Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Doctoral thesis

Dimensional inspection of railway tracks using computer vision

Submitted by

Álvaro Fernández Millara

in partial fulfillment of

the requirements for the degree of

Doctor por la Universidad de Oviedo

(Programa de doctorado en informática)

Directed by

Dr. Julio Molleda Meré

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

RESUMEN DEL CONTENIDO DE TESIS DOCTORAL

1.- Título de la Tesis	
Español/Otro Idioma: Inspección dimensional de carriles de tren utilizando visión por computador	Inglés: Dimensional inspection of railway tracks using computer vision

2.- Autor	
Nombre: Álvaro Fernández Millara	DNI/Pasaporte/NIE:
Programa de Doctorado: Informática	
Órgano responsable: Centro Internacional de Postgrado	

RESUMEN (en español)

Para garantizar la seguridad y la calidad del transporte sobre carriles, hay un gran número de normas nacionales e internacionales que imponen tolerancias muy ajustadas sobre distintas propiedades de los carriles. Esta tesis estudia técnicas de triangulación láser para el control y la inspección de calidad, así como algoritmos de calibración de cámaras y de autoevaluación adecuados para los sistemas de triangulación láser. Para este fin, se propone un caso de estudio: un sistema de inspección de calidad para carriles de tren, llamado galga de medición de perfiles (PMG).

Entre todas las propiedades de los carriles, la PMG se ocupa de comprobar las distancias entre diferentes regiones del perfil del carril. Estas distancias también se denominan dimensiones del carril, e incluyen la altura, la anchura y el espesor en varias zonas. Para detectar defectos dimensionales, en la PMG se usan técnicas de triangulación láser para reconstruir el perfil de los carriles poco después de su fabricación. Se miden distintas propiedades del perfil, y los resultados se comparan con los valores nominales. A su vez, estas diferencias se comparan con las tolerancias permitidas por las normas aplicables, y las dimensiones que estén fuera de tolerancia se marcan como tales.

El objetivo principal de este trabajo es tratar la cuestión de cómo pueden mejorarse y cuantificarse la precisión y la fiabilidad de los sistemas de triangulación láser. En general, las posibles mejoras pueden agruparse en dos áreas: calibración y autoevaluación.

En este trabajo se exploran todos los pasos necesarios para el funcionamiento de un sistema de inspección de calidad basado en triangulación láser: captura de imágenes, extracción de líneas láser, traducción de coordenadas, generación de perfiles, medición y salida. Los principales algoritmos se describen en detalle: la extracción de líneas láser, la generación de nubes de puntos a partir de ellas, la alineación de las nubes con un modelo de carril, la descomposición del modelo en primitivas y el ajuste de la nube de puntos a dichas primitivas.

Luego, se examinan los principios generales de la calibración de cámaras, para determinar cuál es el mejor modo de calibrar un sistema como la PMG. Se consideran tanto los parámetros de calibración intrínsecos como los extrínsecos. Se proponen distintos algoritmos de calibración extrínseca que están diseñados para una plantilla de calibración especialmente diseñada para la PMG y sistemas similares. Estos algoritmos se comparan entre sí, y con un procedimiento de calibración conocido de tipo *sheet-of-light*. Se muestra que la exactitud del mejor algoritmo de los considerados es similar a la del procedimiento *sheet-of-light*. También se propone un modo de que los operadores humanos comprueben la calidad de una determinada calibración.

Por último, se considera la posibilidad de implementar funcionalidades de autoevaluación para la PMG. Se consideran varias características que pueden usarse como indicadores de la salud del sistema, y se examinan técnicas para estimar estas características. Se proponen y evalúan distintas reglas para emitir eventos de alarma. Para ello, estas reglas se aplican a los valores de las características calculados sobre resultados de mediciones anteriores (obtenidas en la fábrica de ArcelorMittal en Gijón).

De este modo, este trabajo ha aportado el diseño y la evaluación de distintos algoritmos de medición y calibración, así como técnicas de autoevaluación. Estas contribuciones pueden usarse para asistir en el mantenimiento, y de este modo proteger la exactitud y la fiabilidad del sistema, logrando el alto grado de robustez que es necesario en un entorno industrial.

RESUMEN (en inglés)

In order to ensure the safety and quality of rail transportation, there are a number of national and international standards that impose tight tolerances on different properties of rails. This thesis studies laser triangulation techniques for quality control and inspection, as well as camera calibration and self-assessment algorithms suitable for a laser triangulation system. To this end, a case of study is proposed: a quality inspection system for railway rails, called the profile measurement gauge (PMG).

Among all the properties of the rails, the PMG is concerned with the distances between different regions of the rail cross-section, also called rail dimensions. These include the rail height, and its width or thickness at various locations. In order to detect dimensional defects, the PMG uses laser triangulation techniques to reconstruct a cross-sectional profile of rails shortly after they are manufactured. Different profile dimensions are then measured, and the measurements are compared with the nominal values. These differences are in turn compared with the allowed tolerances for the applicable standards, and out-of-tolerance dimensions are flagged as such.

The main goal of this work is to address the question of how the accuracy and reliability of laser triangulation systems can be improved and quantified. For the most part, improvements can be made in two main areas: calibration and self-assessment.

In this work, all the required steps for a quality inspection laser triangulation system are explored: image acquisition, laser line extraction, coordinate translation, profile generation, profile measurement and output. The main algorithms involved are described in detail: extracting laser lines and generating point clouds from them, matching point clouds to a rail model, decomposing the model into primitives, and fitting those primitives to point cloud points.

Then, the general principles of camera calibration are examined to determine how best to perform camera calibration for a system such as the PMG. Both intrinsic and extrinsic calibration parameters are considered. In order to do this, different extrinsic calibration algorithms are proposed that are designed for a special calibration target suitable for the PMG and similar systems. These algorithms are then compared with each other, and with a well-known sheet-of-light calibration procedure. It is shown that the accuracy of the best algorithm among those proposed is similar to that of the sheet-of-light procedure. A way for human operators to assess the quality of a given calibration is also proposed.

Finally, the possibility of implementing autonomic computing features for the PMG is evaluated. A number of different features are considered that may be used as indicators of the health of the system, and techniques to implement sensors to measure such features are examined. Different rules for triggering alarm events are proposed and evaluated by applying them to the results of running the sensors on historical rail data (obtained at the ArcelorMittal factory in Gijón).

In this way, this work has contributed the design and evaluation of different measurement and calibration algorithms, as well as self-assessment techniques. These contributions may be used to aid in maintenance and therefore protect system accuracy and reliability, achieving a great degree of robustness which is required in an industry environment.

SR. PRESIDENTE DE LA COMISIÓN ACADÉMICA DEL PROGRAMA DE DOCTORADO EN INFORMÁTICA

Abstract

In order to ensure the safety and quality of rail transportation, there are a number of national and international standards that impose tight tolerances on different properties of rails. This thesis studies laser triangulation techniques for quality control and inspection, as well as camera calibration and self-assessment algorithms suitable for a laser triangulation system. To this end, a case of study is proposed: a quality inspection system for railway rails, called the profile measurement gauge (PMG).

Among all the properties of the rails, the PMG is concerned with the distances between different regions of the rail cross-section, also called rail dimensions. These include the rail height, and its width or thickness at various locations. In order to detect dimensional defects, the PMG uses laser triangulation techniques to reconstruct a cross-sectional profile of rails shortly after they are manufactured. Different profile dimensions are then measured, and the measurements are compared with the nominal values. These differences are in turn compared with the allowed tolerances for the applicable standards, and out-of-tolerance dimensions are flagged as such.

The main goal of this work is to address the question of how the accuracy and reliability of laser triangulation systems can be improved and quantified. For the most part, improvements can be made in two main areas: calibration and self-assessment.

In this work, all the required steps for a quality inspection laser triangulation system are explored: image acquisition, laser line extraction, coordinate translation, profile generation, profile measurement and output. The main algorithms involved are described in detail: extracting laser lines and generating point clouds from them, matching point clouds to a rail model, decomposing the model into primitives, and fitting those primitives to point cloud points.

Then, the general principles of camera calibration are examined to determine how best to perform camera calibration for a system such as the PMG. Both intrinsic and extrinsic calibration parameters are considered. In order to do this, different extrinsic calibration algorithms are proposed that are designed for a special calibration target suitable for the PMG and similar systems. These algorithms are then compared with each other, and with a well-known sheet-of-light calibration procedure. It is shown that the accuracy of the best algorithm among those proposed is similar to that of the sheet-of-light procedure. A way for human operators to assess the quality of a given calibration is also proposed.

Finally, the possibility of implementing autonomic computing features for the PMG is evaluated. A number of different features are considered that may be used

as indicators of the health of the system, and techniques to implement sensors to measure such features are examined. Different rules for triggering alarm events are proposed and evaluated by applying them to the results of running the sensors on historical rail data (obtained at the ArcelorMittal factory in Gijón).

In this way, this work has contributed the design and evaluation of different measurement and calibration algorithms, as well as self-assessment techniques. These contributions may be used to aid in maintenance and therefore protect system accuracy and reliability, achieving a great degree of robustness which is required in an industry environment.

Resumen

Para garantizar la seguridad y la calidad del transporte sobre carriles, hay un gran número de normas nacionales e internacionales que imponen tolerancias muy ajustadas sobre distintas propiedades de los carriles. Esta tesis estudia técnicas de triangulación láser para el control y la inspección de calidad, así como algoritmos de calibración de cámaras y de autoevaluación adecuados para los sistemas de triangulación láser. Para este fin, se propone un caso de estudio: un sistema de inspección de calidad para carriles de tren, llamado galga de medición de perfiles (PMG).

Entre todas las propiedades de los carriles, la PMG se ocupa de comprobar las distancias entre diferentes regiones del perfil del carril. Estas distancias también se denominan dimensiones del carril, e incluyen la altura, la anchura y el espesor en varias zonas. Para detectar defectos dimensionales, en la PMG se usan técnicas de triangulación láser para reconstruir el perfil de los carriles poco después de su fabricación. Se miden distintas propiedades del perfil, y los resultados se comparan con los valores nominales. A su vez, estas diferencias se comparan con las tolerancias permitidas por las normas aplicables, y las dimensiones que estén fuera de tolerancia se marcan como tales.

El objetivo principal de este trabajo es tratar la cuestión de cómo pueden mejorarse y cuantificarse la precisión y la fiabilidad de los sistemas de triangulación láser. En general, las posibles mejoras pueden agruparse en dos áreas: calibración y autoevaluación.

En este trabajo se exploran todos los pasos necesarios para el funcionamiento de un sistema de inspección de calidad basado en triangulación láser: captura de imágenes, extracción de líneas láser, traducción de coordenadas, generación de perfiles, medición y salida. Los principales algoritmos se describen en detalle: la extracción de líneas láser, la generación de nubes de puntos a partir de ellas, la alineación de las nubes con un modelo de carril, la descomposición del modelo en primitivas y el ajuste de la nube de puntos a dichas primitivas.

