostrar.

4.2. Configuración de las dimensiones

Puesto que el programa visualizador es independiente, en su funcionamiento interno, del significado de cada dimensión (puesto que no emplea ninguna funcionalidad de medición, y conoce solamente los resultados que se le envían, como se señala en la sección 4.1.2.2), resulta factible evitar asumir, dentro del código del programa, la existencia y la naturaleza de determinadas dimensiones. Esto es deseable, puesto que de este modo se facilita el mantenimiento del sistema en caso de añadirse dimensiones, modificarse sus nombres, etc.

Sin embargo, existen ciertos datos de cada dimensión que la funcionalidad de visualización debe conocer. Estos datos son su descripción, que puede mostrarse en determinadas vistas como *tooltip* sobre el nombre, y su ubicación en la vista de perfil (*RailProfileView*, 4.1.2.2). Esta información debe almacenarse en un fichero de configuración.

4.2.1. Fichero de configuración

Se emplea un fichero de configuración, *dimensions.xml*, de cuyo elemento raíz, *visualization*, cuelgan otros dos, *dimensions*, que contiene una lista de dimensiones con sus descripciones asociadas, y *profiles*, que contiene una lista de imágenes de perfil. Esta estructura general puede verse en la Figura 16.

```
<visualization>
    <dimensions>
        <!-- Dimensiones -->
    </dimensions>
    <profiles>
        <!-- Perfiles -->
    </profiles>
</visualization>
```

Figura 16: Estructura general del fichero de configuración

4.2.1.1. Dimensiones

El elemento *dimensions* contiene una lista de dimensiones está formada por una sucesión de elementos *dimension*, que incluyen un nombre (atributo *name*) y una descripción (atributo *description*).

```
<dimensions>
    <dimension name="RH" description="Altura del carril"/>
    <dimension name="RAS_Arcelor" description="Asimetría Arcelor"/>
    <dimension name="RAS_l" description="Asimetría izquierda"/>
    <dimension name="RAS_r" description="Asimetría derecha"/>
    <dimension name="WT" description="Ancho del alma"/>
    <dimension name="FW" description="Ancho del pie"/>
</dimensions>
```

Figura 17: Lista de dimensiones

4.2.1.2. Perfiles

El elemento *profiles* contiene una lista de las imágenes de perfil que se pueden utilizar, y la posición de las dimensiones en cada una de ellas.

Por cada imagen de perfil existe un elemento *profile* en el que se especifica la ruta de la imagen empleada (atributo *image*), opcionalmente las reglas que han de cumplir los valores esperados de las dimensiones para utilizarla (elemento *rules*) y la lista de posiciones de las dimensiones (elemento *dimplacements*).

```
<profiles>
    <profile name="symmetric" image="base.png">
        <rules>
            <eq l="RAS_l" r="RAS_r" epsilon="2"/>
        </rules>
        <dimplacements>
            <dimplacement name="RH" x="15" y="53" angle="-90" image="1.png"/>
            <dimplacement name="FW" x="540" y="1015" image="2.png"/>
            <dimplacement name="WT" x="540" y="570" image="WT.png"/>
        </dimplacements>
    </profile>
    <!-- ... -->
</profiles>
```

Figura 18: Lista de imágenes de perfil

Cada posición (elemento *dimplacement*) tiene un nombre (atributo *name*), que ha de coincidir con uno de los definidos en la lista de dimensiones, y una posición en píxeles en la imagen (atributos *x* e *y*). Puede tener también un ángulo de inclinación en grados (atributo *angle*, por defecto 0) y una imagen (atributo *image*), que se superpondrá a la imagen base del perfil. Un ejemplo de esto puede verse en la Figura 19.

Figura 19: Imagen base, cota de la dimensión y combinación de ambas, de izquierda a derecha

La imagen de perfil que se muestra en la vista es la correspondiente al primer elemento que aparezca bajo *profiles* y cuyas dimensiones satisfagan las reglas que se le apliquen.

4.2.1.3. Reglas

El elemento *rules* del perfil admite una lista de reglas. Si los valores esperados de un carril concreto satisfacen todas las reglas que estén presentes para un perfil, ese perfil será el que se

muestre, salvo que otro que satisfaga todas sus reglas aparezca con anterioridad en el fichero de configuración.

Para cada regla han de especificarse dos valores (atributos *l* y *r*). Estos valores podrán ser numéricos, en cuyo caso se tomarán tal cual están, o bien identificadores de dimensiones que aparezcan en la lista correspondiente, en cuyo caso se empleará su valor esperado para el carril para el que se evalúen las reglas. Los tipos de reglas que pueden aparecer son:

- *eq*, que se cumple si *l* y *r* son iguales.
- *neq*, que se cumple si *l* y *r* son distintos.
- *lt*, que se cumple si *l* es menor que *r*.
- *nlt*, que se cumple si *l* no es menor que *r*.
- *gt*, que se cumple si *l* es mayor que *r*.
- *ngt*, que se cumple si *l* no es mayor que *r*.

eq y *neq* admiten además un atributo *epsilon*, que especifica un valor numérico que representa la distancia máxima que puede haber entre *l* y *r* para que puedan considerarse iguales.

El motor de reglas está implementado siguiendo la estructura de clases que puede verse en la Figura 20. La interfaz *DimensionRule* incluye un método *IsSatisfied*, que se emplea para comprobar si para un carril determinado se cumple una regla. Implementan esta interfaz *AndRule*, que se cumple si se cumple una lista de reglas al completo, *NotRule*, que se cumple si no se cumple una regla determinada, y *CmpRule*, que se cumple si se satisface una relación determinada entre dos valores. De CmpRule heredan *EqRule*, que se cumple si los valores son iguales (teniendo en cuenta un *epsilon*, como se explicó anteriormente), y *LtRule*, que se cumple si el primer valor es menor que el segundo.

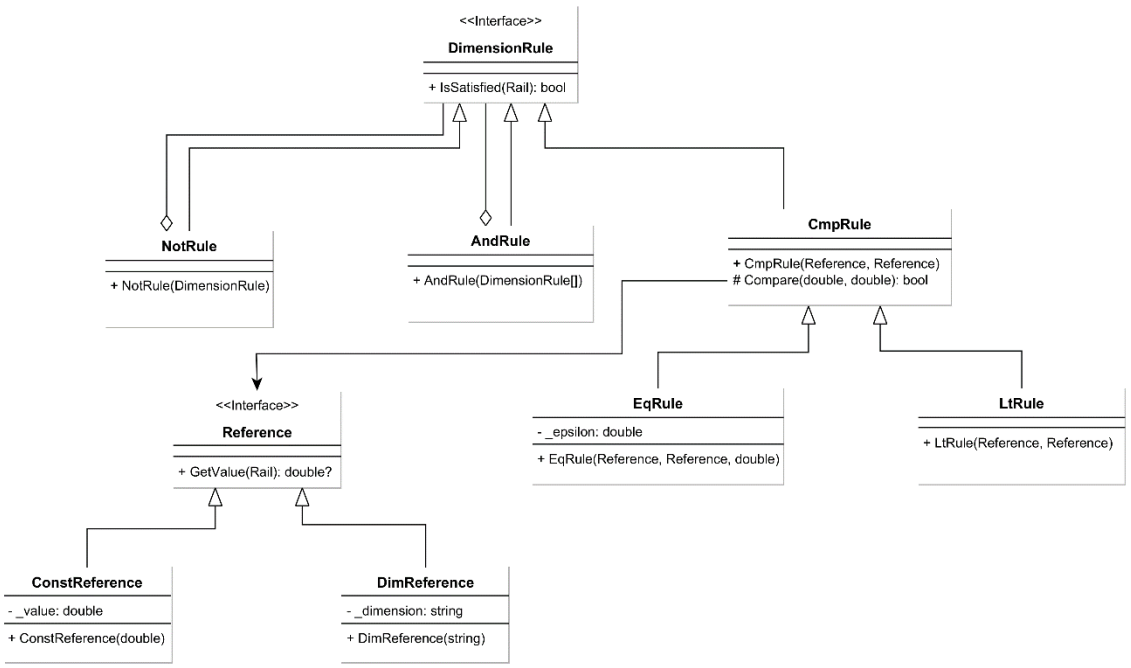


Figura 20: Diagrama de clases del motor de reglas

La interfaz *Reference* representa una referencia a un valor concreto. Esta referencia puede contener un número (clase *ConstReference*), en cuyo caso el valor es siempre un número determinado, independientemente del carril, o el nombre de una dimensión (clase *DimReference*), en cuyo caso el valor es el esperado para una dimensión concreta.

En la configuración por defecto, se emplea la imagen del carril simétrico cuando el valor esperado de las dimensiones RAS_l y RAS_r es el mismo, con un *epsilon* de 2 milímetros, como se muestra en la Figura 21.

```
<eq l="RAS_l" r="RAS_r" epsilon="2"/>
```

Figura 21: Regla empleada en el perfil simétrico

4.3. Colores

Los colores empleados para señalar si una medición se encuentra fuera o dentro de las tolerancias son los mismos para todas las vistas que muestran resultados gráficamente.

En notación hexadecimal estos colores son #E13D3D (rojo, para las mediciones que se encuentran fuera de las tolerancias) y #30B430 (verde, para las mediciones que se encuentran dentro). Los colores pueden verse en la Figura 22.

Figura 22: Colores utilizados para los resultados que se encuentran fuera y dentro de las tolerancias, respectivamente

Los colores se han escogido procurando que sean fácilmente distinguibles aún para personas que padezcan defectos de la vista como el daltonismo, sin renunciar por ello a los significados "tradicionales" de los colores rojo (incorrecto) y verde (correcto). Esto puede apreciarse en la Figura 23.

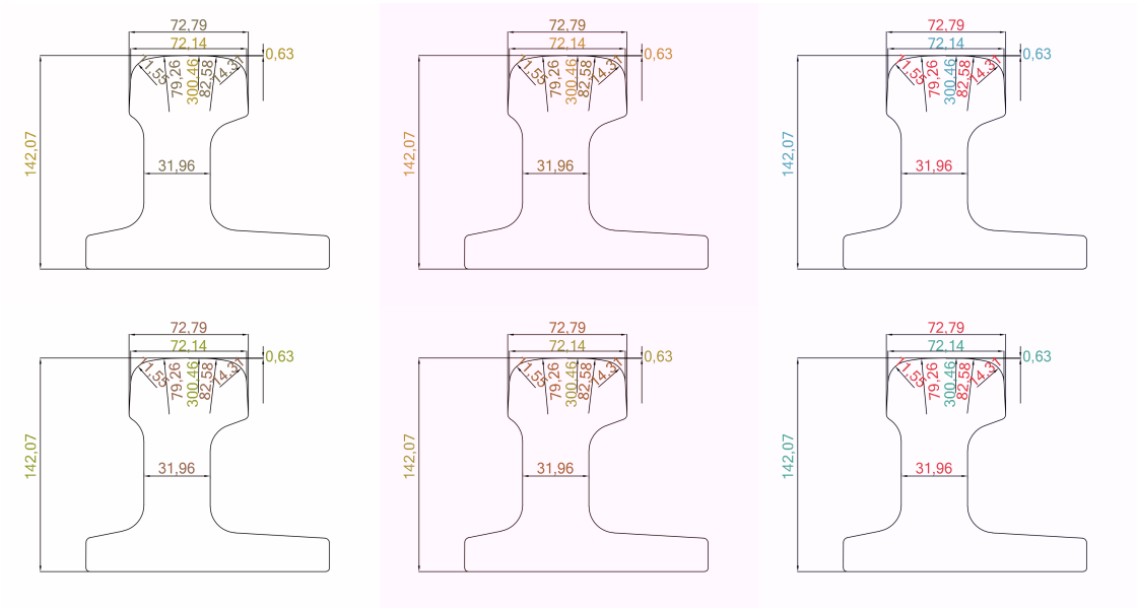


Figura 23: Vista de perfil (4.1.2.2), con los colores alterados para simular, de izquierda a derecha y de arriba abajo, protanopía, deuteranopía, tritanopía, protanomalía, deuteranomalía y tritanomalía

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

DOCUMENTO VII

PLANIFICACIÓN Y PRESUPUESTO

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Planificación

El reparto de tareas a lo largo del proyecto fue como se muestra en el diagrama de Gantt que puede verse en la Figura 1.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Análisis**	**30 días**	**lun 01/07/13**	**vie 09/08/13**
2	Análisis de requisitos	15 días	lun 01/07/13	vie 19/07/13
3	Estudio de software existente	10 días	lun 22/07/13	vie 02/08/13
4	Estudio de normas	5 días	lun 05/08/13	vie 09/08/13
5	**Diseño general**	**30 días**	**lun 12/08/13**	**vie 20/09/13**
6	División en componentes	10 días	lun 12/08/13	vie 23/08/13
7	Estructura del sistema de visión	20 días	lun 26/08/13	vie 20/09/13
8	**Programa configurador**	**85 días**	**lun 07/10/13**	**vie 31/01/14**
9	Diseño	5 días	lun 07/10/13	vie 11/10/13
10	Implementación	25 días	lun 30/12/13	vie 31/01/14
11	**Cálculo dimensional**	**65 días**	**lun 23/09/13**	**vie 20/12/13**
12	Diseño	10 días	lun 23/09/13	vie 04/10/13
13	Implementación	50 días	lun 14/10/13	vie 20/12/13
14	**Programa visualizador**	**25 días**	**lun 18/11/13**	**vie 20/12/13**
15	Diseño	5 días	lun 18/11/13	vie 22/11/13
16	Implementación	20 días	lun 25/11/13	vie 20/12/13
17	**Funcionalidad de visualización**	**35 días**	**lun 30/09/13**	**vie 15/11/13**
18	Diseño	10 días	lun 30/09/13	vie 11/10/13
19	Implementación	25 días	lun 14/10/13	vie 15/11/13
20	**Programa medidor**	**95 días**	**lun 23/12/13**	**vie 02/05/14**
21	Diseño	5 días	lun 23/12/13	vie 27/12/13
22	Implementación	20 días	lun 07/04/14	vie 02/05/14
23	**Funcionalidad de calibración**	**40 días**	**lun 06/01/14**	**vie 28/02/14**
24	Elaboración del algoritmo	30 días	lun 06/01/14	vie 14/02/14
25	Implementación	10 días	lun 17/02/14	vie 28/02/14
26	**Programa de comunicaciones**	**25 días**	**lun 03/03/14**	**vie 04/04/14**
27	Diseño	5 días	lun 03/03/14	vie 07/03/14
28	Implementación emulador PA	10 días	lun 10/03/14	vie 21/03/14
29	Implementación emulador PLC	10 días	lun 24/03/14	vie 04/04/14
30	Implementación cliente	20 días	lun 10/03/14	vie 04/04/14
31	Pruebas de medición	2 días	lun 05/05/14	mar 06/05/14
32	Integración	5 días	mié 07/05/14	mar 13/05/14
33	Instalación	10 días	mié 25/06/14	mar 08/07/14
34	Elaboración de memoria	30 días	mié 14/05/14	mar 24/06/14
35	Fin del proyecto	0 días	mar 08/07/14	mar 08/07/14

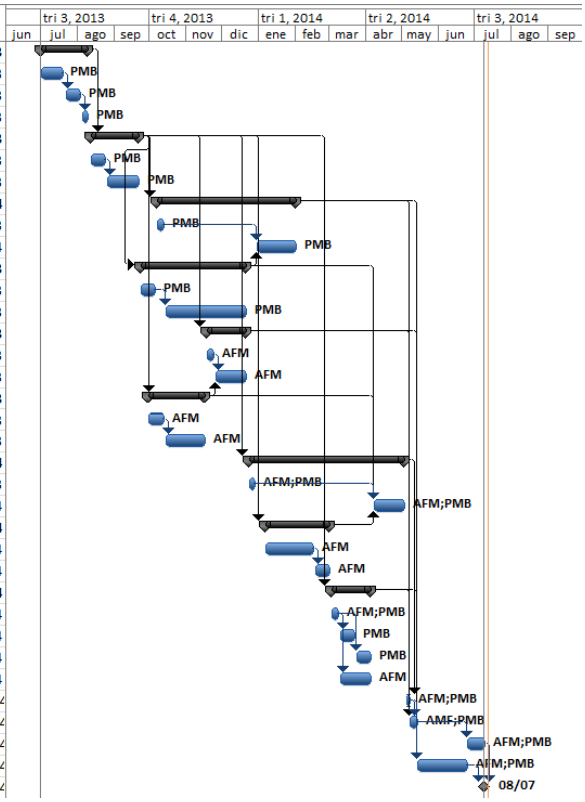


Figura 1: Diagrama de Gantt del proyecto

A continuación se describen las tareas que figuran en la anterior planificación:

- **Análisis:** Estudio de la documentación aportada por el cliente, ArcelorMittal, incluyendo el manual de usuario del sistema anterior (la PMG) y algunas de las normas que regulan la fabricación de carriles de tren, y averiguación de los requisitos del nuevo sistema.
- **Diseño general:** Diseño del sistema en conjunto, incluyendo su descomposición en programas y las relaciones entre ellos, y determinación del hardware necesario.
- **Programa configurador:** Desarrollo de la interfaz gráfica del programa configurador, que queda fuera del ámbito de este TFM.
- **Cálculo dimensional:** Desarrollo de la funcionalidad de cálculo de dimensiones, empleada en el programa configurador y en el medidor, que queda fuera del ámbito de este TFM.
- **Programa visualizador:** Desarrollo de los formularios del programa visualizador, que se describe en el documento VI - Visualizador.
- **Funcionalidad de visualización:** Desarrollo de la funcionalidad de visualización y los controles empleados por ella, que se utilizan en el programa visualizador y en el programa medidor, y que se describe en el documento VI - Visualizador.

- **Programa medidor:** Desarrollo de la interfaz gráfica y la estructura general del programa medidor, que se describe en el documento III - Medidor.
- **Funcionalidad de calibración**: Desarrollo de la funcionalidad de calibración presente en el programa medidor, que se describe en el documento V - Calibración.
- **Programa de comunicaciones:** Desarrollo del programa de comunicaciones que permite que el programa medidor se relacione con elementos externos, y que se describe en el documento IV - Comunicaciones.
- **Pruebas de medición**: Realización de pruebas de medición con segmentos de carril reales.
- **Integración:** Implementación de los cambios necesarios para el funcionamiento conjunto de todos los programas desarrollados, y pruebas de los mismos.
- **Instalación:** Despliegue de los programas desarrollados en el entorno de producción.
- **Elaboración de memoria**: Elaboración de la presente documentación.

2. Presupuesto

En esta sección del documento se recoge el presupuesto del proyecto que se ha llevado a cabo de acuerdo a la planificación de la sección 1.