Luego, se examinan los principios generales de la calibración de cámaras, para determinar cuál es el mejor modo de calibrar un sistema como la PMG. Se consideran tanto los parámetros de calibración intrínsecos como los extrínsecos. Se proponen distintos algoritmos de calibración extrínseca que están diseñados para una plantilla de calibración especialmente diseñada para la PMG y sistemas similares. Estos algoritmos se comparan entre sí, y con un procedimiento de calibración conocido de tipo *sheet-of-light*. Se muestra que la exactitud del mejor algoritmo de los considerados es similar a la del procedimiento *sheet-of-light*. También se propone un modo de que

los operadores humanos comprueben la calidad de una determinada calibración.

Por último, se considera la posibilidad de implementar funcionalidades de autoevaluación para la PMG. Se consideran varias características que pueden usarse como indicadores de la salud del sistema, y se examinan técnicas para estimar estas características. Se proponen y evalúan distintas reglas para emitir eventos de alarma. Para ello, estas reglas se aplican a los valores de las características calculados sobre resultados de mediciones anteriores (obtenidas en la fábrica de ArcelorMittal en Gijón).

De este modo, este trabajo ha aportado el diseño y la evaluación de distintos algoritmos de medición y calibración, así como técnicas de autoevaluación. Estas contribuciones pueden usarse para asistir en el mantenimiento, y de este modo proteger la exactitud y la fiabilidad del sistema, logrando el alto grado de robustez que es necesario en un entorno industrial.

Contents

List of figures

List of tables

List of algorithms

Chapter 1

Introduction

Steel rails enable the movement of trains to transport both cargo and passengers, and also cranes, to manipulate cargo in heavy industry and warehousing. Rails are engineered, designed and produced to support high-speed, heavily loaded modern trains.

In order to ensure the safety and quality of transportation, tight control must be exerted over both the internal structure of the rails, which affects their behavior under fatigue; and their shape, which affects the stability of both the rails and the railroad vehicles they support, as well as the amount of vibration suffered by the vehicles. These requirements have been codified in multiple national and international standards.

Rail inspection is performed in two different scenarios to assess whether the rail successfully fulfills the required standards and the requirements of the clients: the rolling mill, where all rails should be inspected upon manufacture; and the railroad, where the rail track should be periodically checked for wear damage.

In the rolling mill, quality inspection must detect any internal, surface and shape defects, among others, and determine whether they are within applicable tolerances. One of the key inspection processes is dimensional measurement: Dimensional inspection of rail profiles is essential to provide precise feedback to upstream rolling stages and to ensure the rails meet the required standards.

Profile measurement systems inspect the shape of the rail profiles to assess their dimensional quality. Laser triangulation techniques are typically used to this end. This assessment can be used in order to provide feedback for shape control devices in upstream manufacturing, and also to check whether the products are compliant with rail standards and client requirements.

Laser triangulation systems are usually installed in housings to provide a controlled lighting environment and to prevent human operators from being exposed to laser reflections. Human operators may only access these installations when the rolling mill is not in operation. Therefore, a laser triangulation system should be very resilient to faults caused by a hostile industrial environment. In order to do this, the most direct approach is to build a robust system that can remain in operation under degraded conditions. However, robustness can never completely eliminate the need

for maintenance, so a way to measure system health would be useful in order to detect, or even predict, degraded accuracy or failure.

1.1 Scope and goals

This work attempts to determine how to improve and quantify the accuracy of industrial laser triangulation systems. This can be divided into the following goals:

1. Design and evaluate calibration procedures for laser triangulation systems in an industrial environment, and quantify their contribution to the accuracy of the results.

2. Devise strategies to measure calibration error, and to determine whether recalibration is needed at a given point in time.

3. Explore possible techniques to measure general system health, detect and, if possible, predict, accuracy issues and system failure; to correct such issues automatically if possible; and, when needed, to alert system operators to the fact that a maintenance action should be performed.

1.2 Outline

First, in Chapter 2, railway rails are described, along with their structure and their manufacture process. Then, rail standards and the different requirements they impose on rails are discussed, including those related to rail dimensions. Testing and inspection techniques, including mechanical, acoustic and optical techniques are discussed. Finally, different ways are discussed to apply these techniques to the task of quality inspection for railway rails.

Chapter 3 introduces a proposed solution for dimensional inspection of rails. The goals of the system are discussed, and its physical and logical structure is described. Algorithms for the different stages of rail measurement are extensively explained.

In Chapter 4, the calibration of the proposed system is discussed. First, the theoretical background of camera calibration is detailed. Then, specific challenges that arise when calibrating laser triangulation systems are discussed, and different algorithms are shown that deal with these challenges. Multiple strategies to measure calibration error, and to check whether a given set of calibration parameters is reliable, are proposed. Finally, different experiments are described that compare the reliability of the proposed algorithms with that of a well-known calibration technique.

Chapter 5 introduces the concept of autonomic computing and how it might be applied to the proposed system. Different sensors for system health are described,

the results of running them on historical rail measurement data are shown. Multiple ways of using sensor output and conveying the results to system operators are shown. Then, possible rules to detect unacceptable system performance are discussed, and the results of applying them to historical data are shown.

Finally, Chapter 6 contains the conclusions of this work, as well as a description of possible future work.

Chapter 2

Background and state of the art

In order to provide context for this work, this chapter first describes railway rails, their manufacture process and the defects they may present, and then a number of testing techniques that can be applied for industrial purposes. The final section covers the intersection of these two fields, the techniques generally used for quality control in rail manufacture.

2.1 Railway rails and their manufacture

Centuries before the first steam locomotive was designed and built by John Fitch in 1794, wagons pulled by horses were guided and supported by rails. Early rails were first made of wood, and later of iron, which, being more durable, greatly reduced maintenance costs. This rudimentary form of railway, the wagonway, was mostly used for the purpose of transporting ore from mines and stone from quarries [Tzanakakis, 2013].

John Fitch's prototype was quickly followed by commercially successful models, and soon the horses gave way to steam engines. Faster, heavier vehicles required better materials for the rails, and at the same time innovations in steel manufacturing made it cheaper and more readily available, so it was not long until steel replaced iron.

Today, rails are exclusively made of steel. Quality steel must be used, as small faults that would be of no consequence for other usages cannot be tolerated in the case of rails because of the high forces they must endure, and the risk of derailment if they fail to do so and break.

2.1.1 Rail profiles

A rail profile can be divided into three regions, called the head, the web and the foot:

- **Head:** The head is the top part of the rail, and the only one that is intended to be in direct contact with rolling stock. This means that the rail heads are normally subject to much more intense wear than the rest of the rail.

For this reason, a head hardening treatment is sometimes applied during manufacturing.

- **Web:** The web is the narrow part that connects the head and the foot. Identification markings are usually located there.
- **Foot:** The foot is the bottom part of the rail. Its underside must be flat*, and is intended to rest on the railway ties.

Rails may be symmetrical or asymmetrical around the vertical axis. Symmetrical rails are used for most of the track, while asymmetrical rails are mostly used for switches and crossovers.

Additionally, special grooved rails are used for tramway tracks. These rails have the head to one side and a guard on the other side, with a groove between them to accommodate the train wheels. This way, the tracks can be embedded in the street pavement.

Figure 2.1 shows the profile, or cross-section, of three different rail models.

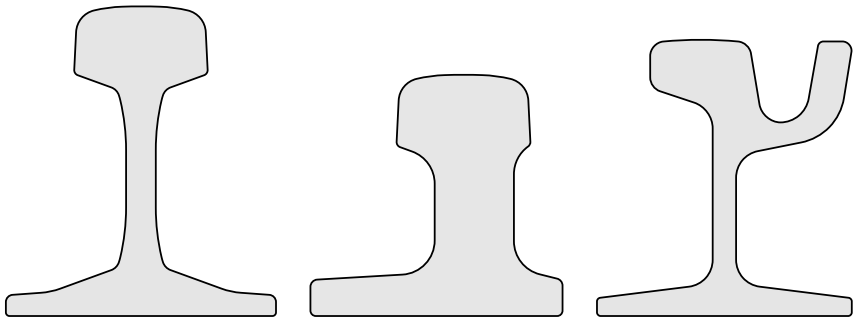

Figure 2.1: Profiles of three different rail models. From left to right, these are the symmetrical 60 E1, the asymmetrical 60 E1 A1 and the grooved 35 GP.

2.1.2 Steel and its manufacture

Steel is an alloy of iron and carbon, and sometimes other elements**, that presents better mechanical properties than iron, mainly an increased tensile strength.

A high proportion of carbon in a steel makes it more resistant to stress, but also less malleable. For rails, beams and other products for structural applications, where

* A small amount of concavity may be acceptable, depending on the applicable standards, but convexity is never tolerated, as it would make the rail unstable.

** For instance, stainless steel is a steel with chromium added into the mix. When this steel is exposed to air, a thin layer of chromium oxide forms on the surface, instead of rust. This inert layer prevents corrosion, rather than speeding it up as rust would.

resistance is more important than malleability, high-carbon steel* is used, while medium-carbon or low-carbon steel are generally used for all other purposes.

Evidence suggests that steel was in use in Central Anatolia during the third millennium BCE [Akanuma, 2008]. However, mass production did not start until the middle of the 19th century, when Henry Bessemer patented his steelmaking process [Bessemer, 1856]. This considerably reduced the costs of steel production and made it feasible as a building material.

Modern steel manufacture is generally carried out as follows [Centro de formación ArcelorMittal Asturias, 2007]:

1. Iron ores, such as hematite, limonite and magnetite, are extracted and broken into small pieces, which are called fines. After separating and discarding the gangue, the fines are combined with coke (distilled coal), forming a mass called the sinter.

2. Sinter pieces are placed in a blast furnace, along with additional coke and flux. In the blast furnace, the iron is separated from its impurities and combined with carbon. The result is pig iron, which is iron with a very high carbon content, making it extremely brittle.

3. Pure oxygen is applied to the still molten pig iron at a high speed. This forms carbon monoxide and dioxide and lowers the proportion of carbon until it is low enough for the iron to be considered steel.

4. The molten steel is continuously dropped into a bottomless mold**. After the steel exits the mold, it is sprayed with water or air until it is completely solid. It is then cut into bars.

The result is an intermediate product, which is then shaped into the finished products. In the case of rails, this is described in Section 2.1.3. Different types of intermediate products exist, depending on the shape of the mold. For instance, slabs are intermediate products with a rectangular cross-section that is much larger in width than in height, while blooms are products with a square or near-square cross-section.

2.1.3 Rail manufacture

Rails, like beams and similar long products with a constant cross-section, are manufactured by shaping steel blooms into the desired final product. This can be done in the following way [Centro de formación ArcelorMittal Asturias, 2007]:

* Steel with a carbon content between approximately 0.6 % and 1 % by weight.