2.1. Recursos humanos

La Tabla 1 muestra el presupuesto del proyecto para los recursos humanos.

Categoría	Número de personas	Nº horas	Precio unitario	Importe total
Programador	2	3000	12,00 €	36.000,00 €
			Total:	36.000,00 €

Tabla 1: Presupuesto para los recursos humanos del proyecto

2.2. Software para el despliegue

La Tabla 2 muestra el presupuesto del proyecto para los recursos software que se han utilizado para el proyecto.

Descripción	Unidades	Precio unitario	Importe total
MVTec Halcon	1	1.300,00 €	1.300,00 €
		Total:	1.300,00 €

Tabla 2: Presupuesto para los recursos software utilizados para el despliegue del proyecto

2.3. Recursos para el desarrollo

La Tabla 3 muestra el presupuesto del proyecto para los recursos utilizados para llevar a cabo el despliegue del proyecto.

Descripción	Unidades	Precio unitario	Amortización	Importe amortizado	Importe total
Computador personal	2	600,00 €	25%	150,00 €	300,00 €
Windows 7 Professional	2	200,00 €	25%	50,00 €	100,00 €
Microsoft office 2013	2	219,00 €	25%	54,75 €	109,50 €
Visual studio	2	616,00 €	25%	154,00 €	308,00 €
				Total:	817,50 €

Tabla 3: Presupuesto para los recursos hardware externo utilizados para el proyecto

Se ha añadido una columna para la amortización, la cual se ha calculado teniendo en cuenta la vida útil de los recursos, que suele ser de aproximadamente cuatro años.

Como el proyecto ha durado un año, la amortización ha de ser de un 25%.

2.4. Despliegue del proyecto

La Tabla 4 muestra el presupuesto del proyecto para los recursos hardware que van a formar parte del prototipo del sistema.

Descripción	Unidades	Precio unitario	Importe total
Cámaras Genie HM1400	4	2.318,00 €	9.272,00 €
Fuente Aliment. 12V Cámara	4	33,00 €	132,00 €
Cable alimentación (10 m.) + Trigger	4	160,00 €	640,00 €
Objetivo Goyo de 16mm	4	461,00 €	1.844,00 €
Filtro pasabanda para láser	4	87,00 €	348,00 €
Láser Stingray de 50 mW	4	482,00 €	1.928,00 €
Fuente de alimentación para Láser	4	37,00 €	148,00 €
Cable de alimentación para Láser	4	42,00 €	168,00 €
Cabezal óptico para láser Stingray(Convierte el haz láser en una línea con apertura de 30º)	4	168,00 €	672,00 €
Cable ethernet para conectar cámaras	4	12,00 €	48,00 €
Computador de visión	1	1.500,00 €	1.500,00 €
Tarjeta ethernet Intel i350 de 4 puertos	1	255,00 €	255,00 €
Perfiles para sujetar cámaras y láseres	1	135,00 €	135,00 €
Bastidor de aluminio para los láseres	1	750,00 €	750,00 €
		Total:	17.840,00 €

Tabla 4: Recursos hardware utilizados para llevar a cabo la construcción del prototipo del sistema

En la Tabla 4 se muestran los siguientes apartados:

- En color amarillo las cámaras y los componentes de las mismas.
- En color rojo los láseres y los componentes de los mismos.
- En color verde el computador de visión así como sus componentes principales y elementos de interconexión.
- En color gris la estructura de soporte del prototipo.

2.5. Resumen del presupuesto

La Tabla 5 se muestra un resumen de las secciones anteriores.

Nombre de la categoría	Importe
Recursos humanos	36.000,00 €
Software de terceros	1.300,00 €
Desarrollo	817,50 €
Despliegue del proyecto	17.840,00 €
Beneficio industrial (15 %)	8.393,63 €
Total	64.351,13 €
IVA (21%)	13.513,74 €
Total con IVA	77.864,86 €

Tabla 5: Resumen del presupuesto del proyecto

El precio del proyecto es de 77.864,86 € con el IVA incluido.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ANEXO A

ELECCIÓN DEL SISTEMA DE VISIÓN

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Determinación de la zona de visualización de las cámaras

Para determinar la zona de visualización se parte del plano de sistema antiguo de medición mostrado en la Figura 1.

En el plano se puede ver como los ejes de los 4 láseres coinciden en un punto situado 85 mm por encima del plano sobre el que se desplazan los carriles (coordenada vertical) y centrado en el eje de todo el soporte mecánico (coordenada horizontal).

En el plano también se puede observar que los ejes ópticos de las 4 cámaras (representados en color azul en la Figura 1) también coinciden en un mismo punto del plano definido por los 4 láseres. Este punto está situado 138 mm por encima del plano de desplazamiento del carril.

El diseño del nuevo medidor se basará en esta geometría sin modificación alguna.

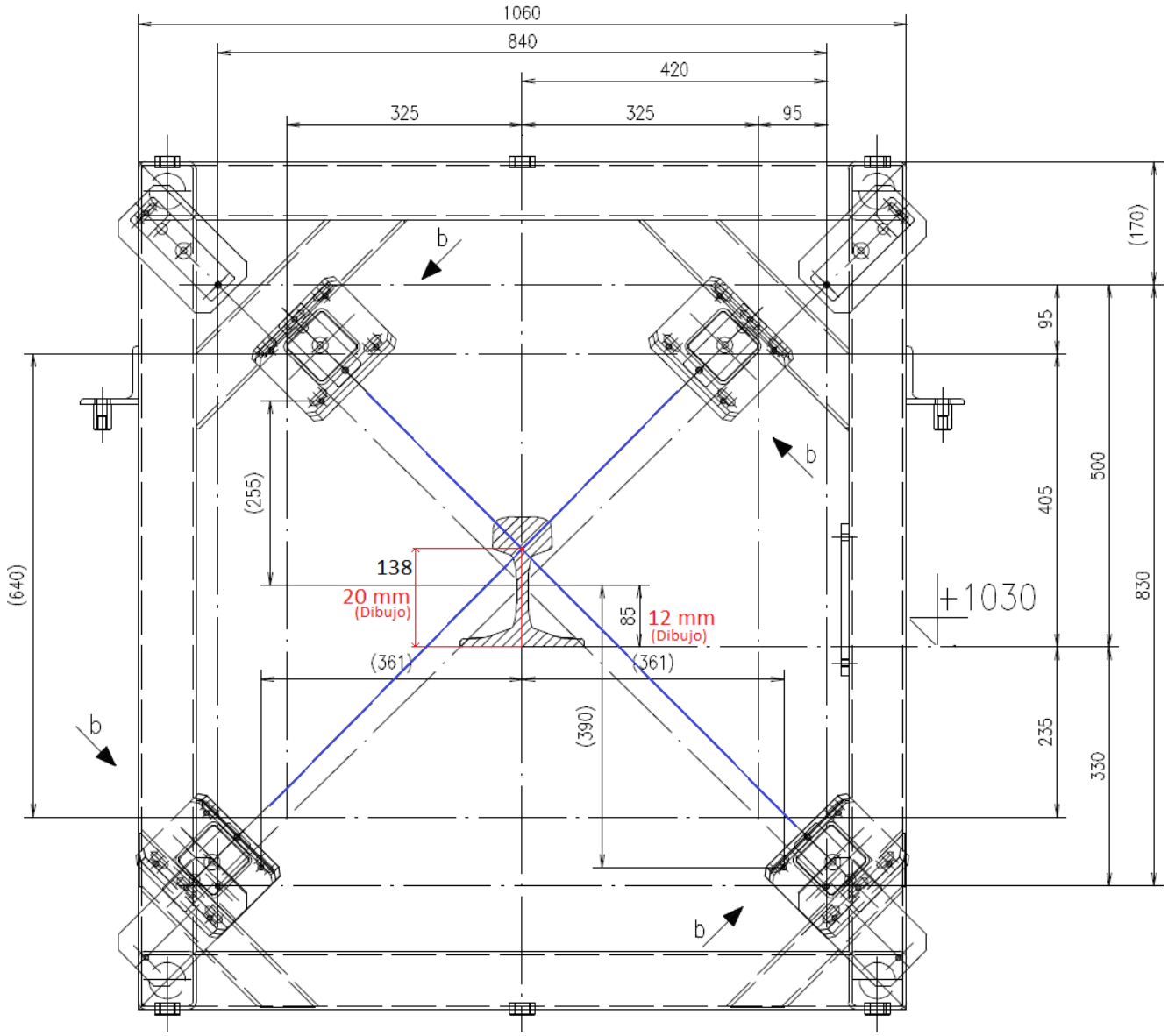


Figura 1: Plano físico

Para determinar la sección del carril que deben "ver" las cámaras se utiliza el siguiente procedimiento con el dibujo de cada perfil usando el programa AutoCad.

Se trazan dos rectas (azules a trazos) que pasan por el punto donde coinciden los ejes ópticos de las cámaras a 45º cada una con el eje horizontal, pues las cámaras están inclinadas 45º respecto al eje horizontal.

Se trazan dos líneas paralelas rojas a cada una de estas dos rectas azules que sean tangentes al perfil, tal como se puede observar en la Figura 2.

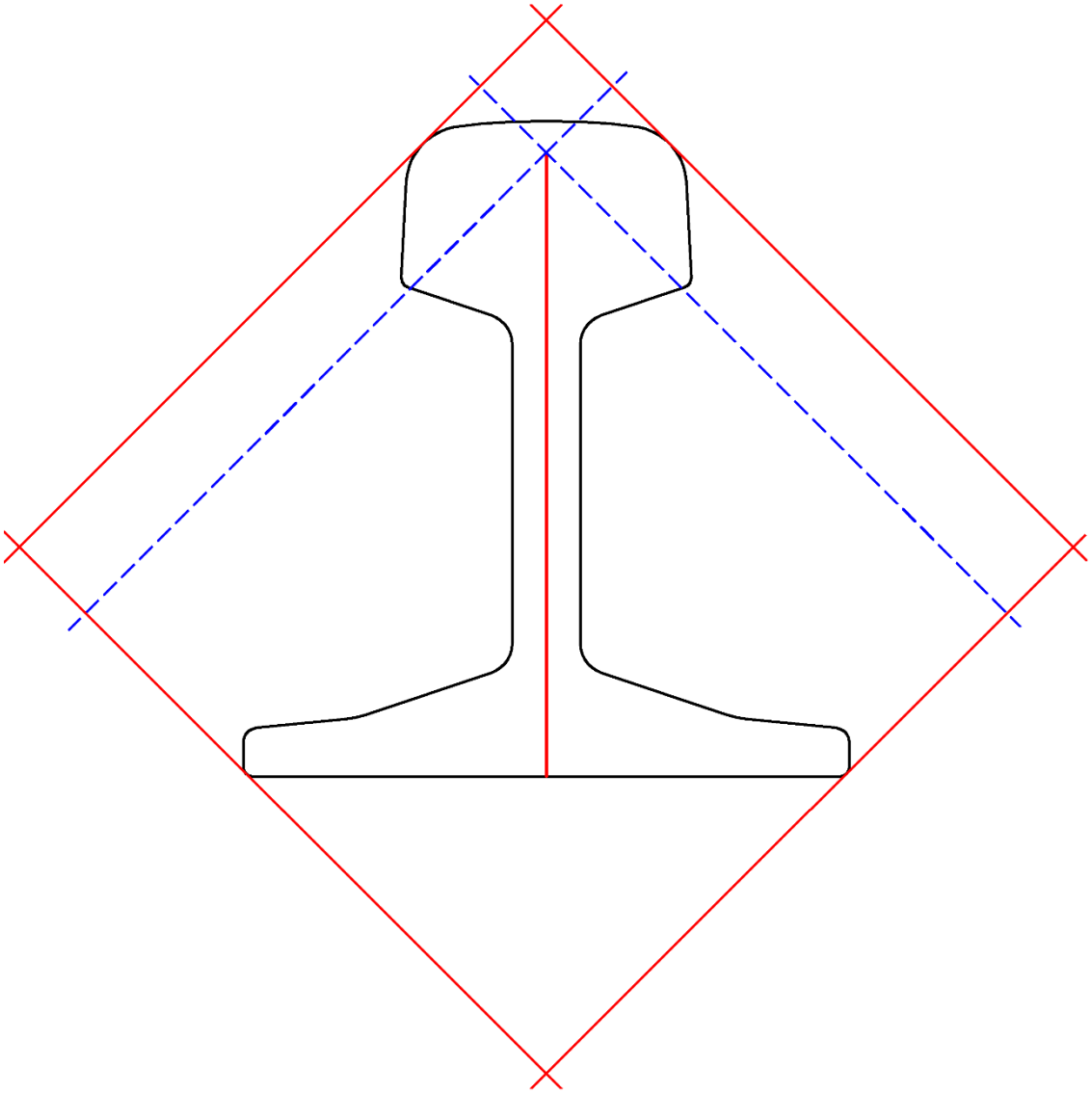


Figura 2: Paralelas tangentes al perfil

Las cuatro rectas rojas definen el rombo que circunscribe a la sección del carril. Se calcula el rombo definido en la Figura 2 para todos los perfiles de los que se tiene la descripción en formato “DXF”, esto se documenta en la sección siguiente.

2. Estudio de los perfiles

Con el objetivo de establecer las cámaras a adquirir para la realización de este proyecto, se va a llevar a cabo un estudio de los perfiles que se han conseguido en formato de AutoCAD (“DXF” o “DWG”), estos perfiles son los siguientes:

- Norma EN-13674: 46 E2, 50 E6, 54 E4, 60 E2, 60 E1 A4, 60 E1 T2, S 49.
- Norma UIC: UIC 54, UIC 60.
- Norma AREMA: 115 RE, 136 RE, TR 45.
- Norma GOST: R 65.
- Norma Australiana: A 60, A 69, A73.
- Norma Británica: BS 80, BS 90, BS 100.

Los carriles estudiados se muestran en las Figura 3 a la Figura 22.