** This is called continuous casting [Tarquinee and Scovill, 1955], as opposed to conventional casting using individual ingots.

1. Blooms are reheated to approximately 1250 °C using burners. Before reheating, the blooms can be at approximately 600 °C, if used immediately after casting, or at ambient temperature, if recovered from storage. Heating the steel over its recrystallization temperature makes reshaping it easier, but at the same time it decreases the accuracy of the final shape, because the material shrinks as it cools.

2. Mill scale is removed from the blooms. Mill scale is a flaky layer of iron oxide that forms on the surface of the blooms, which must be removed before the blooms can be reshaped. Otherwise, the scale could get embedded in the rails, causing surface defects. In the case of rails and beams, mill scale is usually removed by having the blooms pass through a ring of pressurized water jets, which chip the scale away. The recovered scale can then be recycled into sinter.

3. The bloom is passed through the rolling mill in order to shape it into a rail. The rolling mill is comprised of a series of rolling cages, and each cage contains two rollers that rotate in opposite senses. Each cage contains rollers whose shapes are closer to that of a rail than those of the cage before it. As the bloom passes between the rollers, it is compressed and deformed, going from its original cross-section to a smaller near-square cross-section, then to rough approximations and then to its final, intended shape. This means that small defects in the roughing and intermediate cages will not affect the final product, while any issues with the shape of the rollers in the last cage (called the finishing cage), or with their rotation, will translate to defects in the shape of the rails. Contact with the rollers may cause more mill scale to appear, which must be removed before the product can be shaped further. This is done after the bar exits each of the cages.

4. Small regions at the beginning and end of the bar, which have an irregular shape after rolling, are cut away and discarded. These discarded pieces, along with other pieces of scrap metal, are usually molten again in an electric arc furnace.

5. Identifying marks are applied to the hot bars. As bars are usually cut into multiple rails afterwards, such marks include both a bloom identifier, which is the same for the whole bloom, and a rail identifier, which serves the purpose of differentiating the final rail from other bars from the same bloom. As such, the latter identifier varies for different lengths along the bar.

6. The bars are curved. As the bars cool, the difference in mass between the head and the foot causes curves to form. Curving the rails before cooling, in the sense opposite to that of the expected curving, will reduce the need to straighten the final product.

7. Optionally, special treatments are applied to the rail. For instance, head-hardened (HH) rails can be produced by rapidly cooling the rail head using water jets. This causes a fine pearlite* microstructure to form [Sahay et al., 2009], making the rail head strong and hard.
8. The bars are cooled to ambient temperature. Controlled water jets are usually employed to this end.
9. The bars are straightened, both horizontally and vertically.
10. The bars are inspected for various kinds of defects, as detailed in Section 2.3.
11. Bars that have been deemed valid are cut into smaller, commercial length rails**.
12. The finished rails are put in storage until they can be sent to the customer.

2.1.4 Rail standards

Standards on rail applications regulate the design and manufacturing of railway rails. A typical rail standard describes:

- Steel grades that can be used.
- Chemical composition of the steel.
- Material properties such as fatigue crack growth rate and fracture toughness.
- Size and placement of brand and stamp marks on the rails.
- Required content of such marks.
- Dimension tolerances.
- Surface flatness.
- How the rails must be tested for the specified requirements, when appropriate.
- How the rails that are found to be non-conforming are to be treated.

Different standards apply depending on the destination country for the rail, and also on the quality requirements made by the purchaser. Some of these standards are:

* **Pearlite** is a layered structure of ferrite (α-Fe, iron with a body-centered cubic structure) and cementite (Fe_3C, iron carbide).

** After rolling and cooling, the blooms have a length of approximately 75 m, too long to be easily transported. They are usually split in half.

- EN-13674, 'Railway applications - Track - Rail', by the CEN (Comité Européen de Normalisation, European Committee for Standardization) [European Committee for Standardization, 2011]. The national standardization bodies of all the European Union members, as well as certain other European countries, are bound to give CEN standards the status of national standards.

- The AREMA (American Railway Engineering and Maintenance of way Association) Manual for Railway Engineering [AREMA, 2011], which includes recommended practices for different aspects of railway design, construction and maintenance. These recommendations are usually followed in the United States and Canada.

- The UIC (Union Internationale des Chemins de fer, International Union of Railways) Technical specification for the supply of rails [International Union of Railways, 2008].

- GOST R 51685, 'Railway rails' [Gosstandart of Russia, 2001], a state standard of the Russian Federation.

The tests that are described in these standards can be split into two groups, depending on the frequency that they are to be carried out, as follows:

- **Acceptance tests:** Acceptance tests include (destructive) laboratory tests on chemical composition and material properties such as tensile strength and hardness, and also non-destructive tests on rail dimensions, flatness and surface and internal defects.

  It may be necessary to carry out different acceptance tests at different frequencies. For instance, they may be required once for every heat*, or every 100 to 1000 tonnes. On the other hand, non-destructive tests may be required to be automated and performed on all the rails, as stated in Section 2.2.

- **Qualifying tests:** Qualifying tests include the full set of acceptance tests as described above, plus more elaborate laboratory tests for material properties, such as fatigue tests [Cannon et al., 2003; Ahlström and Karlsson, 2004].

  Qualifying tests must be performed periodically, usually after a given number of years, and every time the production process changes in a significant way.

* A **heat** is a batch of blooms, or more specifically the 'liquid steel melt tapped out of a converter or electric arc furnace which includes after continuous casting a given number of blooms relating to the weight of the heat and the extension of the mixing zone [...]' (from EN-13674 [European Committee for Standardization, 2011], although other standards also employ this term).

2.1.5 Rail dimensions

Among the possible faults that rails can manifest, defects in their cross-section are probably the easiest to understand.

In order to quantify these defects, a set of **dimensions** is defined. Every dimension is a numerical value that can be considered as a function of the rail profile, usually the distance between two specific points.

Some examples of dimensions considered by the standards mentioned in Section 2.1.4 are the height of the rail, the width of the head and the thickness of the foot.

We can define the **rail model profile** as the ideal cross-section of a given rail model, and the **rail section profile** as the measured cross-section of a rail at a given point along its length. Then, by evaluating the rail dimensions on the rail model profile, we obtain a set of ideal or desired values of the dimensions, and by evaluating them on the rail section profile, we obtain a set of measured values. By subtracting the ideal values from the measured values, we can compute the dimensional error.

Finally, by comparing the error with the tolerances specified by the applicable standard, we find whether the section is within the allowed range for every dimension. This must be repeated for sections all along the rail.

2.2 Testing and inspection techniques

Testing is defined, in a wide sense, as the procedures that can be employed to measure the properties of an object and of the materials of which it is made.

Quality can be defined as the 'degree to which a set of inherent characteristics of an object fulfils requirements' [ISO, 2015]. Therefore, the role of product testing in quality control is to check whether the properties of the measured object fulfill a given set of requirements. According to the specific case, this information can be used in order to either

- identify the underlying issues in the manufacturing process, and be able to correct them,
- fix the issues directly in the product itself, or compensate for them in some way,
- classify products in an appropriate quality level,
- or discard the faulty product.

Testing techniques can be divided into two broad categories:

- **Destructive testing techniques** are those testing techniques that destroy, damage or modify the object being tested or parts of it. This can either be because the tests themselves are carried out until the object's destruction, or because the object must be altered as a preparation before the techniques can be applied.

- **Non-destructive testing techniques** are those testing techniques that do not cause damage to the object being tested.

Because of their nature, destructive testing cannot be applied in all the situations where non-destructive techniques can be used. For quality control, destructive testing is only feasible when dealing with mass-produced objects, and it can only be applied to a small sample of the objects, which must be selected and separated from the rest, and usually discarded after the destructive tests are performed.

On the other hand, non-destructive testing is not limited in that way, so it can be applied more widely. Another advantage of non-destructive techniques is that all the products may be tested, rather than a small subset.

However, some information cannot be readily obtained from non-destructive testing, which means that both destructive and non-destructive testing have their place in quality control, and a compromise must be reached between them.

Now we will describe some of the most common non-destructive techniques.

2.2.1 Mechanical techniques

Many tools exist that can measure qualities of an object by mechanical means. For instance:

- **Weighing scales:** Weighing scales are the best known such tools. Scales determine the mass of an object, usually by measuring its weight: the force that the object exerts on the scale due to gravity.

  Many weighing methods exist. For instance:

  - Simple balances merely compare the weight of one object with the weight of a set of known masses.
  - Analytical ('lab') balances measure the force an electromagnet needs to exert in order to compensate for the weight of the object [Emery, 2002].
  - Spring balances measure the distance a spring extends when attached to an object. This distance is proportional to the force exerted on the spring, according to Hooke's law [Chapman and Kent, 2005, Chapter 8].
  - Hydraulic scales measure the pressure variation inside a hydraulic circuit caused by the force exerted by the object on a piston [Douglas and Keen, 1997].

  - Strain gauge balances (including most body weight scales) use a strain gauge, which is an electrical conductor whose resistance varies depending on how much it is deformed [Moncrief et al., 1992].

- **Go/no-go gauges:** Go/no-go gauges are simple tools that measure whether pieces of a specific shape are within their allowed tolerances [O'Leary, 1983].

  A pair of gauges, or two different parts of the same gauge, must be applied to the pieces (by attempting to fit them inside the pieces or the pieces inside them, depending on the specific tolerance that is being checked). In order for a piece to be considered valid, it must pass the 'go' test and fail the 'no go' test.

- **Calipers:** Calipers measure the distance between two sides of an object.

  A compass with outward facing points can be used to measure the internal size of an object (an inside caliper), while a compass with inward facing points can measure the external size (an outside caliper). Using these basic calipers is as simple as fitting the caliper inside an object or an object inside a caliper, respectively, and then taking the caliper and measuring the distance between its tips.

  More advanced calipers are comprised of a long piece with a calibrated scale and a fixed jaw at one end, and a second jaw that can slide along the scale. The distance between the jaws can be read directly from the scale.

  In digital calipers, the calibrated scale is replaced or complemented with a linear encoder to quantify the movement along the body of the caliper, as well as a display to show the distance between the jaws, computed from said movement. Both are usually installed on the sliding jaw.

- **Coordinate-measuring machines:** Coordinate-measuring machines have a touch probe (or an optical probe, for contactless scanning) that is attached to an articulated arm or to a 'bridge' structure that can move along the three orthogonal axes X, Y and Z [Bernhardt et al., 2000]. Objects are placed under the probe, which then moves automatically or under a human operator's control in order to scan the objects. The points (x, y, z) where the probe is located when it touches the object are recorded. The full surface of an object can be reconstructed in this way.

2.2.2 Acoustic techniques

Sound waves are used for non-destructive testing in two different ways:

- **Acoustic emission:** Acoustic emission refers to the sound waves that a material generates due to a change in its structure, usually caused by an external stimulus [Wadley and Mehrabian, 1984].

Rather than using an active sensor that emits waves in order to capture them when they are reflected by or pass through the object undergoing testing, acoustic emission testing entails capturing these sound waves, generated by the object, in order to detect and locate faults inside it.

This technique can be applied to quality control: faults in the product can be found as soon as they appear during the manufacturing process, by detecting the sound waves that are emitted when the material cracks or gets deformed or damaged in any other way.

It is also used for damage detection in structural health monitoring (SHM) [Tan et al., 2009]. As material ages, and a structure is exposed to stress, faults may appear that can weaken it. This technique can be used as described above in order to continuously acquire updated data on structural damage.

- **Ultrasonic testing:** In ultrasonic testing, very short sound waves are sent through an object, and then captured when they come out. Faults inside the object, or differences in the composition of its parts, cause part of the sound waves to bounce, thus allowing a receiver to detect their existence and location [Firestone, 1942].

  The receiver can be placed inside the same device as the emitter, in order to capture the reflected waves, or alternatively in a separate device to be placed on the other side of the object, in order to capture the waves that reach the other side (in this case, an attenuated wave would evidence the presence of an anomaly).

  Contact between the testing device and the object under inspection is required for many ultrasonic testing techniques. A liquid or gel couplant is often applied to the object in order to prevent air from entering the gaps between the testing device and the object, as air is a poor conductor of ultrasonic waves [Molina, 1974].

2.2.3 Techniques based on electric currents

Multiple testing methods exist that employ electric currents or magnetic fields to find faults inside conductive objects. The most common one is eddy-current testing (ECT) [García-Martín et al., 2011].

In eddy-current testing, an alternating current is passed through a coil, which produces a magnetic field around it. This field in turn induces currents* inside the (conductive) object under inspection. Defects on and near the surface of the object can then be detected as changes in the impedance of the coil.

* Called eddy currents, after the somewhat analogous behavior observed in fluid dynamics; or alternatively Foucault currents, after their discoverer.

However, defects that are deep inside the object cannot be detected with ECT, as the current density of alternating current decreases exponentially as the distance to the surface increases. This is called the 'skin effect'.

2.2.4 Optical techniques

Many inspection techniques exist that rely on light. These techniques can be split into image-based and non-image-based techniques, depending on whether images* are generated or not. The first category includes the usage of area-scan and line-scan cameras, whereas the second category includes the usage of other devices such as photodiodes and optical rangefinders.

Most image-based techniques employ machine vision. Machine vision is a vast family of techniques for testing, process control and other industry applications, based on image processing.

The term 'machine vision' is not to be confused with the related term 'computer vision'. Computer vision is the scientific discipline that attempts to design algorithms that model the functionality of the human visual system, or a subset thereof, in order to automate tasks related to image processing. Machine vision has been succinctly defined as 'the application of computer vision to factory automation' [Deshpande, 2008, Chapter 1].

Machine vision systems usually comprise the software and hardware for processing, as well as one or more cameras and a light source. Visible light is commonly used, but machine vision techniques can of course be applied to other frequency ranges in the electromagnetic spectrum**, and to other kinds of sensors as long as they can produce images as their output. However, for the purposes of this brief introduction, only machine vision techniques based on visible light will be considered.

Image processing for machine vision usually starts with the application of different kinds of filters and thresholding in order to remove noise and unwanted objects and highlight all the relevant image elements.

Then, the objects of interest must be located in the image. This stage is called object extraction. Object extraction can be purely algorithmic, using edge detection and simple geometric criteria, or it may use machine learning techniques for pattern recognition, depending on the nature of the objects to be extracted.

Unless the goal of the system is merely the detection, location or counting of the objects, further processing must be done on the extracted objects. Again, this may entail the usage of relatively simple algorithms to measure object dimensions or other properties (for instance, when information must be extracted from barcodes), or elaborate machine learning techniques for object classification (as an example,

* For this purpose, an **image** can be defined as the output of a one-dimensional or two-dimensional array of sensors representing the changes that occurred to them at a given instant in time.

** Especially the long-infrared range, for thermal imaging. An external light source is obviously not needed in this case.

when information must be extracted from handwritten text on the objects, using optical character recognition (OCR)).

The most relevant optical techniques for the purpose of this work are those that measure distances or reconstruct 3D objects. Multiple such techniques exist [Sansoni et al., 2009], both based on machine vision and on rangefinders. Some examples of the most prominent of these techniques follow.

2.2.4.1 Time of flight

Time of flight (TOF) techniques measure the time that it takes for a light beam (usually a laser pulse) to reach the object under inspection, bounce back and arrive at the emitter [Hussmann et al., 2008]. The distance to the object can then be trivially computed.

A matrix of rangefinders working as described, or a single rangefinder if multiple measurements are performed, can reconstruct a surface in this way. This is often called lidar, a portmanteau of light and radar.

The obvious shortcoming of this technique is that inspecting small objects requires circuitry that can measure extremely small time differences. As the speed of light (assuming a vacuum, for simplicity) is 299 792 458 m/s, light covers a distance of 0.299 792 458 nm in a nanosecond, which means that a granularity of nanoseconds would not be enough to perceive differences in distance smaller than that.

2.2.4.2 Laser triangulation

In laser triangulation, a laser emitter is used in order to project either a single point or a stripe on the object under inspection.

When a single point is projected on the object, if the location and bearing of the laser emitter relative to the camera are known, then a line may be defined that passes through the laser emitter and the object under inspection. The laser point is the point where the line enters the object under inspection. We may call this line the measurement line. If we can map the laser point, as seen in the image, to a line from the camera to the laser point, then we can easily find the location of the laser point, relative to the camera, by computing the intersection of the measurement line and this second line. Figure 2.2 shows a simple example of laser triangulation.

By moving (translating or rotating) either the laser emitter or the object under inspection, more points of the surface of the object can be found over time. In any case, such movements must be known precisely.

When a line is projected on the object, a measurement plane (also known as a 'sheet of light') can be defined, rather than a line. Then, the laser line as seen by the camera can be split into a number of points. Optical triangulation can then be applied as before in order to find the location of these points in the measurement plane.

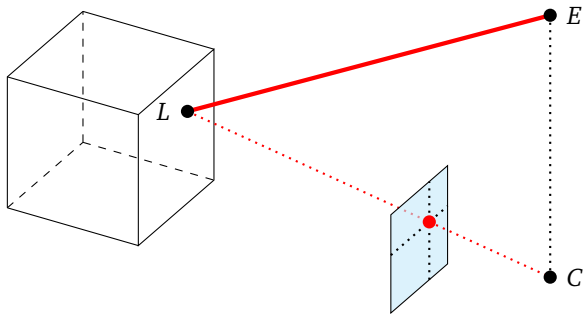

Figure 2.2: Example of laser triangulation with a single point. Here, E, L and C represent the laser emitter, the laser point and the camera, respectively. The cube and the pale blue square respectively represent the object under inspection and the image plane.

Many variations of this basic arrangement exist. For instance, multiple laser-camera pairs can be used. This requires the location of each camera relative to the others to be known, in order to combine their output.

If the laser lines fall inside the same measurement plane, and the emitters are placed in such a way that laser lines are projected on the object all along its intersection with the measurement plane, then the profile of the object can be computed with a single set of images, one from each camera. By moving the object through the plane and computing its profile at different points, a complete 3D reconstruction can be created.

Commercial products exist that combine a camera and a laser emitter into a single device, so that determining the location of the emitter relative to the camera is not an issue.

2.2.4.3 Spectral image differencing

In order to detect small defects, such as cracks, on generally flat surfaces, two light sources of different colors (usually red and blue) can be used. The light sources must be placed over opposite ends of the inspection area, in such a way that the light from each of them falls on the surface of the object at the same oblique angle. A color camera must be placed over the object, at a right angle.

The color images captured by the camera are then split into two grayscale images (if red and blue lights were chosen, this can be done by simply taking the red and blue channels of the images). Afterwards, differential images can be created for each pair of images, so that the regions of the images where the light from both sources falls with equal intensity appear black, and regions where the light of one of the

sources can be seen with more intensity appear in shades of gray.

In regions with cracks, bumps or other sudden differences in height, the light from the two sources will travel different distances, and in some cases one of them will be occluded, and so the defects will be seen in the differential images [Deutschl et al., 2004]. Figure 2.3 shows an example of spectral image differencing.

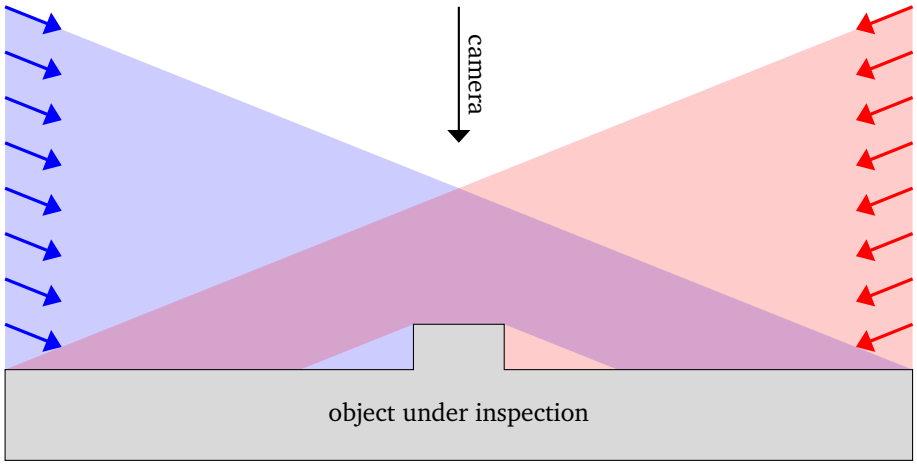

Figure 2.3: Example of a spectral image differencing setup.

2.2.4.4 Structured light

Techniques based on structured light employ the same principles as the simple laser triangulation techniques described in Section 2.2.4.2, except that one or more known patterns are used rather than a single point or a line. In this way, surfaces may be measured quickly and using a single image.

Grids or sets of stripes are usually employed as patterns. When the objects that will be measured are expected to be complex, care must be taken to correctly identify the stripes, grid sections or any other kind of pattern elements on the objects, as their order may not be the same as the order in which they were projected. Determining which pattern element as projected on the object corresponds to which pattern element in the model is called *indexing*. One of many strategies that have been proposed is to employ a different color for every pattern element [Boyer and Kak, 1987].