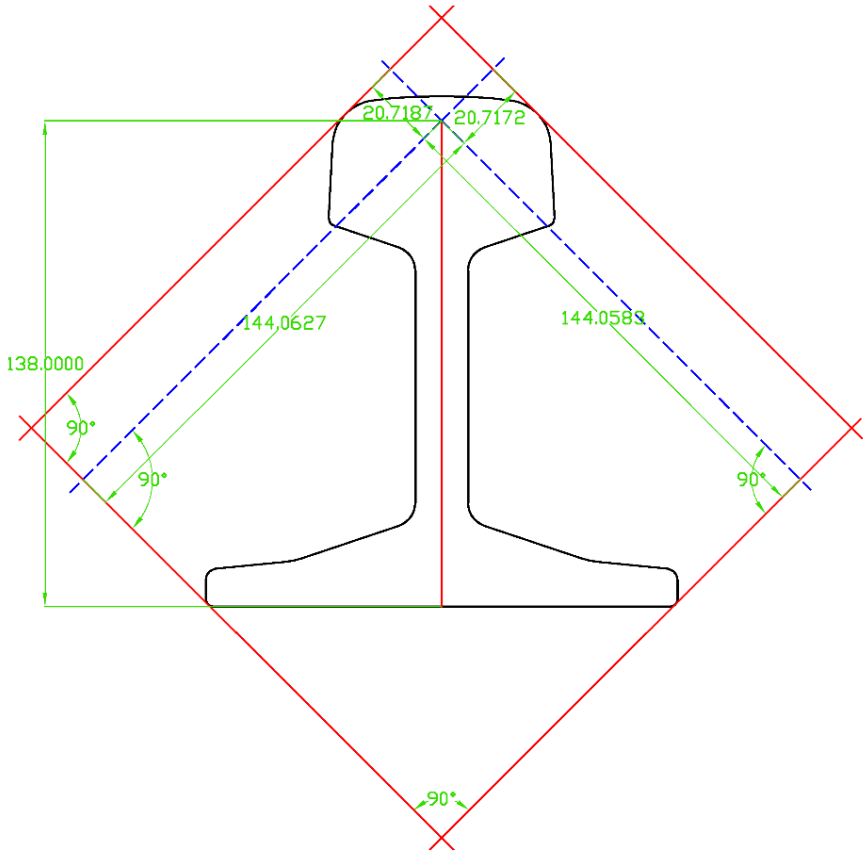


Figura 3: Modelo 46 E2

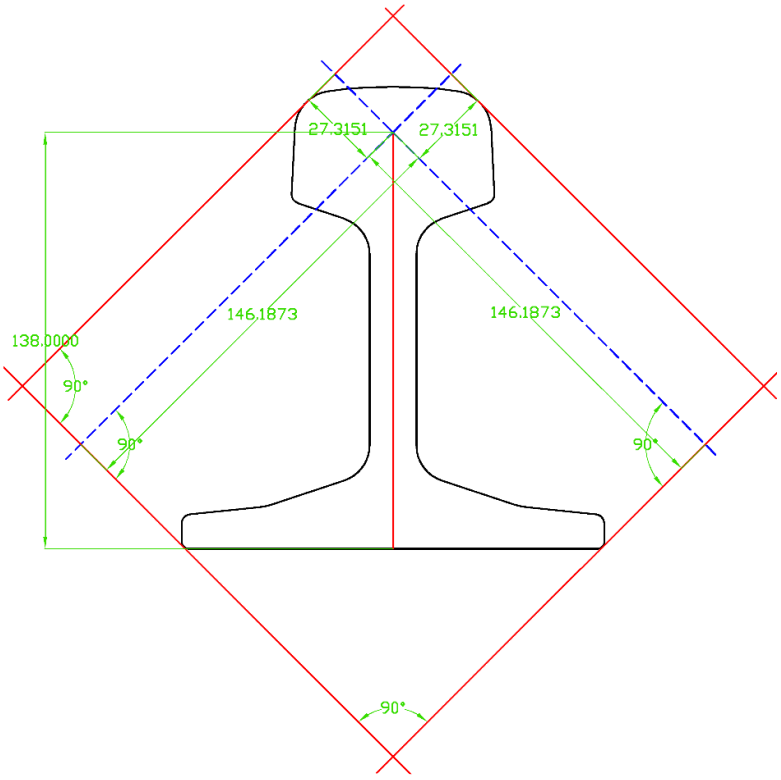


Figura 4: Modelo 50 E6

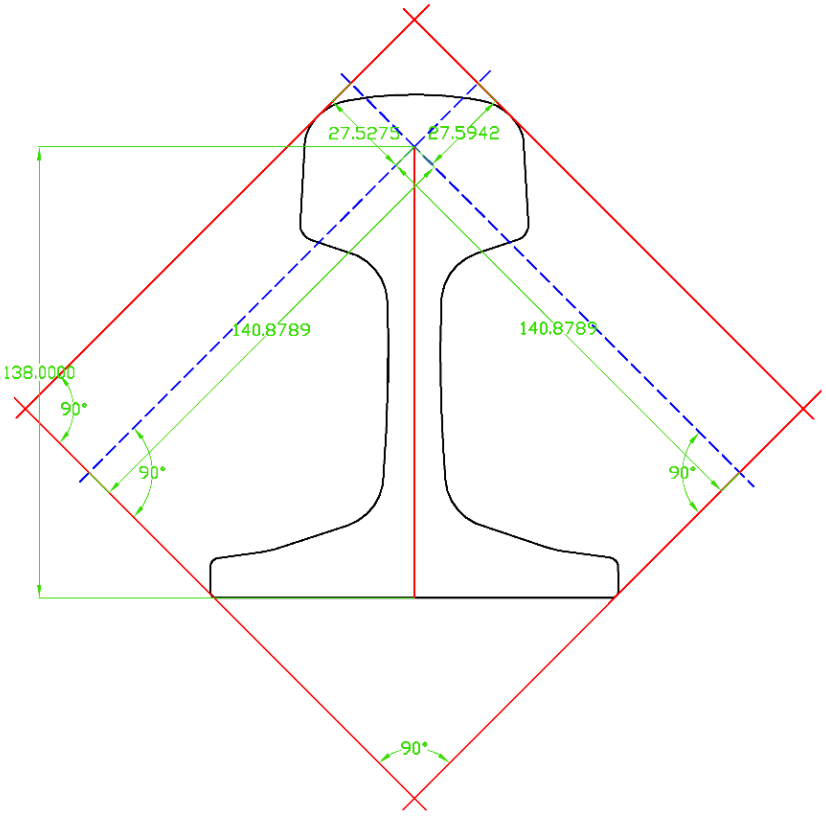


Figura 5: 54 E4

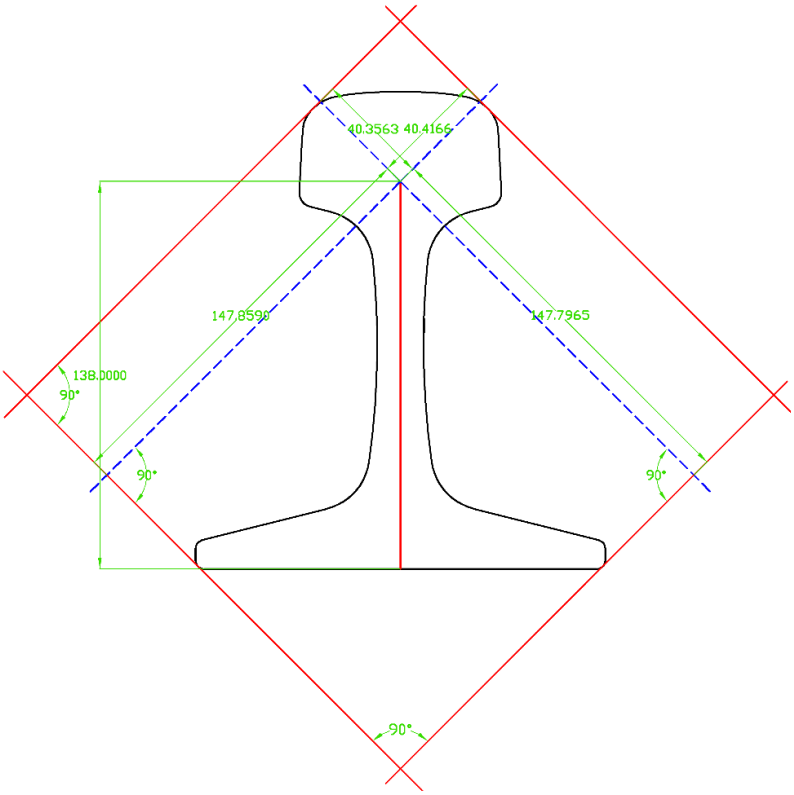


Figura 6: AS-60

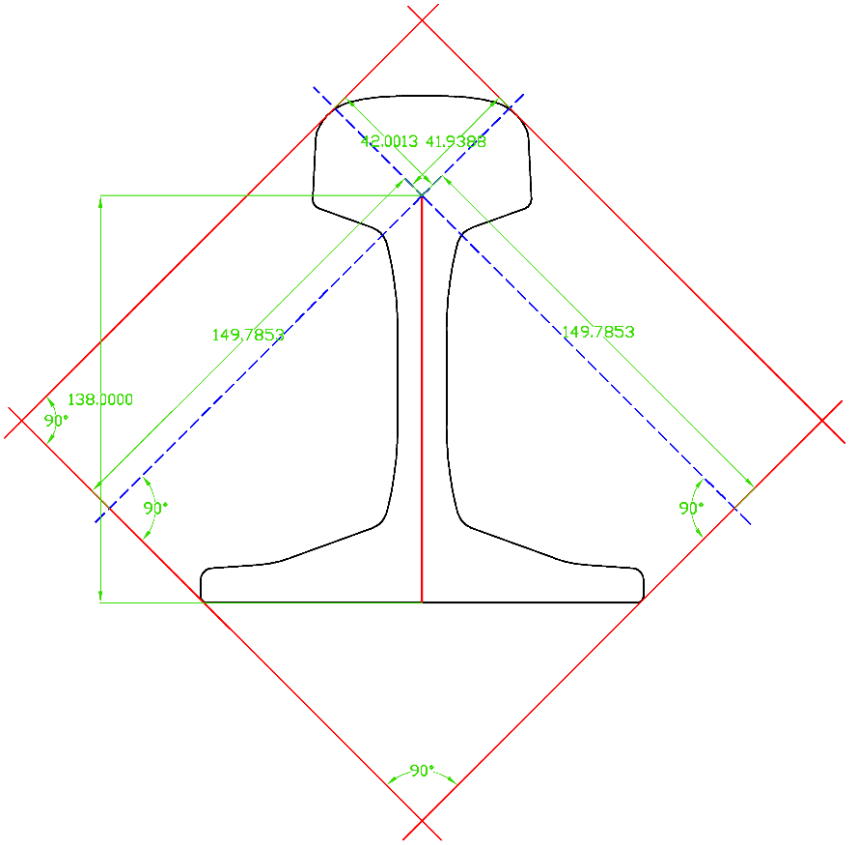


Figura 7: 60 E2

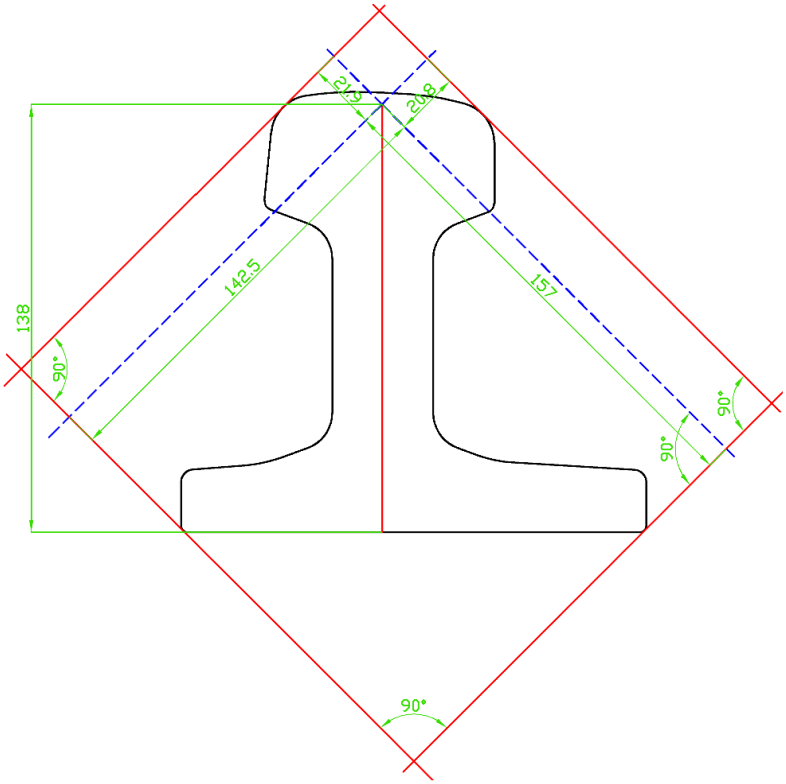


Figura 8: 60 E1 A4

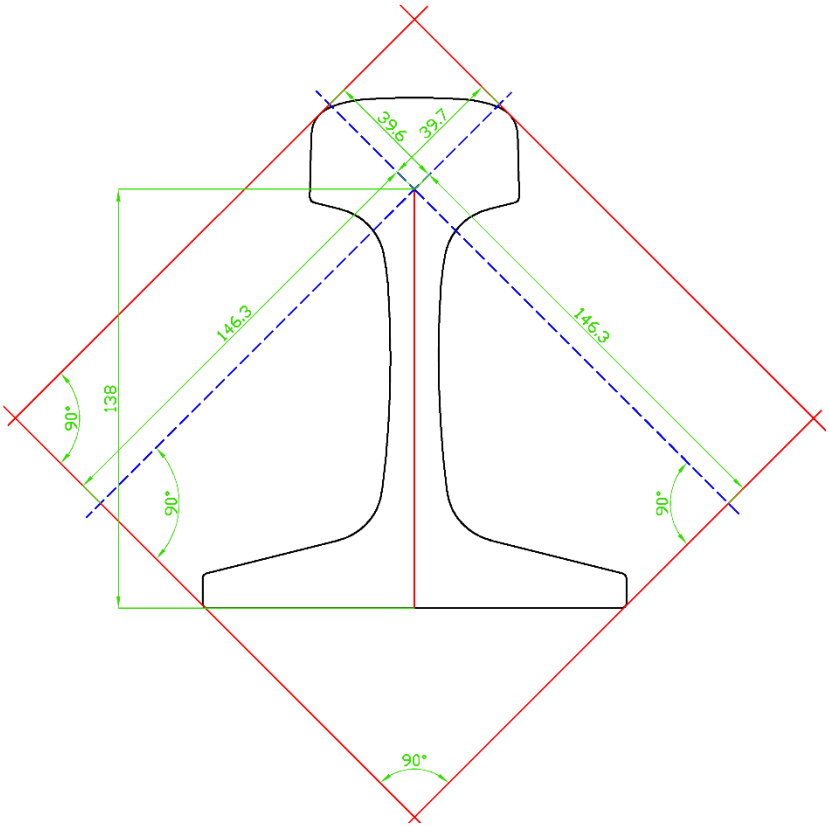


Figura 9: 115 RE

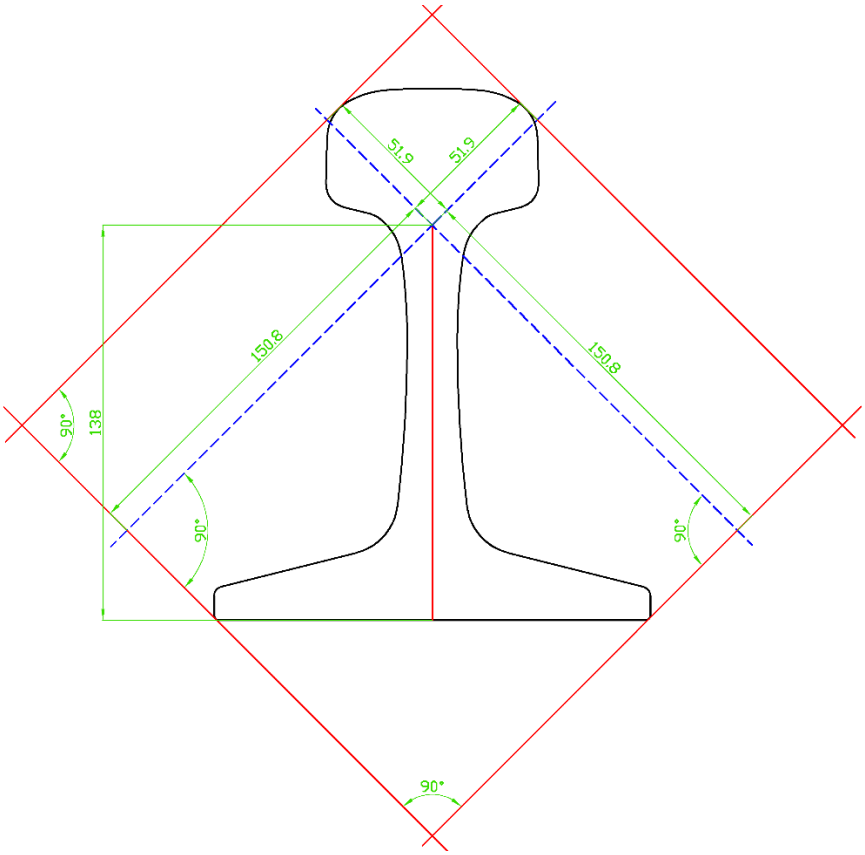


Figura 10: 136 RE

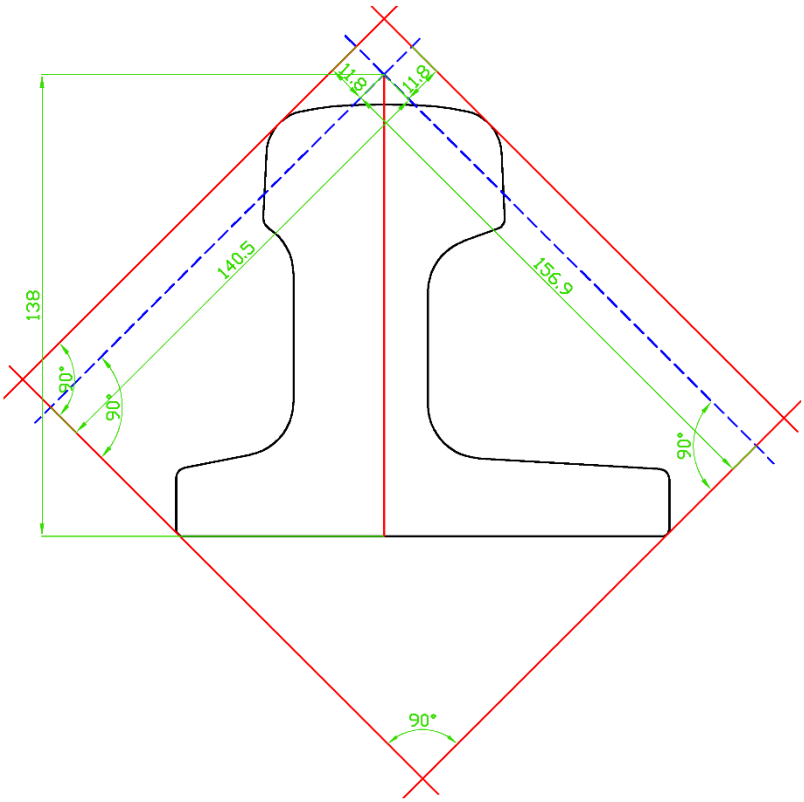


Figura 11: A69

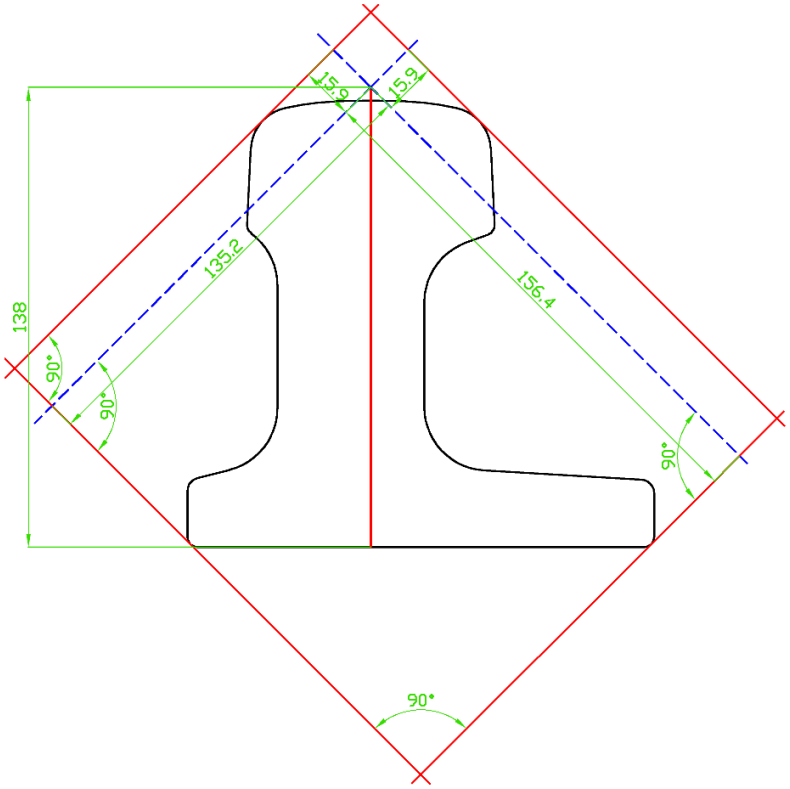


Figura 12: A 73

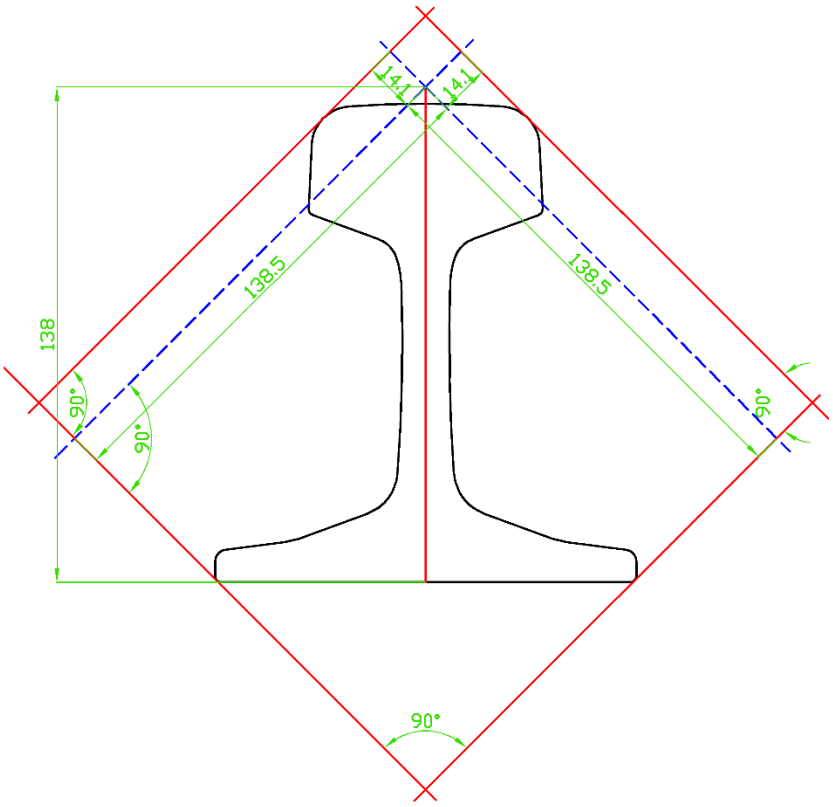


Figura 13: BS 80

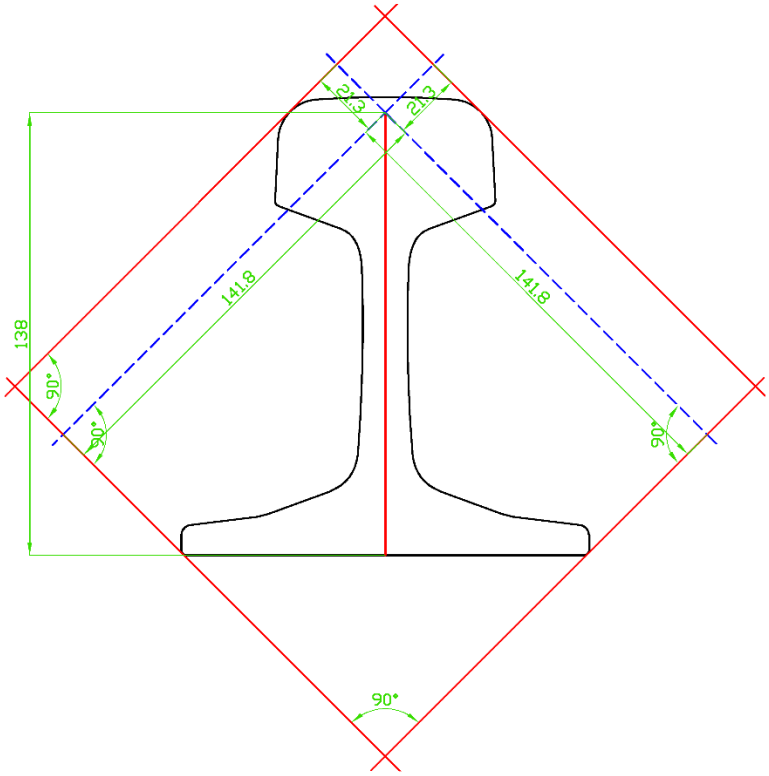


Figura 14: BS90

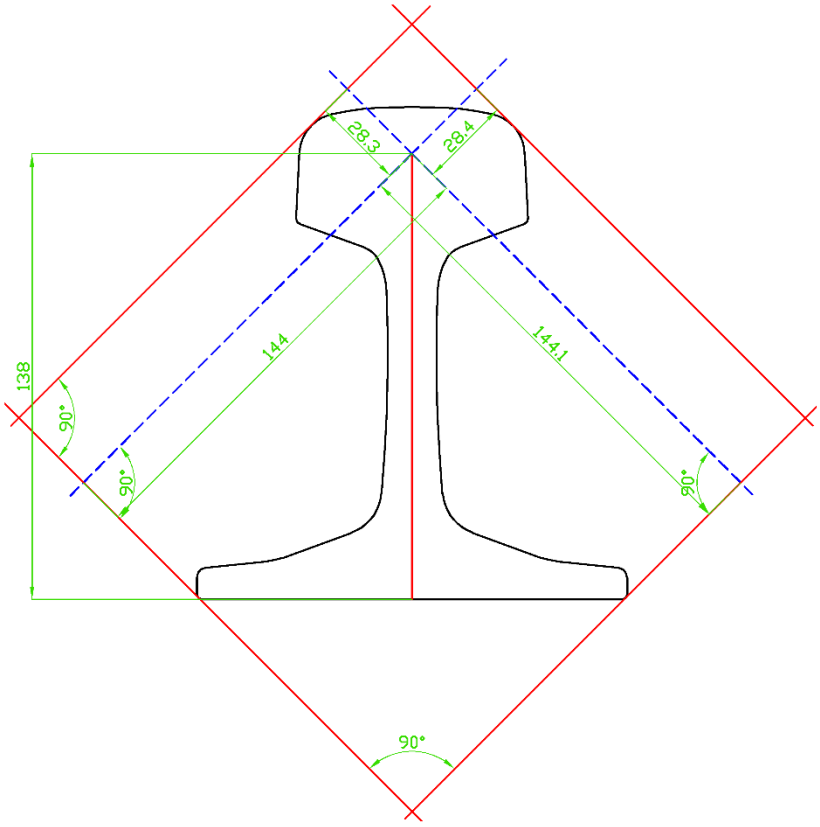


Figura 15: BS 100

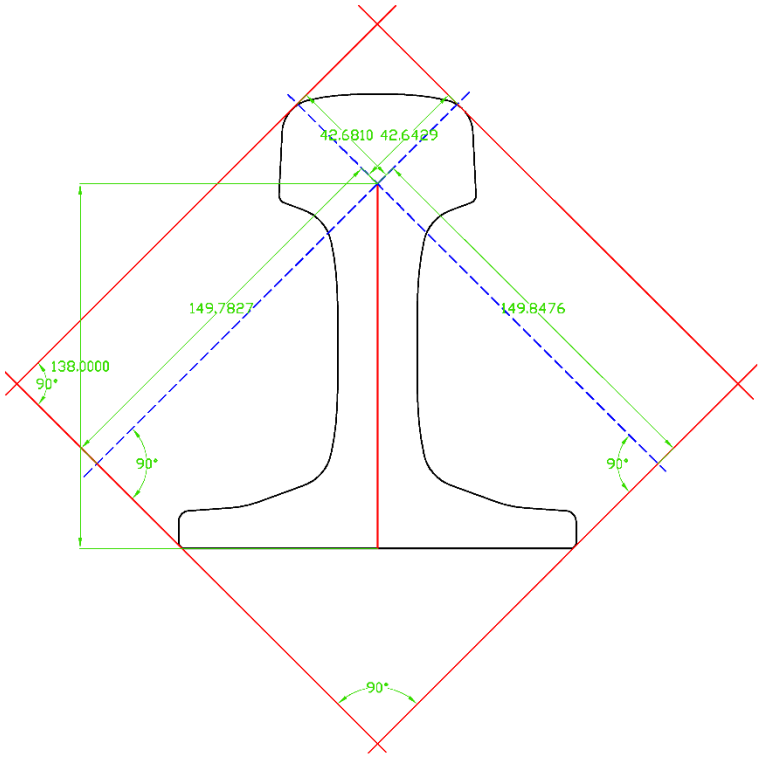


Figura 16: 60 E1 T12

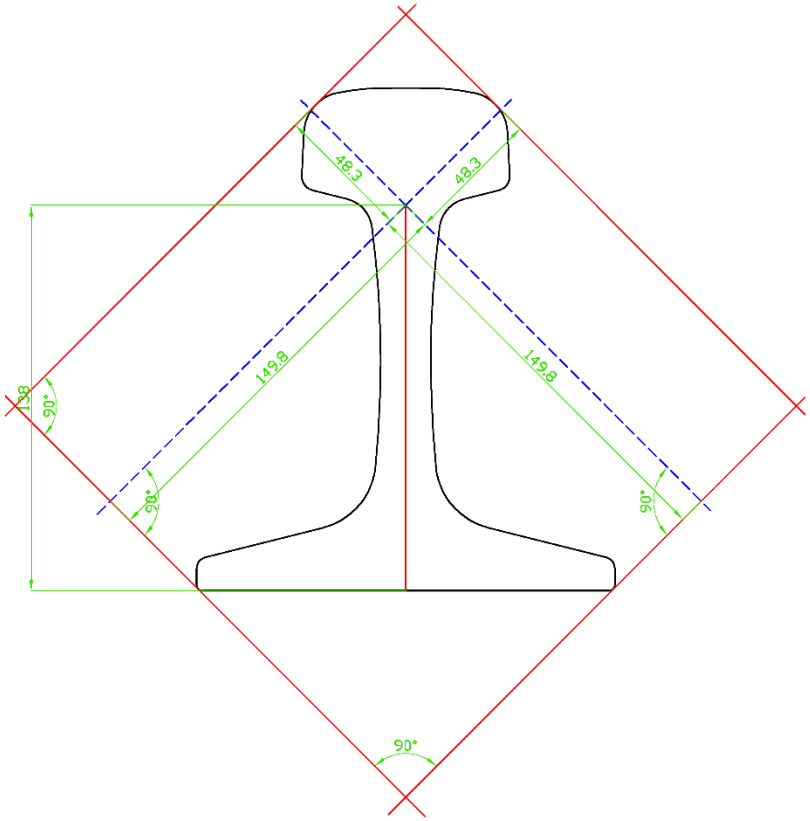


Figura 17: R65 GOST

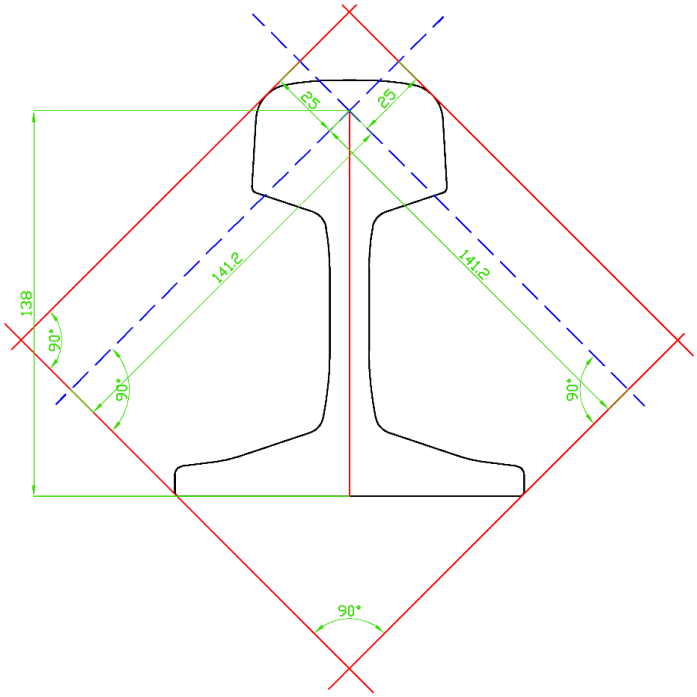


Figura 18: S49

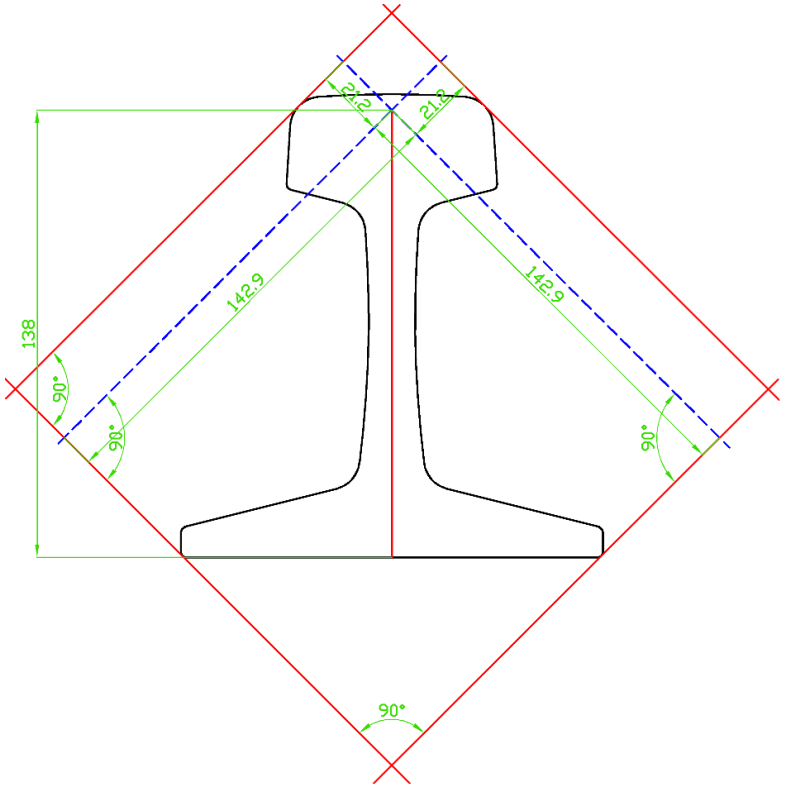


Figura 19: TR 45

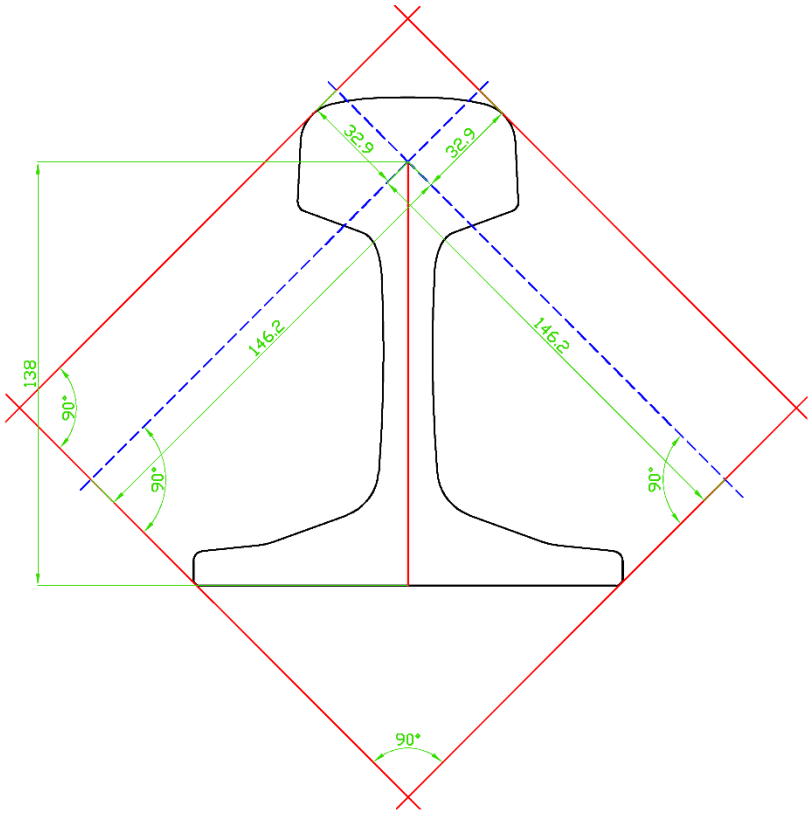


Figura 20: UIC 54

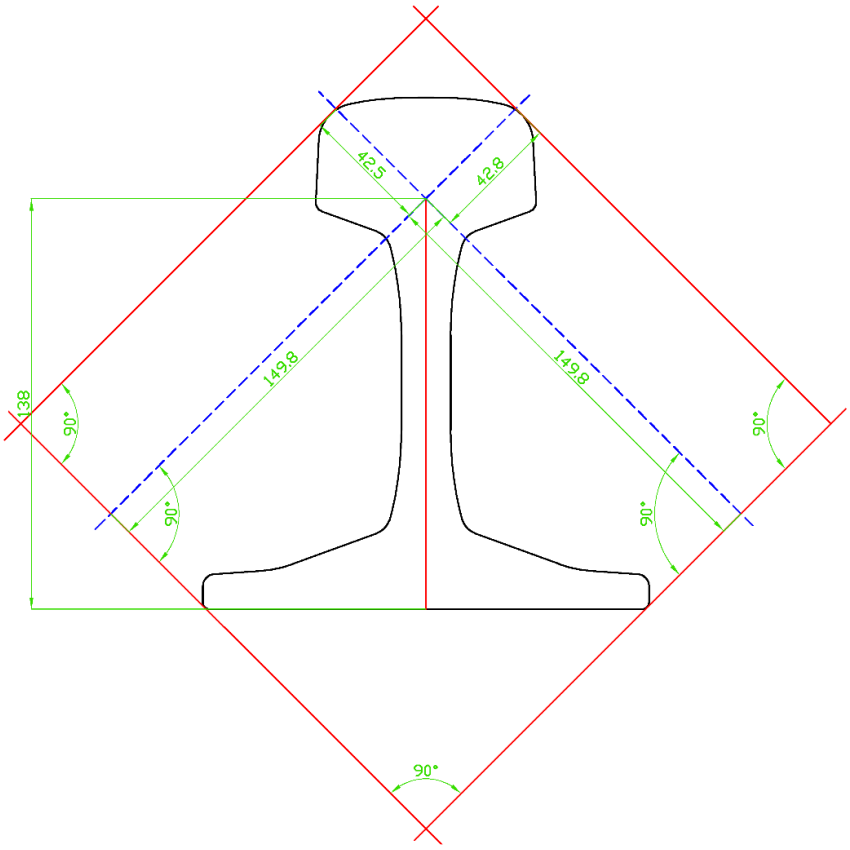


Figura 21: UIC 60

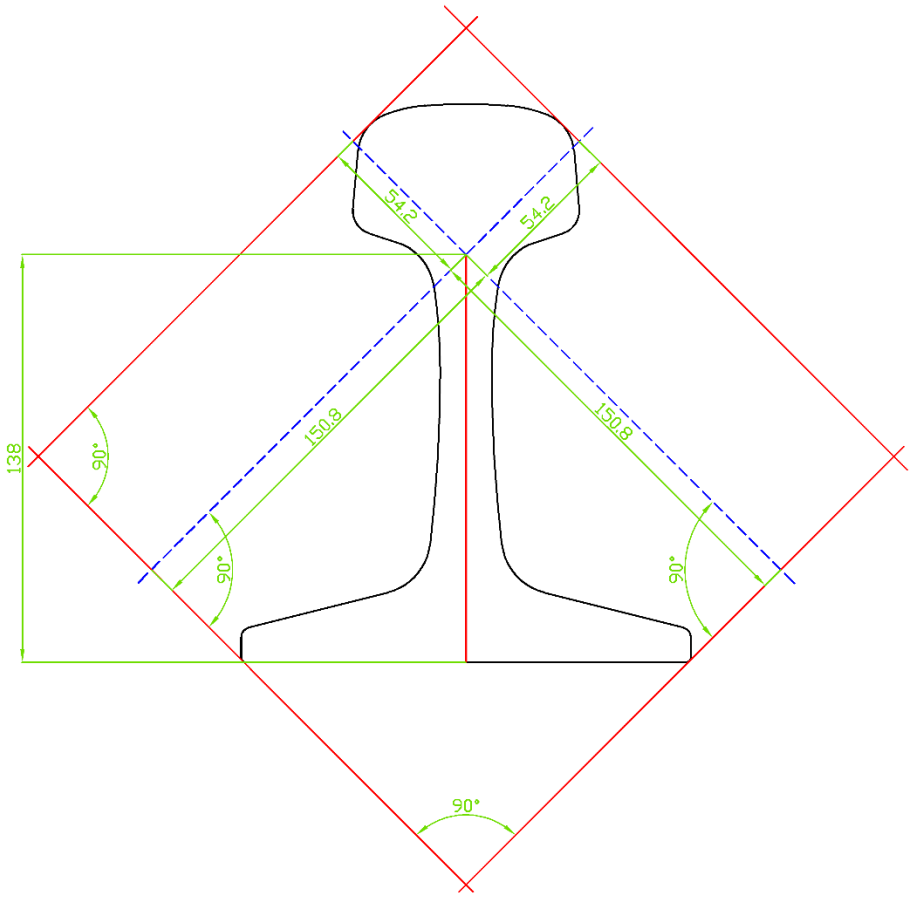


Figura 22: Rail 141

3. Cálculo de variables para elección de la óptica

De las figuras antes mostradas se han extraído una serie de datos, con la finalidad de calcular la zona que las cámaras deben ver, para estos las variables calculadas se corresponden al esquema de la Figura 23.