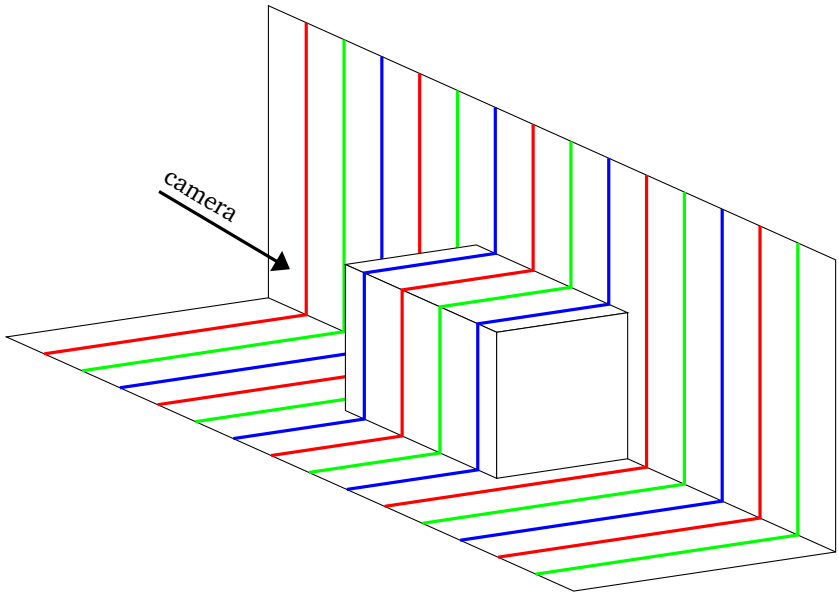

Figure 2.4: Example of a structured light setup. Each of the red, green and blue lines represents a laser stripe.

2.2.4.5 Stereo vision

Computer stereo vision relies on two or more cameras to capture a scene from different angles [Hussmann et al., 2008]. Shapes must be extracted from each image, and they must be matched with the corresponding shapes from the other images. If the position of the cameras relative to each other is known, simple optical triangulation can then be applied in order to find the actual position of the shapes. No external light sources are needed.

The quality of the results depends on the ability to find corresponding features in the images. This is called the correspondence problem, and it is a well-known problem in computer vision*.

As such, the accuracy of stereo vision largely depends on the specific algorithm for feature correspondence, and the kind of shapes the objects under inspection may present. Therefore, other methods, such as those that were described in the preceding sections, should be preferred if high accuracy is important. However, this technique should be considered if accuracy is not a pressing concern and a relatively simple setup is needed.

* It is also well-known in the field of neuroscience [Nieder, 2003], as biological perception of depth, also known as stereopsis, relies on similar principles as computer stereo vision.

2.3 Inspection techniques applied to railway rails

As stated before, rails are inspected for compliance with the relevant standards before they can be shipped to the customer. While many different kinds of tests must be performed, for the most part we will concern ourselves with the subset of non-destructive acceptance tests that are performed on all the rails shortly after cooling. These tests [Papaelias et al., 2008] are generally as follows:

- **Internal defects:** Ultrasound, as described in Section 2.2.2, is used in order to detect discontinuities under the rail surface. Both contact and contactless ultrasound systems exist for rail inspection. Water is generally used as a couplant for both. In the former, the rail is first sprayed with water, and ultrasound devices, comprising both an emitter and a receiver, are in contact with the rail as it passes through. In the latter, the devices emit ultrasonic waves and jets of water at the same time: the sound waves travel from the devices to the rail through the water, go through the rail, bounce and then travel back to the devices through the same jet of water.

- **Surface defects:** The technique described in Section 2.2.4.3 is used in order to build a differential image. Then, a system based on machine learning is used to find possible defects in the differential image, and then classify them (determining what kind of defects they are, e.g.: hot marks, cold marks or embedded mill scale). Detection and classification can be considered as separate stages, and different subsystems can be used. For instance, one neural network to locate the defects and a second one to classify them.

  Eddy currents, as described in Section 2.2.3, are also used in order to detect small cracks and other surface discontinuities. Transverse and longitudinal discontinuities cannot be detected at the same time using this technique, so at least two devices must be used.

- **Head flatness:** Laser triangulation is performed as described in Section 2.2.4.2. Multiple cameras are used in order to reconstruct the head in detail, all along the length of the rail. After the vertical movement of the rail is accounted for, the minute changes in the shape of the head can be interpreted as flatness defects. Flatness is measured both for at the top of the head and at its sides; that is to say, the parts of the rail that will be in direct contact with train wheels [Manso et al., 2017].

- **Dimensions:** Laser triangulation is performed in order to reconstruct the complete rail profile. Then, the dimensions and their error can be computed as described in Section 2.1.5. Although different systems exist for this task, apart from the one described in Chapter 3, the same principles apply to all of them.

Before the arrival of these optical systems, go/no-go gauges were used sin order to check the dimensional tolerances. They are still used occasionally for double-checking or complementing their results.

Some standards describe certain dimensions by providing gauges: apart from being used directly as inspection tools, these gauges can also serve as unambiguous definitions for the dimensions they check.

Apart from quality inspection inside the factory, rails must also be periodically checked once installed, as the rail heads are subject to heavy wear and degrade quickly. Laser triangulation [Li et al., 2007; Liu et al., 2011; Zheng et al., 2012; Wang et al., 2017] systems can be installed on rail maintenance vehicles to this end. Mechanical gauges, and specialized portable coordinate-measuring machines, like Greenwood Engineering's MiniProf [Greenwood Engineering A/S, 2010], are also an alternative.

Chapter 3

Proposed solution

The solution proposed in this work employs laser triangulation in order to compute the cross-sectional dimensions of railway rails immediately after they are manufactured. Results are then compared with nominal values, in order to determine whether the measured dimensions are within applicable tolerances.

This chapter describes the overall solution, while the following chapters will explore specific system components, the challenges they pose, the techniques used to overcome them and proposed improvements over the state of the art.

3.1 Goals

The system, called the profile measurement gauge (PMG), must be able to measure a set of dimensions along the length of a rail, as described in Section 2.1.5. For this purpose, rails are considered to contain a discrete amount of **rail sections**, with a predefined distance between them, as seen in Figure 3.1.

The distance between sections is set to 200 mm. With this spacing between sections, at a rolling speed of 1 m/s, the system will have to process 5 sections each second.

After measuring the dimensions for a given rail section, the result must be reported to a human operator. Both the measured values of the dimensions and the difference between those values and the ideal model values must be shown.

After a complete rail is measured, the results must be recorded for later reference.

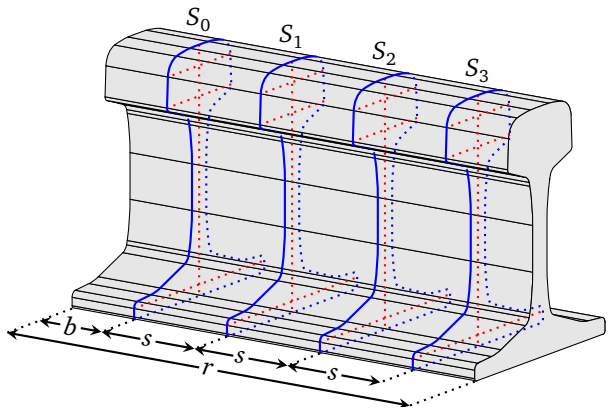

Figure 3.1: Rail sections shown on a rail, in blue. In red, some dimensions. r is the rail length, b is the distance between the beginning and the first section and s is the distance between sections. The depicted distances are not to scale.

3.2 Physical structure

A theoretical model for the system is depicted in Figure 3.2, and can be described as follows:

- Four laser emitters (designated E_1 to E_4, clockwise from top left to bottom left) are placed on the corners of a square. The laser emitters point towards the center of the square, forming a single **measurement plane** which contains the square and all the emitters.

- Four cameras (designated C_1 to C_4, again clockwise from top left to bottom left) are placed on the corners of a second square. The sides of this second square are parallel to those of the first one, and the square is located on a plane, called the **camera plane**, which is parallel to the measurement plane. The cameras point towards the center of the first square.

- Rails enter the system through the center of the second square, their longitudinal axes perpendicular to the measurement plane. While a rail traverses the structure, the cameras are simultaneously triggered every time the rail covers a given distance (s in Figure 3.1), and each of the cameras captures the laser line projected on the rail by its corresponding emitter*. In this way, a camera and its corresponding emitter form a triangulation unit (designated L_n for camera C_n and emitter E_n).

* The corresponding emitter of a given camera, C_n, is the laser emitter E_n.

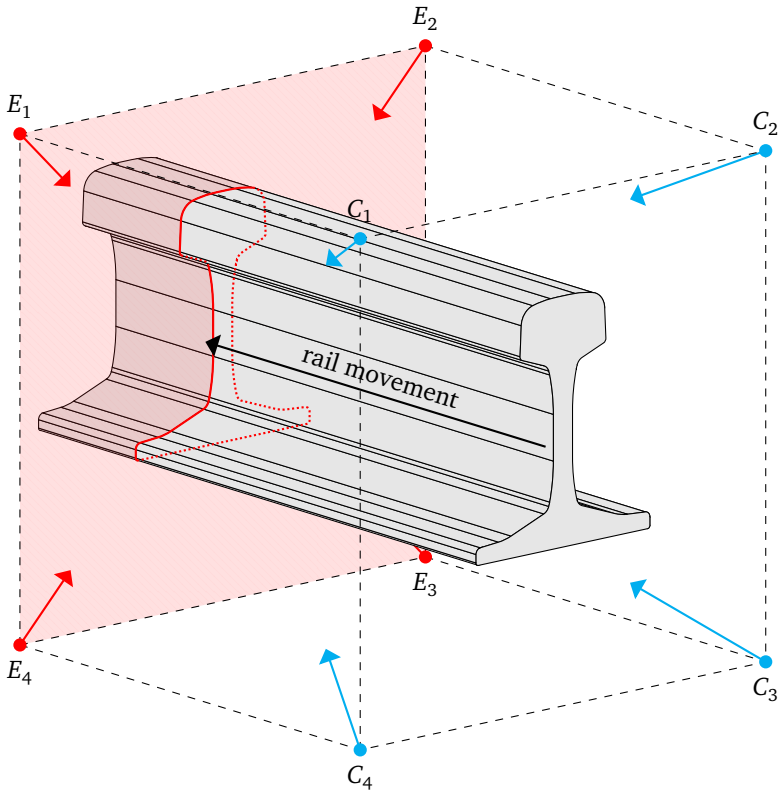

Figure 3.2: Location of the cameras and lasers in the proposed system, relative to the rails. The distances depicted in this figure are not to scale.

- The cameras are connected to a computer, which applies the principles of laser triangulation (see Section 2.2.4.2), extracts the laser lines from the images and their location on the measurement plane and computes the rail dimensions, as described in Section 3.4.

In practice, and especially in harsh industrial environments, mechanical reasons prevent the emitters from actually being placed in such a way that their measurement planes match perfectly. In order for the cameras not to capture points contained in different measurement planes, which would greatly reduce the accuracy of the system, different wavelengths are used for contiguous laser emitters (so that E_1 and E_3 do not use the same wavelength as E_2 and E_4), and band-pass filters are attached to the cameras so that each camera can only capture a narrow range of wavelengths

around that of the corresponding laser emitter.

Apart from this, the four emitters are assumed to form a single measurement plane, unless stated otherwise.