Se van a calcular, a partir de un punto que está a 138 mm en el eje de simetría (valor calculado a partir de los planos conocidos del sistema antiguo, tal como se explica en *"Determinación de la zona de visualización de las cámaras"*), por el cual se ha hecho pasar un aspa la cual está orientada a 45º con respecto al eje horizontal, en la Figura 23 se ha marcado este aspa de color azul y línea discontinua.

A partir de esa aspa se han trazado dos paralelas a cada línea (en la figura se muestran como líneas rojas) y se han hecho que estas sean tangentes al perfil tal como se puede observar en la Figura 23.

Por último se ha calculado la distancia del punto situado a 138 mm del eje de simetría hasta cada punto de tangencia entre el perfil y las líneas paralelas (líneas rojas)

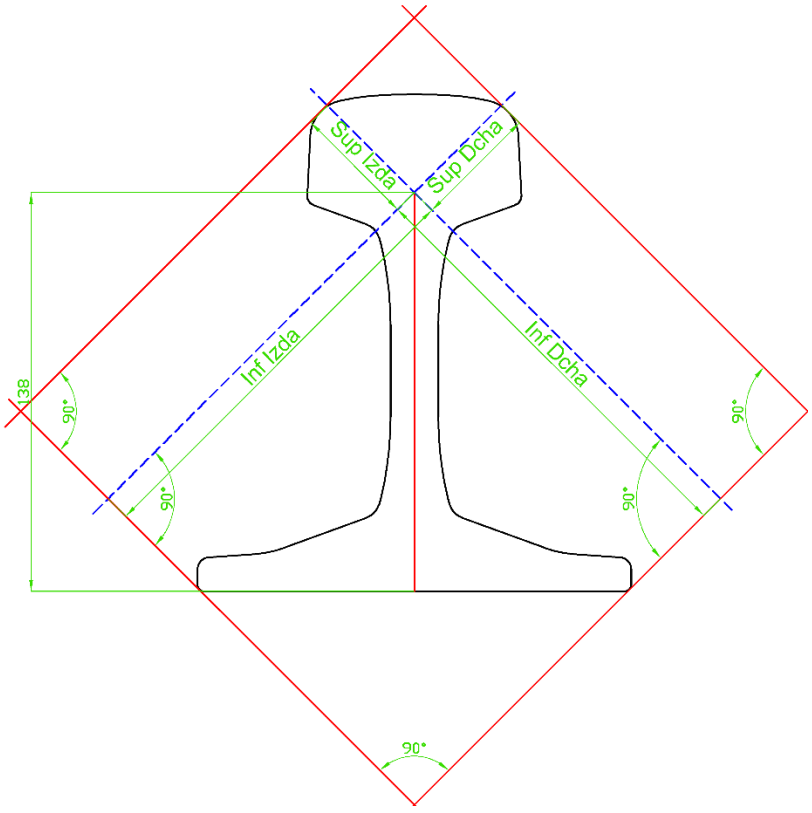


Figura 23: Esquema de variables

Se han obtenido por lo tanto para cada perfil de los que se tienen su descripción en un fichero de formato "DXF" de AutoCAD, las cuatro variables que se muestran en la figura anterior (Figura 23), es decir "SupIzda", "SupDcha", "InfIzda", "InfDcha".

Las variables que se van a calcular, es decir, las de la Figura 23, se obtendrán con el fin de calcular la apertura necesaria de los objetivos de las cámaras y demás variables de las mismas, en la Figura 24.

Estas cámaras se enfocarán al punto donde se cortan las aspas de color azul y línea discontinua de la Figura 24, este punto estará a una distancia de 138 mm en el eje de simetría del perfil, tal como se explicó para la Figura 23.

Por lo tanto cada cámara deberá "ver" la arista de color rojo que tiene más próxima, que es exactamente las variables que se desea calcular.

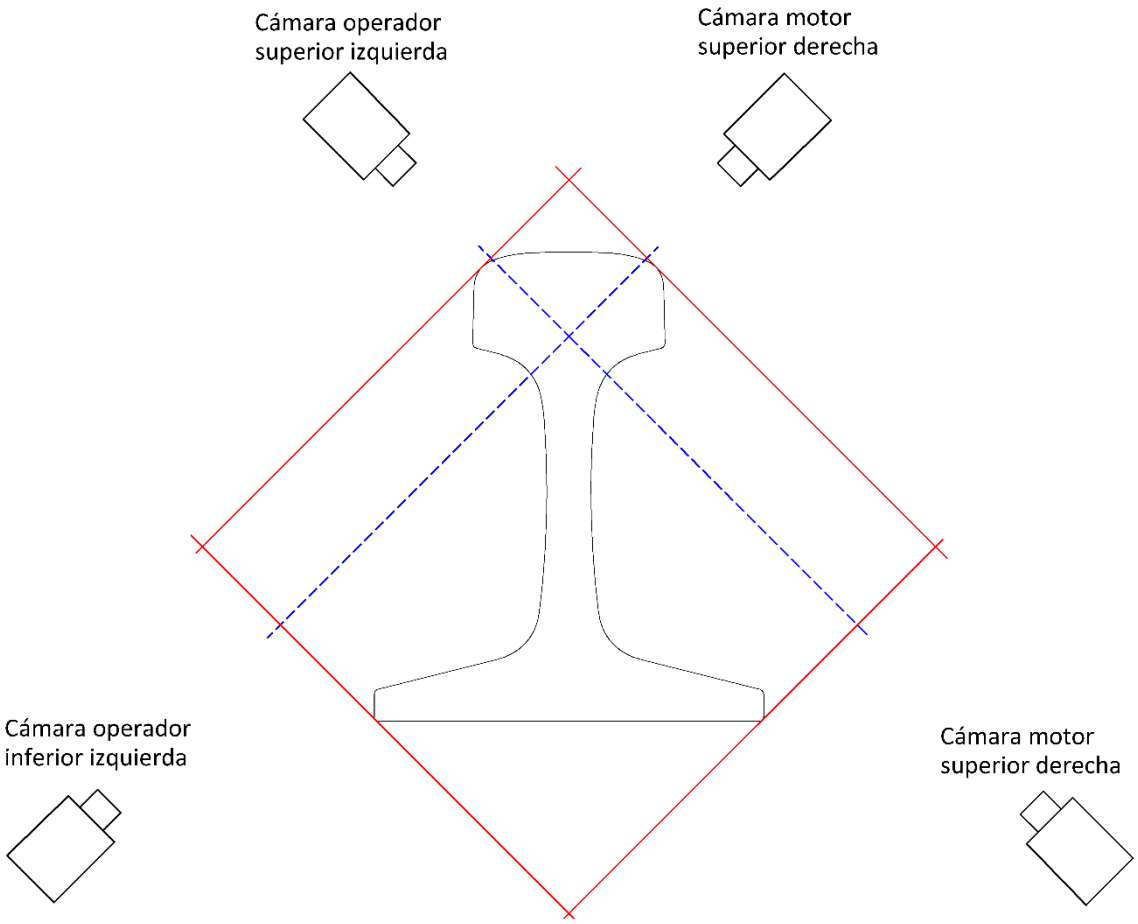


Figura 24: Esquema de cámaras

En la siguiente tabla se muestran para todos los perfiles de los raíles las cuatro variables indicadas en la Figura 23.

Nombre	Sup Izda	Sup Dcha	Inf Izda	Inf Dcha
46 E2	20,72	20,72	144,06	144,06
50 E6	27,32	27,32	146,19	146,19
54 E4	27,52	27,59	140,88	140,88
60 AUSTRALIA	40,36	40,42	147,86	147,8
60 E2	42	41,93	149,79	149,79
60 E1 A4	21,9	20,8	142,5	157
115	39,6	39,7	146,3	146,3
136	51,9	51,9	150,8	150,8
A 69	11,8	11,8	140,5	156,9
A 73	15,9	15,9	135,2	156,4
BS 80	14,1	14,1	138,5	138,5
BS 90	21,3	21,3	141,8	141,8
BS 100	28,3	28,4	144	144,1
60 E1 T2	42,68	42,64	149,78	149,84
R 65 GOST	48,3	48,3	149,8	149,8
S 49	25	25	141,2	141,2
TR 45	21,2	21,2	142,9	142,9
UIC 54	32,9	32,9	146,2	146,2
UIC 60	42,5	42,8	149,8	149,8
MAXIMOS:	51,9	51,9	150,8	157

Tabla 1: Variables de raíles para obtención de la óptica

Estos valores obtenidos se utilizarán para, como se ha comentado anteriormente, calcular la zona de visión de las cámaras y así poder elegir de forma correcta la óptica a utilizar.

3.1. Cálculo de variables para la cámara superior derecha

La cámara superior derecha deberá visualizar la zona que se puede observar en la figura 24.

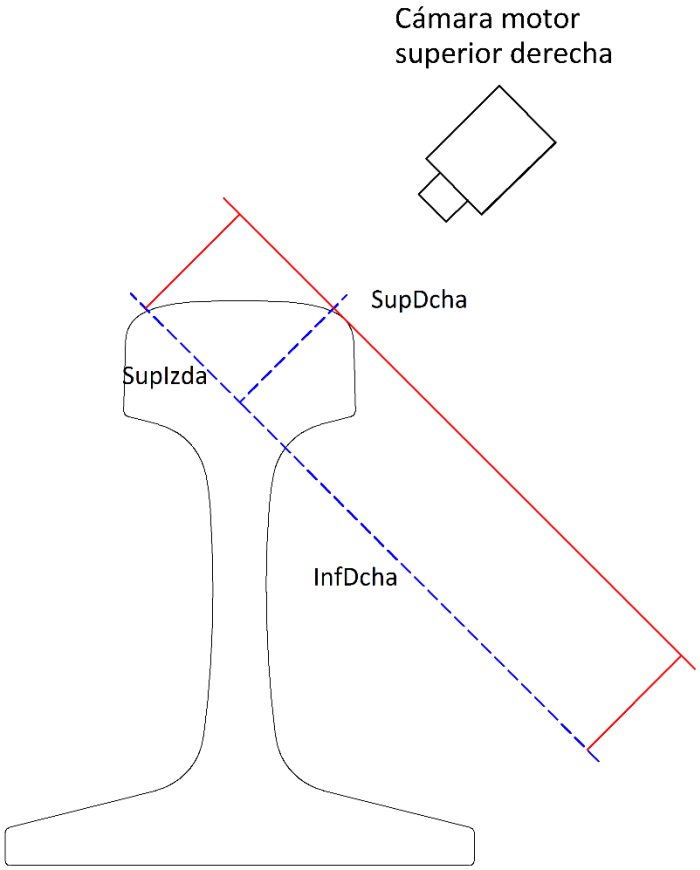


Figura 25: Zona de visualización de la cámara superior derecha

El máximo valor para la variable "SupDcha", la cual se puede observar en la tabla 1, es de 51.9 mm, por lo tanto, la cámara deberá visualizar el doble del valor de esa variable más un porcentaje de seguridad, que será de un 20%, por lo tanto:

$$51.9 \times 2 \times 1.2 = 124.56\, mm$$

Debemos calcular el valor máximo de las variables "SupIzda" y "InfDcha", para calcular el otro lado del rectángulo que la cámara debe visualizar, para ello, los valores máximos de las variables antes comentadas son, respectivamente, 51.9 mm y 157.0 mm, por lo que nos quedamos con el valor mayor, en este caso con InfDcha, es decir con, 157.0 mm y aplicamos el mismo procedimiento que antes:

$$157.0 \times 2 \times 1.2 = 376.8\, mm$$

El área de visión de la cámara superior derecha será de 124.56 mm x 376.8 mm

3.2. Cálculo de variables para la cámara superior izquierda

La cámara superior izquierda deberá visualizar la zona que se puede observar en la figura 25.

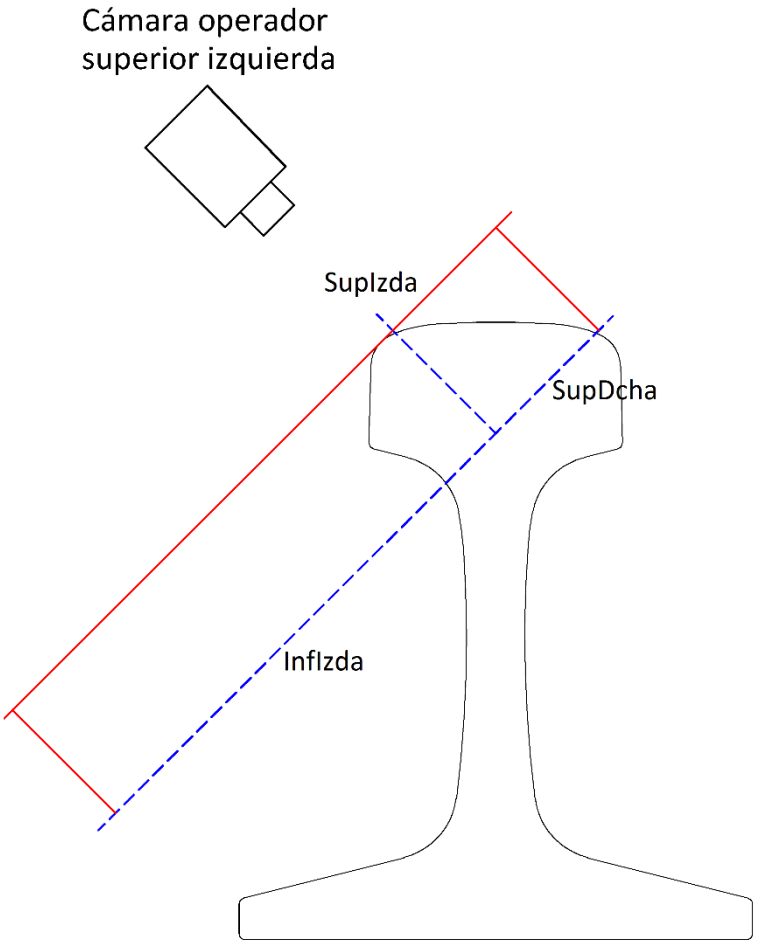


Figura 26: Zona de visualización de la cámara superior izquierda

El máximo valor para la variable "SupIzda", la cual se puede observar en la tabla 1, es de 51.9 mm, por lo tanto, la cámara deberá visualizar el doble del valor de esa variable más un porcentaje de seguridad, que será de un 20%, por lo tanto:

$$51.9 \times 2 \times 1.2 = 124.56\, mm$$

Debemos calcular el valor máximo de las variables "SupDcha" y "InfIzda", para calcular el otro lado del rectángulo que la cámara debe visualizar, para ello, los valores máximos de las variables antes comentadas son, respectivamente, 51.9 mm y 150.8 mm, por lo que nos quedamos con el valor mayor, en este caso con InfDcha, es decir con, 150.8 mm y aplicamos el mismo procedimiento que antes:

$$150.8 \times 2 \times 1.2 = 361.92\, mm$$

Estas medidas para esta cámara son menores que para la anterior, por lo tanto **se ha de elegir** las dimensiones de la **cámara superior izquierda**

3.3. Cálculo de variables para la cámara inferior derecha

La cámara inferior derecha deberá visualizar la zona que se puede observar en la figura 26.

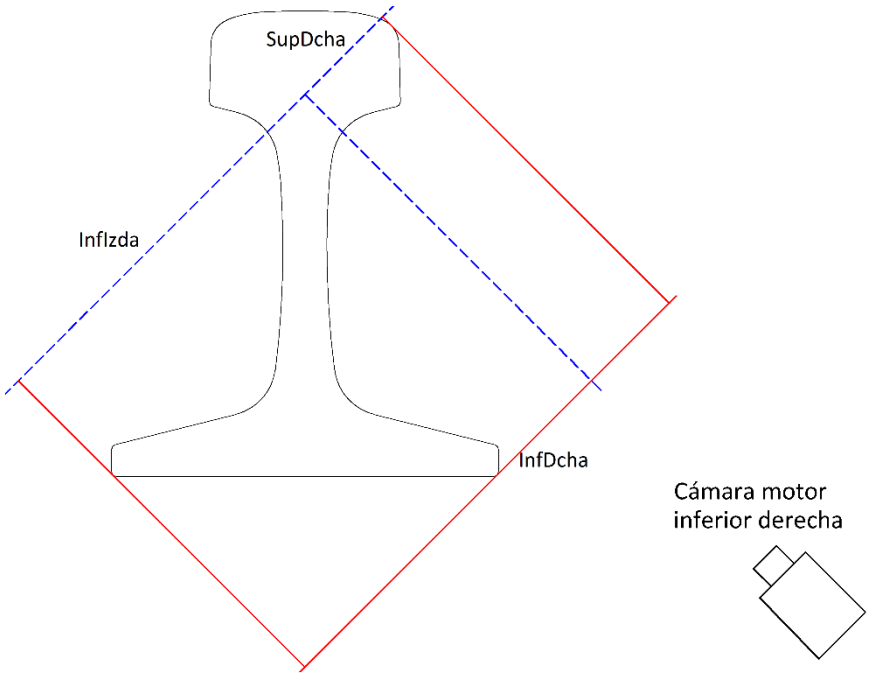


Figura 27: Zona de visualización de la cámara inferior derecha

El máximo valor para la variable "InfDcha", la cual se puede observar en la tabla 1, es de 157 mm, por lo tanto, la cámara deberá visualizar el doble del valor de esa variable más un porcentaje de seguridad, que será de un 20%, por lo tanto:

$$157 \times 2 \times 1.2 = 376.8\, mm$$

Debemos calcular el valor máximo de las variables "SupDcha" y "InfIzda", para calcular el otro lado del rectángulo que la cámara debe visualizar, para ello, los valores máximos de las variables antes comentadas son, respectivamente, 51.9 mm y 150.8 mm, por lo que nos quedamos con el valor mayor, en este caso con InfDcha, es decir con, 150.8 mm y aplicamos el mismo procedimiento que antes:

$$150.8 \times 2 \times 1.2 = 361.92\, mm$$

El área de visión de la cámara superior derecha será de **376.8 mm x 361.92 mm**

3.4. Cálculo de variables para la cámara inferior izquierda

La cámara inferior izquierda deberá visualizar la zona que se puede observar en la figura 27.

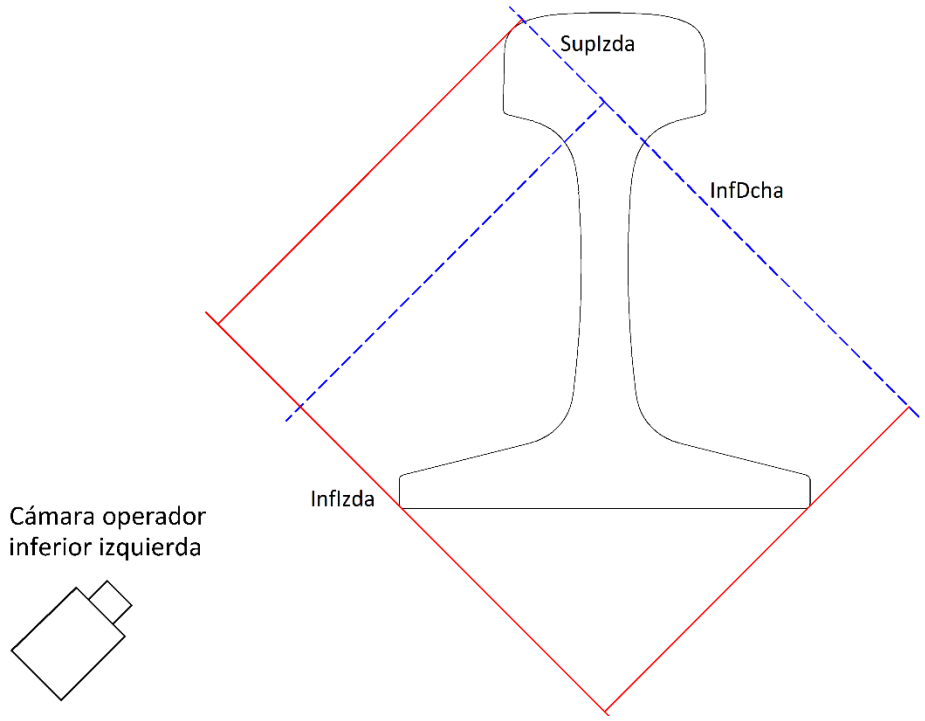


Figura 28: Zona de visualización de la cámara inferior izquierda

El máximo valor para la variable "InfIzda", la cual se puede observar en la tabla 1, es de 150.8 mm, por lo tanto, la cámara deberá visualizar el doble del valor de esa variable más un porcentaje de seguridad, que será de un 20%, por lo tanto:

$$150.8 \times 2 \times 1.2 = 361.92\, mm$$

Debemos calcular el valor máximo de las variables "SupIzda" y "InfDcha", para calcular el otro lado del rectángulo que la cámara debe visualizar, para ello, los valores máximos de las variables antes comentadas son, respectivamente, 51.9 mm y 157 mm, por lo que nos quedamos con el valor mayor, en este caso con InfDcha, es decir con, 157 mm y aplicamos el mismo procedimiento que antes:

$$157 \times 2 \times 1.2 = 376.8\, mm$$

El área de visión de la cámara superior derecha será de **376.8 mm x 361.92 mm**

Para mantener el mismo diseño para las **cámaras inferiores** se elegirá una que capte un cuadrado de **376.8 mm x 376.8 mm**.

Todas las **cámaras en horizontal** tienen que visualizar una distancia de 376.8 mm es decir, unos **40 cms** aproximadamente.

4. Estudio de perfiles de las normas

Ahora se va a realizar una lista con todos los perfiles que contienen las normas que se tienen actualmente para la realización de este proyecto, estas normas son:

- EN-13674
- AREMA Rails
- Algunas especificaciones de raíles de las normas:
    - DIN 536
    - AS 1085
    - GOST R 51685

Además de estas normas se comparan perfiles extraídos de la página web de "ArcelorMittal".

Se van a comparar todos los raíles buscando entre ellos el que tenga mayor valor de altura o de anchura, esto se puede ver reflejado en la tabla 2.

Norma	Nombre	Alto	Ancho
DIN 536	A 150	150	220
DIN 537	A 100	95	200
DIN 538	A 120	105	220
UNE 25122	RN 45	142	130
EN-13674	39 E1	133,35	117,47
EN-13674	45 E1	142,88	127
EN-13674	46 E1	145	125
EN-13674	46 E2	145	134
EN-13674	46 E3	142	120
EN-13674	46 E4	145	135
EN-13674	49 E1	149	125
EN-13674	49 E2	148	125
EN-13674	49 E5	149	125
EN-13674	50 E6	153	140
EN-13674	54 E1	159	140
EN-13674	54 E4	154	125
EN-13674	54 E2	161	125
EN-13674	54 E3	154	125
EN-13674	60 E1	172	150
EN-13674	60 E2	172	150
EN-13674	54 E1 A1	129	147
EN-13674	60 E1 A1	134	140
EN-13674	60 E1 A4	142	150
EN-13674	60 E1 T2	172	150
EN-13674	50 E1	153	134
EN-13674	50 E2	151	140
EN-13674	50 E3	155	133

EN-13674	50 E4	152	125
EN-13674	50 E5	148	135
EN-13674	52 E1	150	150
EN-13674	54 E2	161	125
EN-13674	54 E3	154	125
EN-13674	54 E5	159	140
EN-13674	55 E1	155	134
EN-13674	56 E1	158,75	140
AREMA	100 RE	152,4	136,5
AREMA	115 RE	168,3	139,7
AREMA	119 RE	173	139,7
AREMA	132 RE	181	152,4
AREMA	133 RE	179,4	152,4
AREMA	136 RE	185,7	152,4
AREMA	140 RE	185,8	152,4
AREMA	141 RE	188,9	152,4
AREMA	100ARA-B	143,3	130,6
AS 1085	D1	185,7	152,4
AS 1085	D2	170	146
GOST	P51685	180	150
GOST	P51685	152	132
BS 11	80A	133,35	117,47
BS 11	90 A	142,8	127
BS 11	100 A	152,4	133,3
CHINA	CHINA 50	152	132
CHINA	CHINA 60	176	150
ASCE	85	131,8	130,2
IRS-T-12:1998	IRS 52	156	136
	Máximo:	188,9	220

Tabla 2: Comparación de altura y anchura de railes

En la tabla 2, se han resaltado los perfiles de cada norma que tienen los valores de alto y ancho más grandes de las mismas, y resaltado en rojo se encuentra el valor de alto y ancho más grande de todos los perfiles de todas las normas analizadas.

Estos perfiles que tienen medidas en rojo, es decir, una de sus medidas de ancho o largo es la mayor de toda la tabla, serán los perfiles más desfavorables que se pueden llegar a medir en el sistema, estos perfiles son los siguientes:

- DIN 536 → A150 → Mayor anchura (220 mm).
- DIN 536 → A120 → Mayor anchura (220 mm).
- AREMA Rails → 141 RE → Mayor altura (188.9 mm).

Al ser estos perfiles anteriores los mayores en anchura o en altura, serían también los perfiles más desfavorables que se pueden llegar a medir en el sistema, por lo tanto deben ser tratados de forma independiente.

5. Bibliografía

[AENOR, Marzo 2012] "UNE-EN 13674-1", norma de estandarización europea donde se describe como se han de llevar a cabo las aplicaciones ferroviarias

[AREMA, 2011] "Rail", norma de estandarización americana donde se describe la ingeniería y mantenimiento de las aplicaciones ferroviarias.

[ArcelorMittal] "Types of train rail", página web donde se encuentra una serie de tabla con todos los raíles, sus medidas y sus Normas.

Accesible:

http://www.arcelormittal.com/rails+specialsections/en/types-of-train-rails.html

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ANEXO B

MANUAL DE EJECUCIÓN DE LAS COMUNICACIONES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Organización de la solución

El módulo de comunicaciones de red del sistema de inspección de carril consta de seis proyectos englobados en una solución denominada Communication.

De estos seis proyectos, cuatro implementan la comunicación a nivel de red:

- **PLCServer**: Simulador de PLC. Es un programa servidor.
- **PAServer:** Simulador de ordenador de proceso (Process Automation). Es un programa servidor.
- **MVServer:** Simulador del sistema de visión por computador (Machine Vision) del medidor. Es un programa servidor que envía mediciones y recibe información de los simuladores de PLC y PA.
- **RailCommunication:** Implementa el núcleo de la comunicación, definiendo los tipos de mensajes y el servicio WCF utilizado para comunicarse con el medidor. Es un programa que actúa como cliente de los tres simuladores anteriores. Se conecta a los simuladores de PLC y PA utilizando TCP/IP.

Los dos proyectos restantes, SpinPlatform_Common y SpinPlatform_IO, son proyectos del framework Spin desarrollado por el CDT.

Los simuladores de PLC y PA cuentan con interfaz de usuario gráfica, permitiendo simular las acciones del PLC y del PA en producción.

El simulador MVServer y el programa RailCommunication son aplicaciones de consola.

Estos programas registrarán todos los eventos (conexión, desconexión, recepción o envió de mensajes) bien sea en la interfaz gráfica o en la consola.

2. Ejecución de la solución

Para poder probar la comunicación de forma exitosa se deben iniciar **en primer lugar** los simuladores de PLC, PA y MV. Una vez iniciados se debe ejecutar el cliente, es decir, RailCommunication.

Con los simuladores en ejecución se podrá cerrar el simulador del PLC o el simulador del PA, o ambos a la vez, **pero no** MVServer o RailCommunication. La razón es que la conexión TCP/IP existente entre el PA y el MV o el PLC y el MV se intenta restaurar cada 60 segundos, tal y como dicta el protocolo, pero esto no se realiza en la conexión WCF, ya que no está diseñado para ello.

3. Posibles mensajes de advertencia

Se pueden producir los siguientes mensajes de error:

- **PLC o PA desconectados**: En caso de que alguno de los dos simuladores se haya desconectado, se mostrará un mensaje de error en la consola de RailCommunication y se actualizará el estado en la consola de MVServer.
- **No se puede conectar con MV:** Este mensaje se mostrará si se ha ejecutado RailCommunication antes de iniciar el simulador de MV. En este caso se deberá reiniciar RailCommunication, iniciando previamente el simulador MV, tal y como se explica en el epígrafe anterior.
- **MV desconectado:** Se generará en caso de que el MV se haya desconectado. En este caso se deberá reiniciar RailCommunication, iniciando previamente el simulador MV, tal y como se explica en el epígrafe anterior.

4. Configuración

Es necesario establecer una configuración básica de puertos e IPs para poder ejecutar los cuatro programas descritos. El proceso de configuración se realiza mediante ficheros XML:

- **PLCServer**: Se puede editar el número de puerto en el fichero "**PLCconfig.xml"**.
- **PAServer**: Se puede editar el número de puerto en el fichero "**PAConfig.xml**".
- **MVServer**: No es necesaria configuración.
- **RailCommunication**: Se debe modificar el fichero "RailCommunication.xml", el cual contiene en su región XML "Network":
    - La dirección IP de la máquina donde se ejecuta el simulador PA, así como su puerto.
    - La dirección IP de la máquina donde se ejecuta el simulador PLC, así como su puerto.

No se permite la ejecución del simulador MV y del programa RailCommunication en diferentes máquinas ya que si esto fuese así, sería necesario el uso de un certificado de confianza requerido por WCF.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ANEXO C

COMUNICACIÓN CON EL PLC

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Protocolo de Comunicación MVS-PLC v4
(MVS = *Machine Vision System*)
(PLC = *Programmable Logic Controller*)

Cada mensaje contiene solamente un valor numérico de 8 bits

Mensaje 1: Enter
Dirección: PLC --> MVS
Valor: 1
Descripción: El PLC le indica al MVS que entra un carril en la zona de inspección.

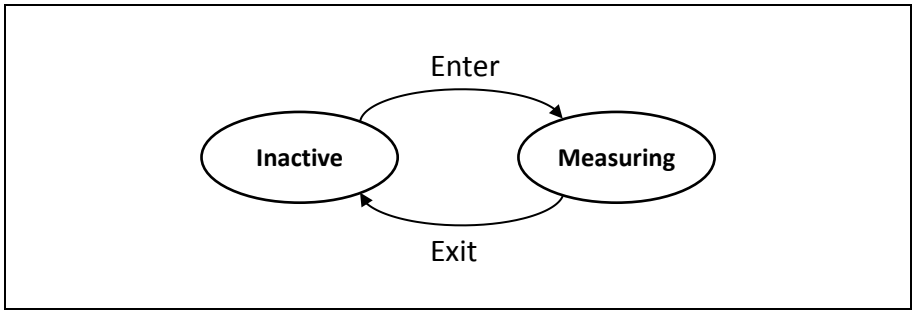
Mensaje 2: Exit
Dirección: PLC --> MVS
Valor: 2
Descripción: El PLC le indica al MVS que sale el carril de la zona de inspección.

Mensaje 3: Deploy_CalibrationTemplate
Dirección: MVS --> PLC
Valor: 3
Descripción: El MVS le indica al PLC que coloque la plantilla de calibración en el plano láser.

Mensaje 4: Retract_CalibrationTemplate
Dirección: MVS --> PLC
Valor: 4
Descripción: El MVS le indica al PLC que quite la plantilla de calibración en el plano láser.

RELACIÓN ENTRE LOS MENSAJES Y EL ESTADO DEL MVS

El MVS tiene dos estados de funcionamiento. Cada mensaje que recibe el MVS del PLC obliga al MVS a realizar una transición de un estado a otro. Aunque para programar el protocolo en el PLC no es estrictamente necesario conocer los estados del MVS, conocer el uso que se da a los mensajes ayuda a comprender el funcionamiento global de todo el sistema.



La secuencia de mensajes debe comenzar con un mensaje Enter y terminar con un mensaje Exit. Entre los mensajes Enter y Exit, cuando las cámaras reciben una señal eléctrica que dispara la captura de una nueva imagen, el MVS procesará las imágenes. Tras recibir el mensaje Exit el MVS debe ignorar cualquier imagen capturada, hasta recibir un nuevo mensaje Enter. Para un determinado carril solo habrá un mensaje Enter y un solo mensaje Exit.

MUY IMPORTANTE: Este protocolo tan simple presupone que el seguimiento de los movimientos del carril lo realiza el PLC completamente. Además, el PLC filtra las señales que genera el encoder para disparar la captura de imágenes, de modo que solo permite que las cámaras reciban una señal de captura cuando realmente deben capturar una imagen. Por tanto, cuando el carril retrocede, el PLC no enviará la señal de captura a las cámaras. Además mientras el carril avanza para recuperar el retroceso, el PLC tampoco enviará la señal.

ESTABLECIMIENTO Y MANTENIMIENTO DE LA CONEXIÓN

La comunicación entre el PLC y el MVS se realizará estableciendo una conexión TCP/IP entre ellos, que se mantendrá "abierta" permanentemente.

El PLC es el Servidor. El MVS es el Cliente.

El PLC siempre debe estar escuchando o esperando por conexiones.

El MVS al arrancar se conectará al PLC, le envía un identificador (byte 10101010) y espera que el PLC se lo devuelva. De esta forma se comprueba el correcto funcionamiento de los equipos.

Si se rompe la conexión, el PLC continuará escuchando y aceptando conexiones, y el MVS se intentará reconectar de forma periódica cada 60 segundos.

FALLOS DEL PROTOCOLO AL NIVEL DE APLICACIÓN

De momento no se ha definido como detectar fallos de protocolo a nivel de aplicación y qué acciones realizar en ese caso.

Por ejemplo, tal como se ha definido, el primer mensaje recibido no puede ser "Exit", o si el MVS está en estado "Measuring", no puede recibir un mensaje "Enter".

Teóricamente, este tipo de fallos no deberían producirse.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ANEXO D

COMUNICACIÓN CON EL PA

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Process Automation Communication Interface

Document:	**Specification**
Version:	2.0
Date:	13/07/2014
Created by:	ArcelorMittal – Process Automation Department ATC-UniOvi
Contact to:	Vanessa Román Leira Vanesa.Roman@arcelormittal.com Daniel F. García dfgarcia@uniovi.es
Status:	<**preliminary**/definite>

Version	**Comments**
1.0	General specifications
2.0	

1 Overview

This document describes the necessities of TCP/IP communication between Automation Process Computer (PA) and Inspection Machine Vision System (MV).

2 Communications

Ethernet TCP/IP connection over Local Area Network (LAN).

In order to keep format of general communications protocol in our host, we propose MV has the client function and PA has the server function.

All data to exchange will be defined as telegrams.

Host	Function	IP Address	Port
Process Automation	Server	To be defined	Defined by PA
Machine Vision	Client	To be defined	Not necessary

In following items of this document we describe data needed to exchange information.

3 Check of connection status and message transfer

3.1 Check message validity

In order to detect the correct start and end of the data stream, the receiver will have to check on the message header and end flags.

In case one of this flags cannot be found considering the message ID, message length, etc, the receiver must restart communication with the sender.

Restart communication will be done by disconnecting and reconnecting the socket.

3.2 Series value

The sender increments the series value by 1 for each message sent (excluding life message).

The series value will be reset to 0 after reaching 9999.

This value could be used as help in commissioning to detect missing messages.

4 Messages

4.1 Data check

The data receiver will not check the validity of the data received. The sender will be responsible of the correct format of the data sent.

4.2 Data format

Message telegrams will be transmitted as ASCII strings.