3.3 Logical structure

The proposed system comprises a number of software components, as depicted in Figure 3.3.

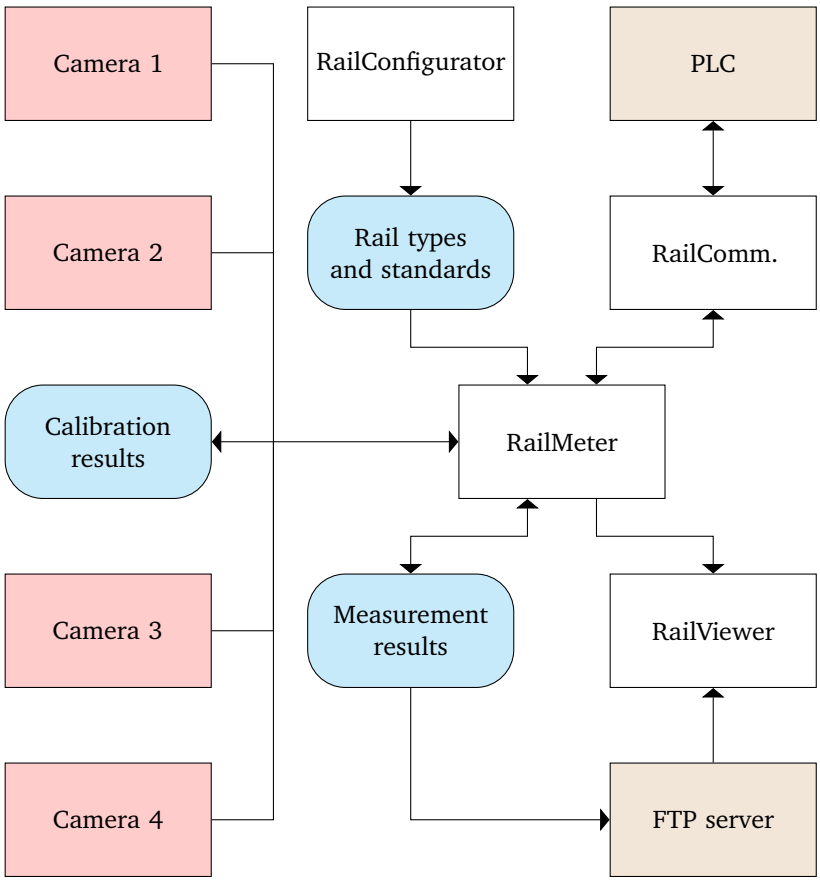

Figure 3.3: Logical structure of the proposed system. In white, software components; in red, cameras; in brown, external systems; in blue, sets of data.

These components can be briefly described as follows:

- **RailMeter:** The RailMeter program is the main software component and implements most of the system functionality described in this chapter. This functionality is split into two modes:
  - **Calibration mode:** The extrinsic camera parameters are computed for all cameras.
  - **Measurement mode:** Camera output is processed; laser triangulation is performed, applying the latest set of extrinsic camera parameters; the dimensions of the resulting contours are computed and a set of measurement results is generated.
- **RailConfigurator:** RailConfigurator allows users to add and modify rail types and standards to the configuration file used by RailMeter. Rail types can be loaded from DXF files depicting the corresponding rail profiles, while standards must be entered by hand.
- **RailViewer:** RailViewer allows users to view both stored measurement results (either through an integrated FTP client or retrieving them from a local file) and the state of rails currently being measured in a remote RailMeter instance.
- **RailCommunication:** The RailCommunication program acts as a gateway between RailMeter and other factory systems, thus insulating RailMeter from the specific protocols used to communicate the type and identifier of incoming rails and the time they will enter and exit the system, for instance.

These system elements produce and/or use the following sets of data:

- **Configuration file:** The configuration file, generated by RailConfigurator, contains a set of rail types and standards for use in RailMeter:
  - **Rail types:** A rail type is a sequence of segments and arcs which form the expected contour of a given rail model. The dimensions of the rail model are computed and stored, to serve as the ideal or expected dimensions for the corresponding rail type.
  - **Standards:** Standards are modeled as sets of dimensional tolerances. For every dimension, a range of valid values is specified by entering the maximum acceptable positive and minimum acceptable negative differences between the measured value and the expected value, computed from the rail model.
- **Calibration results:** Calibration results are automatically stored for every successful calibration performed by RailMeter, and include the extrinsic camera parameters described in Section 4.1.1, along with quality indicators such as the estimated error. However, intrinsic camera parameters must be provided beforehand, as RailMeter cannot compute them.

- **Measurement results:** Measurement results for every rail include the full contours and the values of the dimensions for all its sections. These results are generated by RailMeter and stored for later analysis, either by operators using RailViewer or by automated processes.

3.4 Pipeline

The system must be able to measure rail sections and present the results for each of them as soon as they are available, even before the rail has been processed in its entirety.

In order to do this, a pipeline structure has been devised: A series of threads communicate through queues, each thread (except the initial threads in the pipeline) taking its input from one or more queues and placing its output into a different queue (except the final thread). This pipeline is shown in Figure 3.4.

In this way, the system can endure higher loads for a certain time when needed: if a thread is congested, intermediate work is stored in its input queue to be retrieved as soon as the congestion ends.

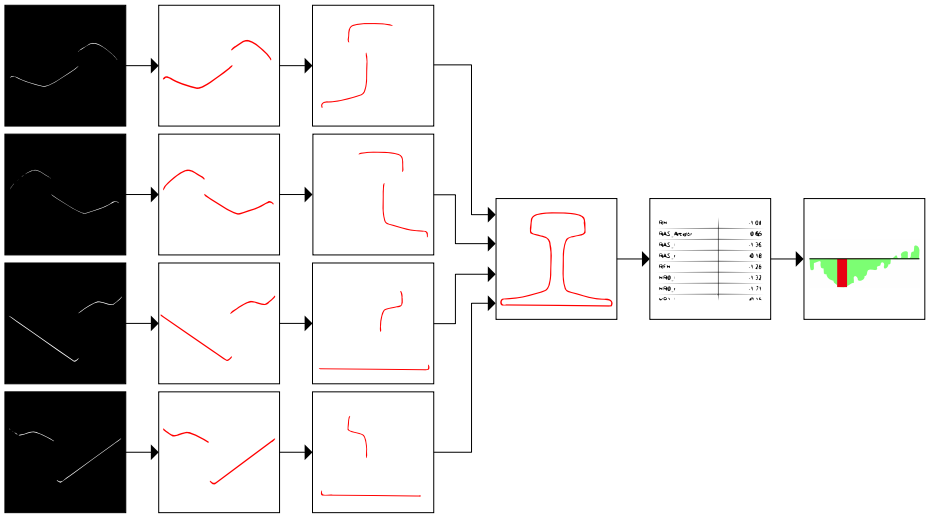

Figure 3.4: The system pipeline. From left to right: image acquisition, laser line extraction, coordinate translation, profile generation, profile measurement and output.

3.4.1 Image acquisition

As the rail moves, digital trigger signals are sent to the cameras from a rotary encoder. Every time the cameras are triggered, each of them captures an image and sends it to the computer. The raw images are similar to the ones that are shown in Figure 3.5.

In this way, the contour of the rail is captured for multiple locations along the rail: the rail sections.

Four acquisition threads and four image queues exist. Each of the acquisition threads retrieves the images from one of the cameras as soon as they are available, and then inserts them into the corresponding image queue.

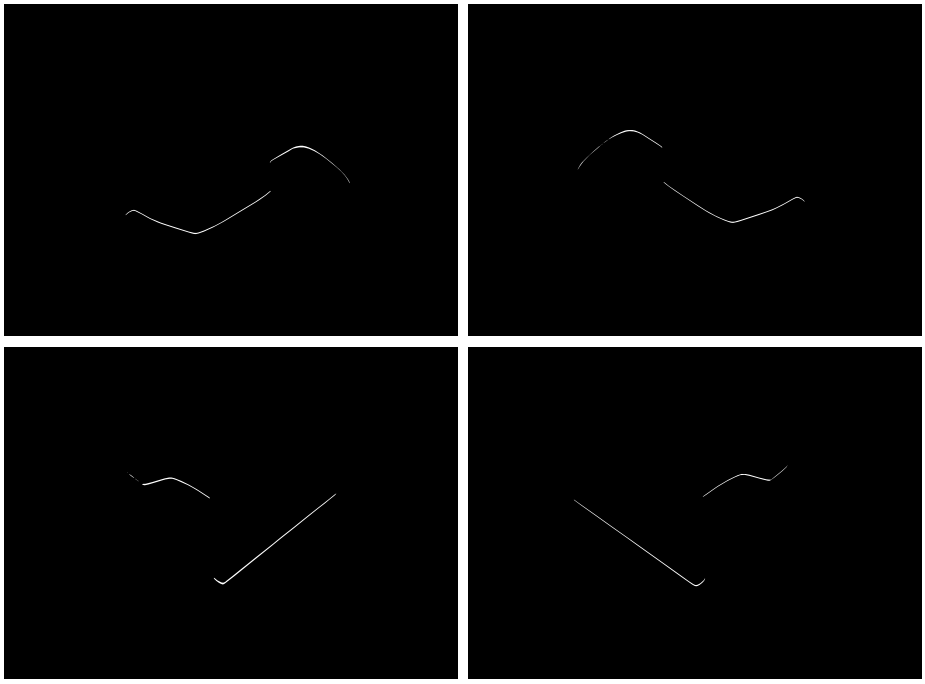

Figure 3.5: A rail profile as seen from the four cameras.

3.4.2 Laser line extraction thread

Four laser extraction threads exist. Each of them takes the images from one of the image queues, finds the laser lines present in the image and stores them in a laser line queue.

Line extraction is performed using a well-known algorithm [Steger, 1996]:

1. A Gaussian smoothing kernel is applied to the image. A value for σ, the standard deviation of the Gaussian distribution*, must be decided beforehand, depending on line size, which in turn depends on the size of the image, the intensity of the laser emitter, the condition of the camera lens and the surface where the laser line is projected.

2. The partial derivatives of the kernel are used to determine the parameters of a quadratic polynomial for every point in the image. These parameters are used to calculate the direction of the line. Then, the points where the second directional derivative along the perpendicular to the line direction reaches a local maximum are considered to be line points.

3. Line points are linked into lines depending on the value of the second derivative. Those where the second derivative is higher than a given threshold are accepted. Those where it is lower than another predefined value are rejected. The rest are accepted if they are connected to accepted points.

4. If multiple lines are present and the distance and the angle between them do not exceed certain thresholds, the lines are linked.

In order to reduce the number of computations required in this step, only the pixels that reach a certain value of gray and those that surround them up to a predefined distance are considered for line extraction.

Finally, the extracted lines are divided into points, and all the points are inserted into a list, called a **point cloud**.