Numbers

- “0” left fill

Example: Five character value 321 → “00321”, -321 → “-0321”

- Decimal separator “.”
- Missing or unknown values filled with “0”

Strings

- space code right fill

Example: Five character string PLC → “PLC “

- the strings are not terminated by “\0”
- Missing or unknown values filled with spaces

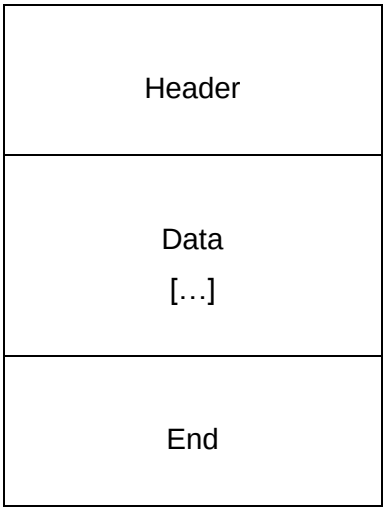
5 Message structure

In order to keep general format of communications in our host, we propose the following message structure:

5.1 General structure

Each message contains:

- a header of constant length
- data that can vary depending on message ID
    - for the same message ID the length of data will be constant
    - unused values will be filled in with reset values (numbers “0”, strings space code)
- an end of constant length



5.2 Message fields description

- Nº

  Reference to the field for documentation purposes
- Field Name

  Identifies the field
- Field Description

  Field explanation
- Field Format

  All fields are ASCII. Numbers will be extracted from the characters as needed.

  String value (A), Numeric Value (N)

  Format is followed by size of the data in number
- Unit

  Mm, kg, ºC, etc
- Comments

  Any comments or description of possible values of the field.

5.3 Message header

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Start_flag	Message start identifier	A6		“Start0”, fixed value
2	Message_length	Indicates the whole length of the message	N8	Byte	Length of Data
3	Date	Date of sending message	A8	YYYYMMDD	
4	Time	Time of sending message	A6	HHMMSS	
5	Series_Value	Unique incrementing value for all messages	N4		0-9999
6	Message_ID	Message identification	A2		
7	Reserved	Reserved for future	A60		

The fixed length of the header is 90 bytes.

5.4 Message end

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS

1	End_flag	Indicates the end of the message	A2		"\0"

6 List of messages

MESSAGE ID	DESCRIPTION	SENDER	RECEIVER
10	Life message	PA	MV
20	Confirmation Life message	MV	PA
11	New rail data	PA	MV
21	Confirmation of new rail data received	MV	PA
22	Last rail measurement results	MV	PA
12	Confirmation of reception of last rail measurement results	PA	MV
13	Request one rail measurement results	PA	MV
23	One rail measurement results	MV	PA
24	MV status	MV	PA

7 Data exchange protocol

Life message

Starting from the last communications reset, PA sends a life message to MV every 60 seconds. If MV does not receive a life message after 120 seconds, MV has to restart communications with PA.

After receiving life message from PA, the MV sends a confirmation of life message back to PA. In case PA does not receive a life message from VM after 120 seconds, it has to restart communications with MV.

Data sending from PA to MV.

- All data describing the rail will be sent before the rail enters into the MV (after last results are received), time enough in advance to the MV to be prepared.
- Data to send are defined in following sections of this documentation.
- MV will send to the PA a confirmation of data received.

Inspection process.

- New rail enters to the Inspection System. PA sends all data from the rail to the MV.
- Inspection results are sent from MV to PA.

Repeat result data

- In case of communication lost between PA and MV, MV may be inspecting anyway. In order to let PA to load inspection result data afterwards, a message of request one rail measurement is sent from PA to MV.

8 Life message

Direction	PA → MV
Message ID	10
Sending	Every 60 Seconds.

9 Confirm Life message

Direction	MV → PA
Message ID	20
Sending	Sent after life message has been received.

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Status	Status ID of the MV: See Table 1	N3		
2	Description	Status ID of the MV: See Table 1	A40		

10 New rail data

For each new rail entering to the MV, the next data are sent from PA to MV:

Direction	PA → MV
Message ID	11
Sending	Before new rail enters the gauge and after results of predeceasing rail received by PA

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Serial_Number	Identification of rail	A10		Example: N680873561
2	Rail_Type	Identification of rail type	A20		Example: CAR60E
3	Tolerance_Std	Identification of tolerance	A20		Example:

		standard			CVD7A73
4	Reserved	Reserved for future	A 60		

10.1 About tolerance standard

@@ Hay que definir un tamaño

IMPORTANTE: ACTUALMENTE EL CONFIGURADOR NO ADMITE ESPACIOS. HAY QUE USAR GUIÓN BAJO. TAMPOCO ADMITE QUE LOS NOMBRES EMPIECEN POR NÚMERO.

Por ejemplo la norma europea denominada "UNE-EN 13674-1" de 14 caracteres con espacio en medio. Otra son, AREMA, GOST, etc.

ArcelorMittal tiene que definir como serán y de donde los obtiene el Ordenador de Proceso

11 Confirmation of new rail data received

After new rail data sent from PA to MV, a confirmation of data received is needed.

Direction	MV → PA
Message ID	21
Sending	After MM receives new rail order data from PA

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Rail_Serial_Number	Identification of rail	A10		Copy of new rail data Example: N680873561
3	Reserved	Reserved for future	A50		

12 Last rail measurement results

After inspection, results data must be collected by PA.

Direction	MV → PA
Message ID	22
Sending	After MV has measured and evaluated

	the whole rail

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Rail_Serial_Number	Identification of rail	A10		
2	Rail_Data	Measured rail data	A??		Variable length

13 Confirmation of reception of last rail measurement results

After last rail measurement results are received by PA, a confirmation of reception is sent to MV.

Direction	PA → MV
Message ID	12
Sending	After PA has received the measurement results

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Rail_Serial_Number	Identification of rail	A10		Copy of new rail data Example: N680873561
3	Reserved	Reserved for future	A50		

14 Request one rail measurement results

The PA can request the measurement results of any rail already inspected.

Direction	PA → MV
Message ID	13
Sending	At any time after a rail has been inspected.

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Serial_Number	Identification of rail	A10		Example: N680873561
2	Reserved	Reserved for future	A 60		

After MV has received this order message, results are sent to PA.

15 One rail measurement results

After MV receives a request of one rail data, the measurement results are sent to PA.

Direction	MV → PA
Message ID	23
Sending	After MV has measured and evaluated the whole rail

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Rail_Serial_Number	Identification of rail	A10		
2	Rail_Data	Measured rail data	A??		Variable length

16 System status

Direction	MV → PA
Message ID	29
Sending	After the status of the system changed

If the system status changes, a message will be send to the PA.

After new connection established between PA and MV status should be sent.

N	NAME	DESCRIPTION	FORMAT	UNIT	COMMENTS
1	Status code	Status ID of the VM: See Table 1	N3		
2	Status name	See Table 1	A40		

Status code	Status Name	Description
0	SYS_ERR_UNKNOWN	Unknown system error.
1	SYS_OK	It is all OK.
2	SYS_IDLE	The system is idle.
3	SYS_BUSY	The system is busy.
4	SYS_ERROR	The system is in an error state.
101	CFG_ERR_UNKNOWN_PRODUCT_TYPE	An unknown product type was received. Warning that should be manager by operator.
103	CFG_ERR_PRODUCT_AT_DEVICE	New rail data could not be read because a rail is in the gauge
200	RES_OK	Inspected and result available Just information, no necessary to wait for this status to send new bar order data.
201	RES_ERR_NOT_INSPECTED	Rail not inspected.
300	NIS_OK	Ready for next pass.
301	NIS_ERR_PRODUCT_AT_DEVICE	There is still a rail in the device. If new bar order data is sent while inspecting, this status code will be received.
302	NIS_ERR_DUPLICATE_ID	The serial number has been already assigned. If two bar order data with the same serial id is sent, this status code will be received.
304	NIS_ERR_DB	Unexpected behaviour of the database server.
305	NIS_ERR_INSERT_PRODUCT	The new rail could not be added to the database.
306	NIS_ERR_UPDATE_SYSSTATE	The system state could not be updated.

Table 1: Status codes, names and description

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ANEXO E

FORMATO DEL FICHERO DE RESULTADOS

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Introducción

Es preciso definir el formato que se sigue para el almacenamiento de las mediciones realizadas y su comunicación.

Por razones de retrocompatibilidad[1], se parte del formato de fichero ya utilizado en el PMG de Joanneum Research, si bien es preciso plantear algunos cambios en el mismo para adaptarlo a las características del sistema que se desarrolla.

Este formato tiene tres usos distintos dentro del sistema:

1. En los ficheros de resultados que genere el programa medidor, que el programa visualizador deberá poder leer.
2. Para la comunicación directa entre el programa medidor y el programa visualizador (modo *En línea*).
3. Para el envío de resultados del programa medidor al ordenador de proceso.

Estos usos pueden, a su vez, agruparse en dos casos diferenciados:

- Los ficheros generados una vez terminada la medición, en los usos 1 y 3.
- Las comunicaciones llevadas a cabo durante la medición, en el uso 2.

Se emplea el mismo formato para ambos casos: los ficheros generados y las comunicaciones durante la medición, con algunos cambios. En este documento se describe en primer lugar el formato de los ficheros, y más adelante se especifican las particularidades cuando se trata de comunicar los resultados parciales durante la medición.

[1] Para permitir que el visualizador pueda acceder a los ficheros de resultados ya existentes y para que otras aplicaciones independientes del sistema puedan acceder a los ficheros generados por el medidor.

2. Estructura general

Los ficheros se estructuran en secciones, encabezadas por un título. El título está rodeado por corchetes. Nada más puede aparecer en la línea de título.

A la línea de título sigue una serie de líneas de datos, formadas por un identificador, un símbolo de igualdad (“=”) y una serie de uno o más valores, separados por espacios, que ocupan el resto de la línea. Estos valores pueden ser números enteros o texto.

Las líneas en blanco y los espacios iniciales y finales se ignoran. El número de espacios de separación entre elementos es irrelevante. Por espacios se entienden los espacios simples (“ “) y las tabulaciones. Los títulos, los identificadores y los valores para los que no se especifique otra cosa pueden estar formados por caracteres alfanuméricos, guiones y guiones bajos. A continuación se muestra un ejemplo de una sección:

```
[NOMBRE_SECCION]
valor           = 0
otro_valor_0    = lorem 1
otro_valor_1    = ipsum 2
```

3. Secciones del fichero de resultados

Las secciones deben aparecer en el mismo orden en el que figuran en este documento. Salvo cuando se especifique lo contrario, las líneas de datos pueden aparecer en cualquier orden dentro de su sección, y se ignoran aquellas cuyos identificadores no coincidan con los esperados.

Los ficheros de resultados de medición contienen las secciones que se indican a continuación.

3.1. HEADER

La sección HEADER ha de contener las siguientes líneas de datos:

- SerialNr: Identificador del carril. Una palabra, sin espacios. El programa medidor recibe este número del ordenador de proceso, o bien lo introduce manualmente el operador. El visualizador lo muestra como metadato del carril, y lo emplea para identificarlo.
- RailType: Código del tipo de carril. Una palabra, sin espacios. El programa medidor lo recibe del ordenador de proceso, o lo introduce manualmente el operador. El visualizador lo muestra como metadato del carril, pero no le da ningún otro uso.
- Standard: Nombre de la norma empleada para comprobar las tolerancias. Se emplea del mismo modo que el tipo de carril.
- CutOffset: Posición en el carril en la que se inicia la medición. Valor entero, en milímetros. El programa medidor le da siempre el valor 0. El programa visualizador no lo utiliza.
- CutLength: Longitud del carril. Valor entero, en milímetros. El programa visualizador no lo utiliza, sino que computa la longitud a partir de las líneas de resultados.

3.2. MEAS_DEFINITION

La sección MEAS_DEFINITION comienza con una línea de datos NrOfDimensions, cuyo valor es el número de dimensiones. A NrOfDimensions han de seguirla las líneas de datos correspondientes a cada dimensión medida. El identificador de cada línea debe ser DIM_nnnn, donde nnnn es el número de dimensión, correlativo y empezando en 0.

Para cada dimensión se especifican su abreviatura, el valor esperado y las tolerancias de cada clase dentro del estándar utilizado, inferior y superior, por ese orden.

Si una tolerancia es, en valor absoluto, superior a 50, se considera que no existe un límite.

3.3. GENERAL_RESULT

La sección GENERAL_RESULT existe por motivos de retrocompatibilidad, y está siempre vacía.

3.4. PROBE_RESULT

La sección PROBE_RESULT existe por motivos de retrocompatibilidad, y está siempre vacía.

3.5. LIMIT_RESULTS_0

La sección LIMIT_RESULTS_0 contiene una serie de líneas de datos correspondientes al mínimo y el máximo de cada dimensión medida. Cada línea comienza con un identificador formado por

el nombre de la dimensión seguido de "_MAX", si es el valor máximo, y "_MIN" si es el mínimo.

Cada línea contiene la posición en metros en la que aparece el valor del que se trate, el valor máximo o mínimo en sí y la clase a la que pertenece el mismo, por ese orden.

La clase a la que pertenece el carril en función del valor máximo o mínimo de una determinada dimensión es la clase de tolerancia más restrictiva que contiene al valor (máximo o mínimo).

3.6. OUT_OF_X_TOLERANCE_SECTIONS_0

La sección OUT_OF_X_TOLERANCE_SECTIONS_0 comienza con una línea NrOfSections, cuyo valor (entero) es el número de mediciones realizadas. Esta línea va seguida de una línea por cada sección de carril medida. El identificador de cada línea debe ser SEC_nnnn, donde nnnn es el número de la medición, correlativo y empezando en 0. Cada línea comienza con la posición en la que se realizó la medición, en metros, y sigue con el valor medido de cada dimensión, en el orden en que se definieron en la sección MEAS_DEFINITION.

El programa medidor designa los valores inválidos por "NaN". El visualizador ignora estos valores así como los marcados como "-1.#IO".

4. Particularidades de la comunicación de resultados parciales

Para la comunicación de resultados parciales se emplea el mismo formato que para el almacenamiento de resultados completos.

Cuando se establece la conexión, el servidor (programa medidor) comienza enviando los metadatos del carril que esté midiendo en ese momento, formados por las secciones HEADER, GENERAL_RESULT y PROBE_RESULT. La sección LIMIT_RESULTS_0 se omite completamente. A estas les sigue la cabecera de la sección OUT_OF_X_TOLERANCE_SECTIONS_0.

Seguidamente, el servidor comienza a enviar líneas de resultados (“SEC_nnnn”) tan pronto como estén disponibles. La línea NrOfSections se omite.

Finalizado el carril, el servidor envía un salto de línea seguido de los metadatos del carril siguiente, del modo descrito antes, y luego los resultados correspondientes a ese carril. Si en algún momento no quedan carriles por medir, el servidor cierra la conexión tras el salto de línea final.

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

SISTEMA DE INSPECCIÓN DE CARRILES: CALIBRACIÓN Y TRATAMIENTO DE IMÁGENES

ANEXO F

TEST CON PATRONES DE PRUEBA

ÁLVARO FERNÁNDEZ MILLARA

JULIO 2014

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

TUTOR: JULIO MOLLEDA MERÉ

Contenido

1. Introducción

Se han fabricado unos carriles a partir de un modelo prestablecido, estos carriles son los que se van a utilizar para llevar a cabo las labores de test del medidor en el laboratorio.

Para ello estos carriles fabricados han pasado una serie de pruebas de calidad, en las cuales se les ha realizado una serie de mediciones para comprobar los errores que se han introducido en el proceso de fabricación, y así tener las medidas exactas de los mismos, para posteriormente documentar estas y comprobar el funcionamiento del sistema.

Las medidas que se han tomado, y que por lo tanto son las que se van a comprobar en el medidor son las que se muestran en la Figura 1.

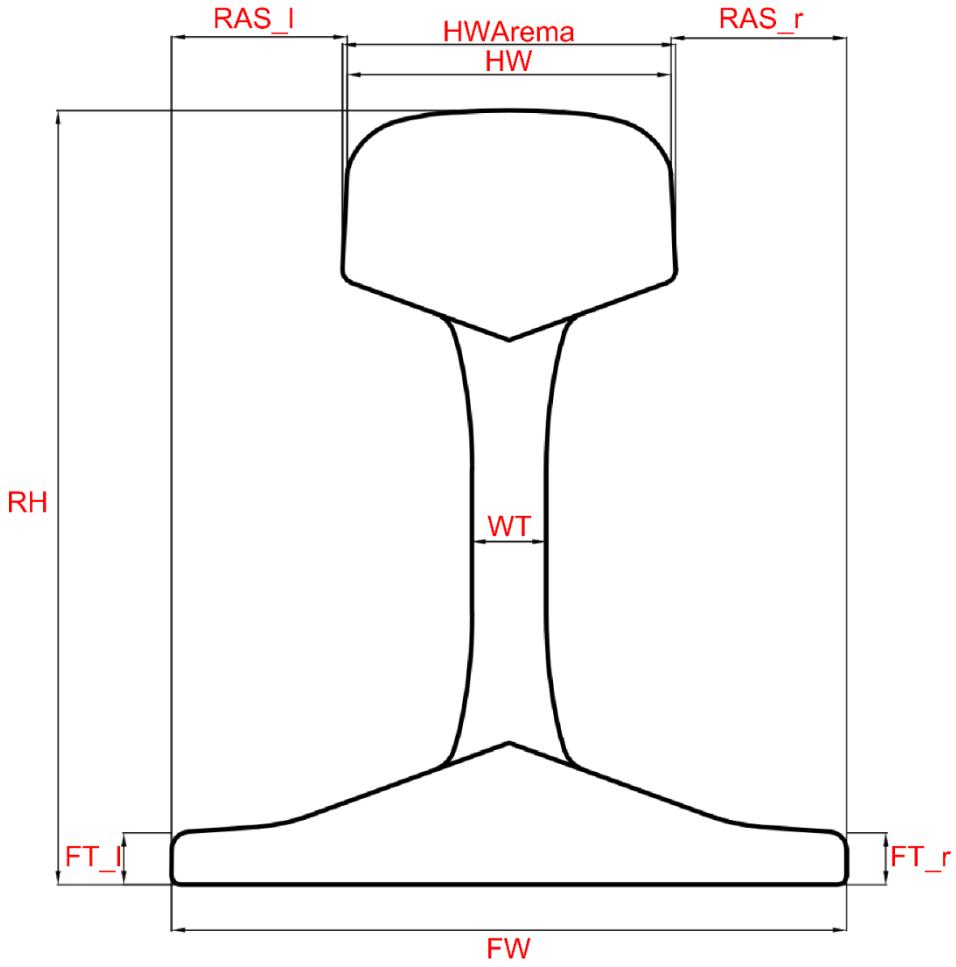


Figura 1: Medidas utilizadas en los test

Estas medidas están incluidas en un par de documentos de referencia creados y certificados por ArcelorMittal, además estas medidas se pueden observar también directamente sobre los patrones que se tienen para realizar las pruebas, ya que se les ha incrustado una serie de líneas en el perfil de los mismos, indicando desde donde hasta a donde se ha medido cada dimensión, la Figura 2.

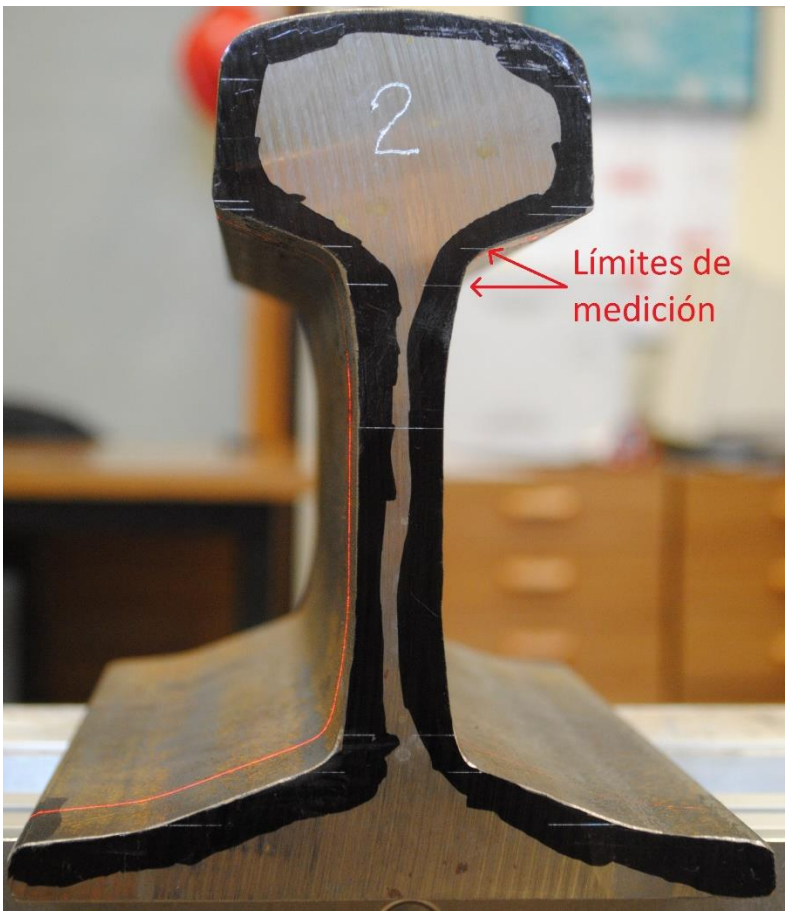


Figura 2: Perfil del patrón de prueba con las marcas para realizar las mediciones

Para realizar las mediciones en el laboratorio y compararlas con los certificados de Arcelor, se ha de medir el carril a una determinada distancia de su longitud, es decir, donde fue medido por los ingenieros de calidad de la empresa, para ello se coloca el carril en la estructura del laboratorio de tal manera, que los láser “cortan” el carril a 3 cm de la parte inicial del carril, que es exactamente donde ha sido medido, tal y como se puede observar en la Figura 3.

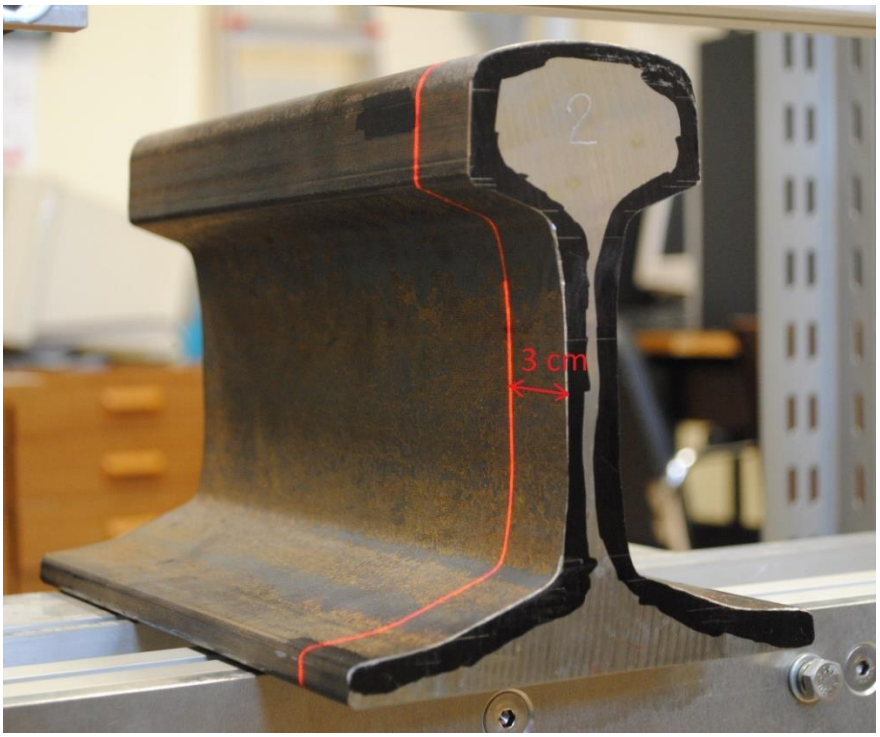


Figura 3: Patrón sobre la estructura de medición del laboratorio

Posteriormente, el carril será desplazado a la derecha e izquierda y arriba y abajo (tal y como se muestra en la Figura 4) sobre la estructura que lo soporta, para comprobar si el resultado obtenido en las mediciones varia, así como medir con y sin luz en el laboratorio, e incluso, dar la vuelta al carril y ponerlo boca abajo para comprobar si esto influye de alguna manera en los resultados de medición.

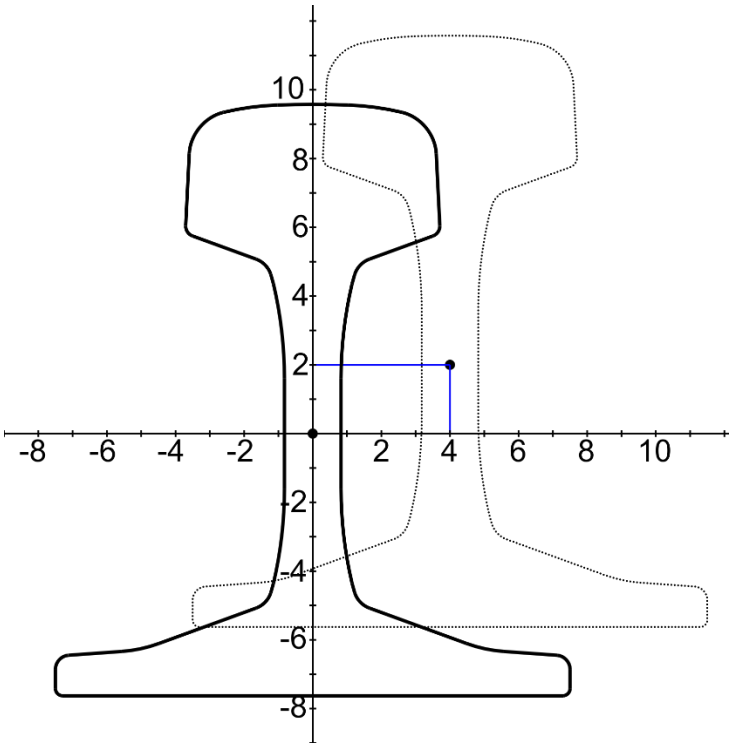


Figura 4: Carril desplazado 4 cm a la derecha y 2 cm hacia arriba

Todo esto se comprobará en los epígrafes siguientes.

2. Test con el Patrón 1

El experimento realizado sobre este patrón se ha comparado con las mediciones realizadas por Arcelor de forma empírica sobre el propio carril.

La Tabla 1 recoge las variaciones, tanto de posición del carril, como ausencia o no de luz e incluso la forma en que se ha colocado el carril para las diferentes mediciones realizadas.

	Desplazamiento			
	Vertical	Horizontal	Con luz	Boca arriba
Test 1	0	0	Si	Si
Test 2	0	0	Si	Si
Test 3	0	0	No	Si
Test 4	0,6 cm	-5 cm	Si	Si
Test 5	1 cm	5 cm	Si	Si
Test 6	≈ 0	≈ 0	Si	No

Tabla 1: Batería de experimentos realizados sobre el patrón 1

Una vez se han obtenido los resultados de todas las mediciones, o tests, realizados al carril, se procede a representar en una tabla, se puede observar esta en la Tabla 2.

	Patrón 1	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	E. Min	E. Med	E. Máx
FT_l	11,55	11,758	11,770	11,773	11,774	11,821	11,431	-	-	-
FT_l err.	-	0,208	0,220	0,223	0,224	0,271	-0,119	-0,119	0,171	0,271
FT_r	11,50	11,257	11,269	11,260	11,306	11,151	11,231	-	-	-
FT_r err.	-	-0,243	-0,231	-0,240	-0,194	-0,349	-0,269	-0,349	-0,254	-0,194
FW	150,17	149,473	149,510	149,507	149,454	149,468	149,613	-	-	-
FW err.	-	-0,697	-0,660	-0,663	-0,716	-0,702	-0,557	-0,716	-0,666	-0,557
HWArema	73,90	73,327	73,341	73,341	73,385	73,733	73,526	-	-	-
HWArema err.	-	-0,573	-0,559	-0,559	-0,515	-0,167	-0,374	-0,573	-0,458	-0,167
HW	72,10	71,325	71,362	71,358	71,398	71,813	71,717	-	-	-
HW err.	-	-0,775	-0,738	-0,742	-0,702	-0,287	-0,383	-0,775	-0,605	-0,287
RAS_l	39,02	39,499	39,514	39,509	39,494	38,899	39,218	-	-	-
RAS_l err.	-	0,479	0,494	0,489	0,474	-0,121	0,198	-0,121	0,335	0,494
RAS_r	38,76	38,512	38,464	38,474	38,602	38,724	38,434	-	-	-
RAS_r err.	-	-0,248	-0,296	-0,286	-0,158	-0,036	-0,326	-0,326	-0,225	-0,036
RH	172,60	172,347	172,357	172,356	172,410	172,451	172,420	-	-	-
RH err.	-	-0,253	-0,243	-0,244	-0,190	-0,149	-0,180	-0,253	-0,210	-0,149
WT	16,88	16,697	16,746	16,745	16,526	16,983	16,817	-	-	-
WT err.	-	-0,183	-0,134	-0,135	-0,354	0,103	-0,063	-0,354	-0,128	0,103

Tabla 2: Resultados de los experimentos realizados sobre el patrón 1

La columna “Patrón 1” contiene los datos medidos por ArcelorMittal sobre el carril; por otra parte las columnas “Test1”, “Test2”, etc., contienen las mediciones realizadas por el sistema en cada uno de los escenarios de test, descritos en la Tabla 1, si la letra es negra, y si la letra es roja, contienen la diferencia entre la medición realizada en el test y la medición real del patrón 1, es decir:

$$Error\ de\ la\ Dimension\ \text{"n"} = Test\ M(n) - Patron1(n)$$

Hay además otras tres columnas las cuales representan los errores mínimos, medios y máximos, respectivamente, de una dimensión.

Estos errores se representan en la gráfica que se encuentra en Figura 5.

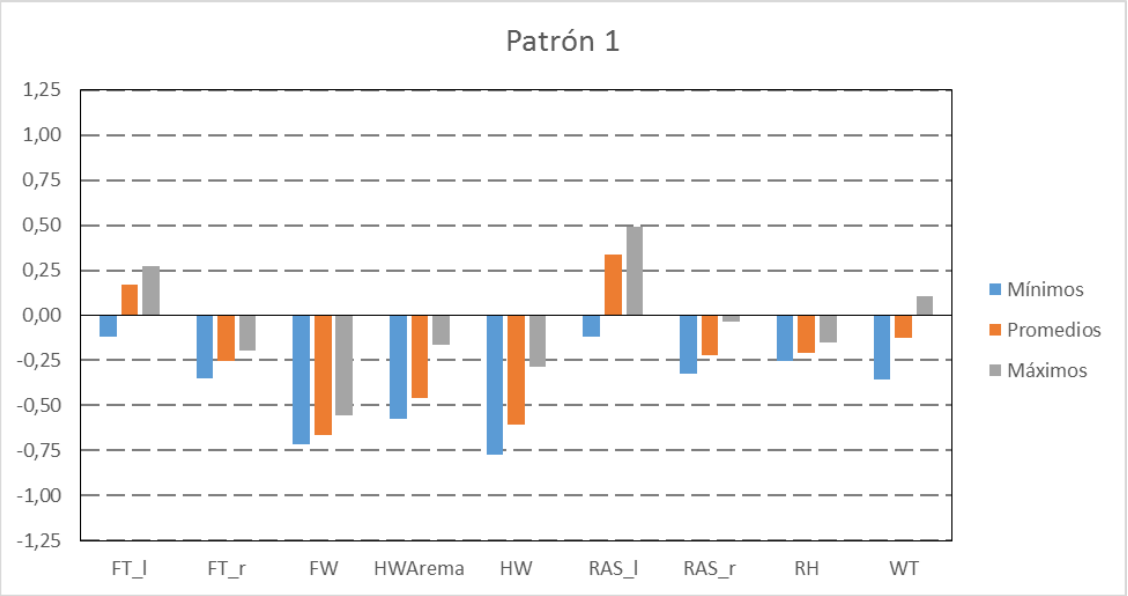


Figura 5: Gráfica de errores del patrón 1

3. Test con el Patrón 2

El experimento realizado sobre este patrón se ha comparado con las mediciones realizadas por Arcelor de forma empírica sobre el propio carril.

La Tabla 3 recoge las variaciones, en este caso tan sólo de posición que se han realizado sobre el patrón 2.

	Desplazamiento	
	Vertical	Horizontal
Test 1	0,6 cm	0
Test 2	0	-5 cm
Test 3	0	5 cm

Tabla 3: Batería de experimentos realizados sobre el patrón 1

Los resultados obtenidos tras realizar los tres experimentos propuestos en la Tabla 3, se recogen, de forma similar a la realizada en el patrón 1, en la Tabla 4.

	Patrón 2	Test 1	Test 2	Test 3	E. Min	E. Med	E. Máx
FT_l	11,52	11,880	11,962	12,007	-	-	-
FT_l err.	-	0,360	0,442	0,487	0,360	0,430	0,487
FT_r	11,50	11,710	11,818	11,446	-	-	-
FT_r err.	-	0,210	0,318	-0,054	-0,054	0,158	0,318
FW	150,40	149,823	149,760	149,678	-	-	-
FW err.	-	-0,577	-0,640	-0,722	-0,722	-0,646	-0,577
HWArema	74,00	73,595	73,378	73,637	-	-	-
HWArema err.	-	-0,405	-0,622	-0,363	-0,622	-0,463	-0,363
HW	72,24	71,684	71,086	71,727	-	-	-
HW err.	-	-0,556	-1,154	-0,513	-1,154	-0,741	-0,513
RAS_l	39,20	39,217	39,864	38,926	-	-	-
RAS_l err.	-	0,017	0,664	-0,274	-0,274	0,136	0,664
RAS_r	39,07	38,928	38,904	38,890	-	-	-
RAS_r err.	-	-0,142	-0,166	-0,180	-0,180	-0,162	-0,142
RH	172,80	172,734	172,720	172,764	-	-	-
RH err.	-	-0,066	-0,080	-0,036	-0,080	-0,061	-0,036
WT	16,77	16,295	15,940	16,281	-	-	-
WT err.	-	-0,475	-0,830	-0,489	-0,830	-0,598	-0,475

Tabla 4: Resultados de los experimentos realizados sobre el patrón 2

La distribución de columnas y el cálculo de los errores, máximo, mínimo y medio, se ha realizado de forma análoga a la llevada a cabo en el patrón 1.

Los resultados se han representado en una gráfica, la cual se puede ver en la Figura 6.

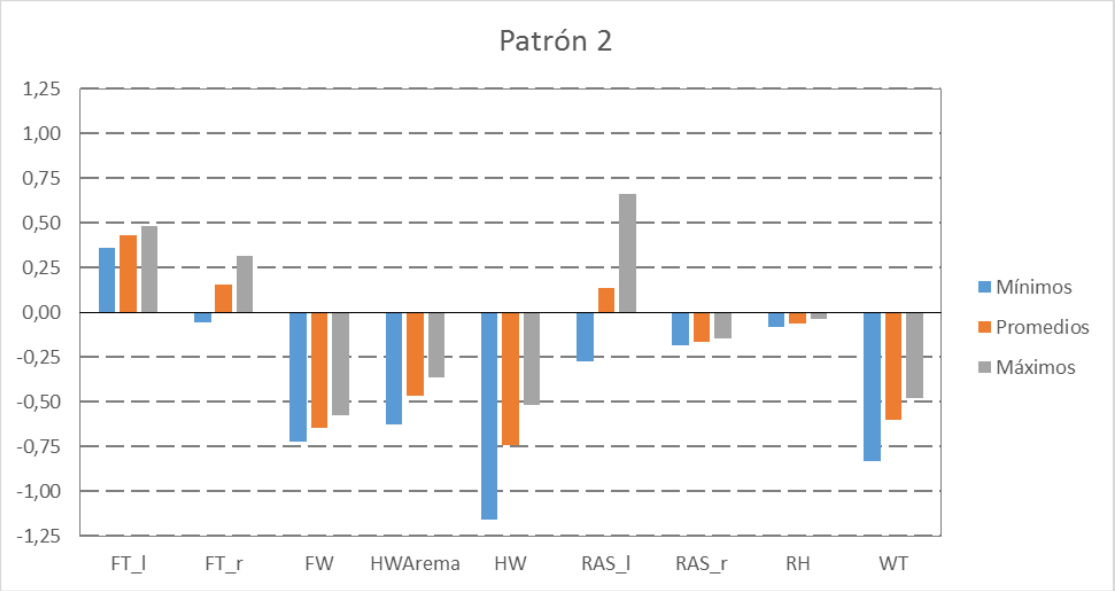


Figura 6: Gráfica de errores del patrón 2

4. Conclusión

El medidor para la mayoría de las dimensiones proporciona unos valores que están por debajo de las medidas reales. Esto se refleja en la Figura 5 y en la Figura 6 observando que el mínimo error y el máximo error son ambos negativos. Esto indica un sesgo determinista en los mecanismos de medición.

La situación ideal sería tener un error mínimo negativo y un error máximo positivo, estando el error medio entre ambos y próximo a cero.

El comportamiento del error es muy similar para ambos patrones, pues para todas las dimensiones excepto una el error medio es positivo o negativo en ambos patrones.

Además la magnitud de los errores sigue la misma tendencia para ambos patrones. Las dimensiones que presentan un error de magnitud elevada en el patrón 1, también suelen presentar un error de magnitud elevada en el patrón 2. Y viceversa, las dimensiones que presentan un error de magnitud reducida en un patrón, también lo presentan en el otro patrón.