3.4.3 Coordinate translation

A coordinate translation thread exists for every laser line queue. These threads translate the laser lines from image coordinates to world coordinates, and then enqueue the translated lines into the corresponding translated line queue.

This is done by using the mapping described in Section 4.1, applying calibration parameters that must have been previously computed. The calibration algorithms used in RailMeter are described in Section 4.2.2.

3.4.4 Profile generation

The profile generation thread retrieves the translated lines from the four queues and merges them into a single contour, which is then pushed into a profile queue.

The lines can be merged by simply concatenating the four point lists into a single one. However, the system must also ensure that all the translated lines correspond

* In this context, it is often spelled in the Latin script: *sigma*.

to the same rail section (in other words, that the images from which said lines were extracted were taken at the same time). Otherwise, if one or more images from one of the cameras were lost for any reason (such as camera malfunction or queue saturation), then the contours that would be produced afterwards would be a mixture of laser lines from different locations along the rail, rather than an accurate depiction of the rail profile at a given point. In order to guarantee correspondence between images and rail sections, the following algorithm is employed:

1. Wait until either:
   - a translated laser line is available from each of the cameras, or
   - the rail end signal has been received and no images or laser lines remain in the preceding queues.
2. If the first condition from Step 1 was met, continue. Otherwise, discard any remaining translated laser lines and end.
3. Check whether all the sequence numbers match. If they do not match, go to Step 6.
4. Merge the translated lines and enqueue the resulting profile.
5. Go back to Step 1.
6. Discard the translated laser lines, except the one(s) with the highest sequence number.
7. Go back to Step 1.

3.4.5 Profile measurement

The profile measurement thread takes the profiles from the profile queue, performs the actual measurement and pushes the results into a result queue.

This measurement stage, the core of the proposed solution, is described in detail in Section 3.5 and Section 3.6.

3.4.6 Output

The output thread shows the queued results on screen, stores them in a file and sends them to any connected RailViewer instances.

3.5 Preparing for profile measurement

Before the actual measurement can take place, the arcs and segments that form the model must be labeled and the generated profiles must be aligned with the model and converted into a set of primitives. This section describes these preliminary steps.

3.5.1 Rail model primitives

For the purposes of the proposed solution, every rail model is considered to be comprised of a series of arcs and segments, called the **model primitives**. The number of primitives varies among rail models.

As an example, Figure 3.6 shows all the primitives for the 60 E1 A1 rail model.

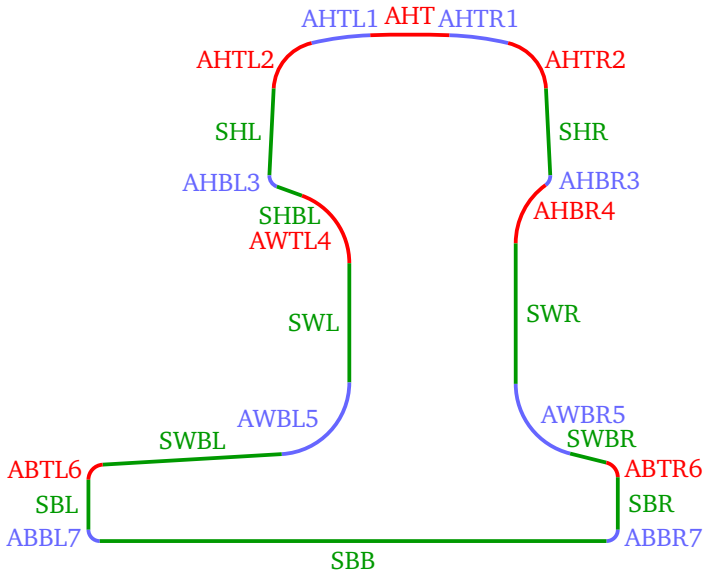

Figure 3.6: A 60 E1 A1 rail profile, decomposed into primitives. The arcs are shown in red and blue, whereas the segments are shown in green. SHBR is not present here, although SHBL is.

Each of the primitives is given a name, according to the following rules:

- The first character of the name is 'A' if the primitive is an arc and 'S' if it is a segment.
- The second character is 'H' for head primitives, 'W' for web primitives and 'B' ('base') for foot primitives.

- 'T' ('top') or 'B' ('bottom') are added to the names of the primitives that are located at the top or bottom of the region they are in, respectively. No letter is added when they are in the middle of the region.
- 'L' ('left') or 'R' ('right') are added to the names of the primitives located on the corresponding side.
- A number* is added to the name of every secondary primitive. These primitives are numbered sequentially from top to bottom, starting at one. The left and right sides are numbered independently from each other.

The **main primitives** are those that determine the shape of the rail. From top to bottom, the main primitives are:

- **AHT:** The central head arc.
- **SHL and SHR:** The segments at the sides of the head.
- **SHBL and SHBR:** The segments at the base of the head. These segments are not mandatory, although they are present in most rail models.
- **SWL or AWL, and SWR or AWR:** The primitives that make up most of the web. They are usually segments, but arcs are used instead in some models.
- **SWBL and SWBR:** The segments at the base of the web.
- **SBTL and SBTR:** The segments at the top of the foot. These segments are not mandatory. When they are not present, SWBL and SWBR generally make up most of the top of the foot. When they are present, there is usually at least one arc between them and the corresponding SWBL and SWBR segments, although in some models the segments are contiguous, forming a concave corner.
- **SBL and SBR:** The segments at the sides of the foot.
- **SBB:** The foot base segment.

The rest of the primitives, called **secondary primitives**, constitute links between the main ones, although they are also relevant for the computation of certain dimensions. Secondary primitives are mostly arcs, while the main primitives are mostly segments, as seen above.

The complete primitive naming algorithm is as follows:

1. The lowest horizontal segment is found and labeled SBB.

*A single digit is enough in most cases, but two are needed for certain rail models that present ten or more secondary primitives to one or both sides.

2. The arc with the highest middle point is found and labeled AHT.
3. Two ordered lists, left side and right side, are created.
4. For every unlabeled primitive:
   4.1. If it is contiguous to AHT, append it to the end of one of the lists, depending on whether it is located to the left or to the right of AHT.
   4.2. If it is contiguous to a primitive in one of the lists, append it to the end of that list.
5. For each side $s \in \{L, R\}$, the primitives from the corresponding list are taken in order and:
   5.1. Let n be one plus the number of secondary primitives to this side that have already been labeled.
   5.2. All the arcs until the first segment are labeled AHTsn.
   5.3. The first segment is labeled SHs.
   5.4. All the arcs until a segment is found or the accumulated angle since the last segment $\geq 100°$ are labeled AHBsx.
   5.5. If a segment was found, it is labeled SHBs.
   5.6. The arc or vertical segment that is closest to the nearest arc or vertical segment to the other side is labeled as AWs or SWs, respectively.
   5.7. All the unlabeled primitives that precede AWs or SWs are labeled tWTsx, where t is their type.
   5.8. The first segment that appears in the list after the accumulated angle of the preceding arcs $\geq 45°$ is labeled SWBs.
   5.9. All the unlabeled primitives that precede this segment are labeled tWBsx, where t is their type.
   5.10. All the arcs until a segment is found or the accumulated angle since the last segment $\geq 45°$ are labeled ABTsx.
   5.11. If a segment was found, it is labeled SBTs.
   5.12. The vertical segment that is furthest from the corresponding vertical segment to the other side is labeled SBs.
   5.13. All the unlabeled primitives that precede this segment are labeled tBTsx, where t is their type.
   5.14. All the remaining unlabeled primitives are labeled tBBsx, where t is their type.

This labeling process is carried out when first loading a new rail model in RailConfigurator.

3.5.2 Point cloud registration

After a 3D transformation has been applied to the laser lines and they have been merged into a single profile, as described in Section 3.4.4, this profile must be aligned with the corresponding rail model, by applying a 2D transformation. The location of the rail profiles in the measurement plane varies with time, because the rail shakes as it moves through the system. Therefore, this 2D transformation must be computed for every profile.

Figure 3.7 shows a rail profile before and after applying the alignment transformation.

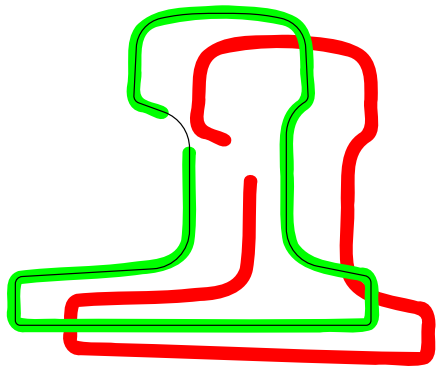

Figure 3.7: An example of profile alignment. In red, the unaligned profile point cloud; in green, the aligned profile point cloud; in black, the rail model (54 E1 A1).

The process of computing a transformation to align two point clouds is called point cloud **registration** or matching. Registration algorithms can be applied to both 2D and 3D points*. Registration may be rigid or non-rigid, depending on the nature of the resulting transformations. In the case of the proposed system, rigid 2D transformations are required, so registration is relatively simple.

One common, well-known registration algorithm is Iterative Closest Point (ICP) [Besl and McKay, 1992]. The basic ICP algorithm is as follows:

1. For every point in the origin point cloud, the nearest point in the destination point cloud is found. The points are now considered corresponding points.

2. A transformation from the origin point cloud to the destination point cloud is computed that minimizes the distance between corresponding points, using a least-squares approach.

* 4D registration also has some applications. For instance, the 4th dimension may represent the hue of the points [Men et al., 2011].

3. The computed transformation is applied to the origin point cloud.

4. This is repeated until convergence is achieved. Convergence is usually considered to have been achieved if a certain number of iterations has been reached, if the average distance between corresponding points is smaller than a given threshold or if the difference between the last two transformation is smaller than a predefined value.

The proposed system employs the ICP algorithm to align the rail profiles to the models, with the following peculiarities:

- Before applying ICP, the origin point cloud is translated so that its centroid* matches the centroid of the model. This initial alignment reduces the number of ICP iterations required for convergence.

- As the models are sets of arcs and segments, rather than points, ICP cannot be directly applied.

  This can be solved by sampling the model in order to build a reference point cloud, and then applying ICP to that point cloud. However, with this approach accuracy is reduced depending on the sampling density [Usamentiaga et al., 2015].

  Instead, the whole model is used instead of a discrete destination point cloud. In every iteration of the algorithm, for each point in the origin point cloud the closest point in the closest arc or segment is computed, and this point is used as the corresponding point.

- Outlier point correspondences, defined as those where the distance between the points is higher than the median distance between pairs of corresponding points by a predefined factor, are discarded.

- Performance is improved [Usamentiaga et al., 2015] by using an R-tree structure [Guttman, 1984] when searching for correspondences.

  R-trees are data structures optimized for spatial searching. More specifically, they are height-balanced trees whose leaves represent objects in 2D space, whereas their non-leaf nodes represent rectangles that contain all their children.

  An R-tree is built that contains bounding boxes for all the primitives in the model, with a predefined amount of padding. In order to check whether a point is contained in any of the primitives, the tree is traversed and any

* The **centroid** is defined here as the point whose X coordinate is the arithmetic mean of the X coordinates of all the points in the point cloud, and whose Y coordinate is the arithmetic mean of their Y coordinates.

nodes whose rectangle does not contain the point are discarded, along with all their descendants. If any of the leaf nodes contains the point, the primitive associated with the leaf node is checked as a possible correspondence.

3.5.3 Fitting

Once the point cloud is aligned with the rail model, and before the values of the dimensions can be computed, the primitives are fit to points. These **fitting primitives** are circles and straight lines.

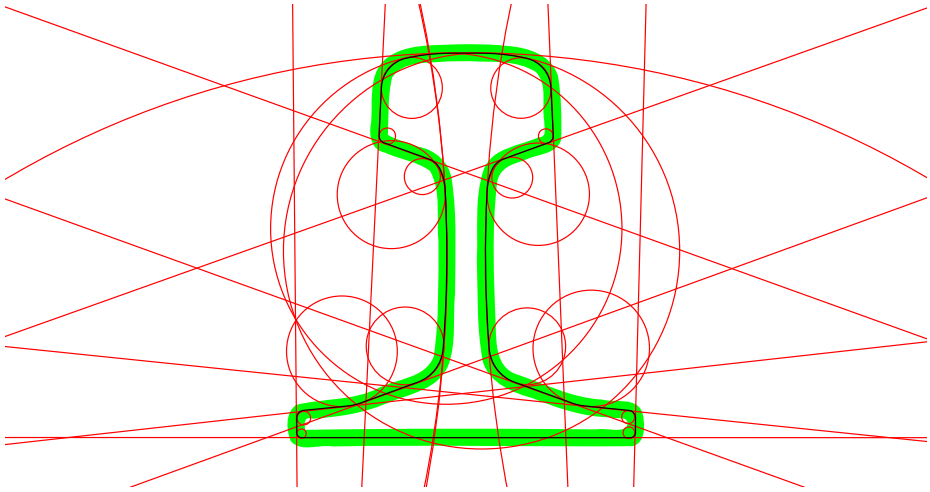

Figure 3.8: An example of profile fitting. In black, the rail model (54 E1); in green, the (aligned) profile point cloud; in red, the fitting primitives.

The result of fitting primitives to the entire point cloud in such a way that each circle or line corresponds to one arc or segment in the model, respectively, can be called a **fit profile**.

Most dimensions are easier to compute by working with circles and lines than with point clouds. For many of them, the primitives that form the fit profile suffice. However, for some dimensions, additional fitting must be done. For instance, foot concavity cannot be computed by merely fitting a single line to all the foot base points, corresponding to the SBB model primitive. Instead, the base must be split into smaller segments, and lines corresponding to those segments are fit to the points.

Figure 3.8 shows a rail point cloud and its fit profile.

In order to create the fit profile, an attempt is made to associate every point in the point cloud with a model primitive. Each model primitive is associated with all the profile points in its enveloping region. Points located in the overlap between the

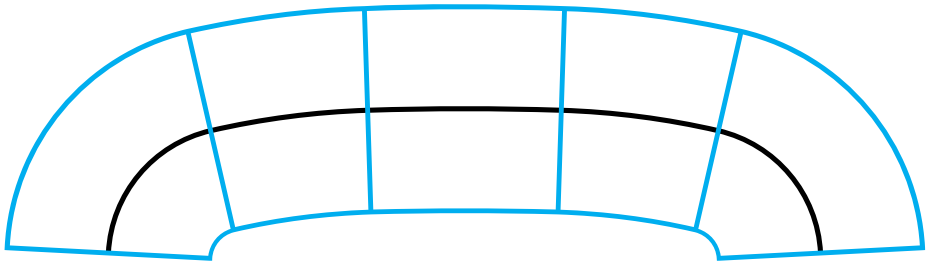

Figure 3.9: Enveloping regions for some model primitives. In black, some of the primitives that form the head of the 54 E1 rail model; in blue, the borders of the enveloping regions for the depicted primitives.

enveloping regions of multiple model primitives are associated with the closest of those primitives. Points located outside the enveloping regions of all primitives are not considered for fitting.

As depicted in Figure 3.9, a point is considered to be contained in the **enveloping region** of a given model primitive if and only if it satisfies both of the following conditions:

- Its distance to the primitive is shorter than a predefined value. This value has been set to 10 mm, big enough to contain all points that could be possible part of a given region of the rail.
- No part of the straight line or circle that contains the primitive is closer to the point than the primitive is.

After this is done, a fitting primitive is created for each of the model primitives, provided they are associated to enough points:

- **Lines:** Linear regression is used to fit a line to the points. The $ax + by + c = 0$ line model is used because it allows for vertical lines (whereas the $mx + n = 0$ model does not). Only the points are taken into account: the location and slope of the model segment do not affect the result. At least two points are needed.
- **Circles:** In order to fit a circle to the points, an iterative approach is employed. The center and radius of the model arc are used as an initial guess. Then, least-squares optimization is used to minimize

$$\sum_{i=1}^{n}(d_i - r)^2, \text{ where } d_i = \sqrt{(x_i - x)^2 + (y_i - y)^2},$$

r is the radius of the circle, (x, y) is its center, (x_i, y_i) is the location of point i and n is the number of points. At least three points are required.

3.6 Measuring the rail dimensions

Profile measurement computes the values of a number of rail dimensions for every rail section. This section describes all the rail dimensions considered by the system, and the way they are measured. It must be noted that the names given here are the names of the dimensions in the system, and do not match those used in the standards.

Not all of the dimensions apply to all the rail models. For instance, HR0_l and HR0_r cannot be computed for rail models whose head has less than three arcs to the left and right of the highest head arc, respectively.

Additionally, fitting may fail for some model primitives when the number of rail section points associated with them is not enough. This happens when the laser lines are weak or missing altogether from the images. When this happens, the dimensions that rely on the affected primitives cannot be computed.

The dimensions as described here can be applied to both the rail model and the rail section profiles. In fact, the dimensions must be computed for the rail model profiles when they are loaded into RailConfigurator, in order to later compare their values with the corresponding values from the rail section profiles.

Unless stated otherwise, whenever the name of a primitive is used in this section, it is to be interpreted as the rail model primitive with that name, when computing the dimensions of the rail model profile, or to the corresponding fitting primitive, when computing the dimensions of the rail section profile.

'Left' and 'right' refer to the corresponding side of the rail as seen from the point where the rail enters the system, looking toward the measurement plane.

Figure 3.10 and Figure 3.11 show most of the dimensions defined here.

- **RH:** Rail height, in mm.

  This is the vertical distance between the base of the foot (SBB) and the highest point of the top head arc (AHT).

- **RAS_Arcelor:** Arcelor rail asymmetry, in mm.

  This is the horizontal distance between the middle point of the highest head arc (AHT) and the point of the base of the foot (SBB) that is halfway between the foot sides (SBL and SBR).

  When computing this dimension from fitting primitives, the middle point of AHT is defined as the point where the AHT fitting circle intersects the line that joins the middle point of the AHT model arc and the center of said arc.

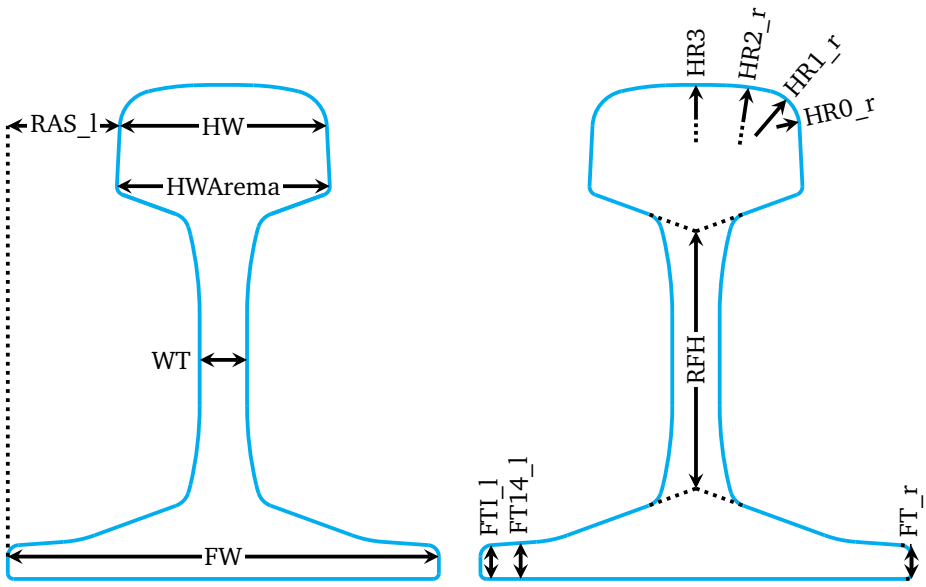

Figure 3.10: Some of the dimensions, shown on a 60 E2 rail profile.

- **RAS_l and RAS_r:** Rail asymmetry, in mm.

  These are the horizontal distances between the head side segment (SHL or SHR) and the foot side segment (SBL or SBR) of the respective side.

  When computing these dimensions from fitting primitives, the distance is measured from the point where the SHL or SHR line meets the line that joins the ends of the model arc to the furthest of the points where the SBL or SBR line meets the horizontal lines that go through the ends of the model SBL or SBR segments.

- **RFH:** Rail flange height, in mm.

  This is the distance between the intersection of the head base lines (the lines that correspond to the segments where the head meets the web: SHBL and SHBR) and the intersection of the foot top lines (the lines that correspond to the segments where the foot meets the web: SWBL and SWBR).

- **HR0_l and HR0_r:** Head radius zero, in mm.

  These are the radii of the third head arcs to the left and right of the highest head arc, respectively (AHTL3 and AHTR3).

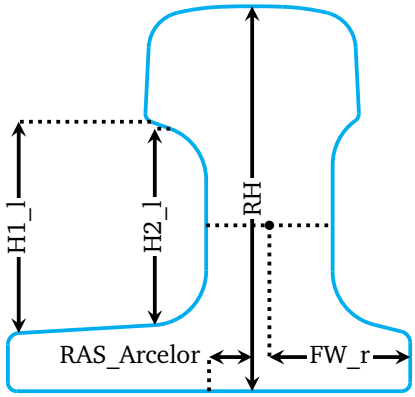

Figure 3.11: Some of the dimensions, shown on a 60 E1 A1 rail profile.

When computing these dimensions for fitting primitives, the distance between the center of the model arc and the point of the fitting circle where it intersects the line between the center of the model arc and the top point of the model arc is used.

- **HR1_l and HR1_r:** Head radius one, in mm.

  These are the radii of the second head arcs to the left and right of the highest head arc, respectively (AHTL2 and AHTR2).

  These dimensions are computed in the same way as HR0_l and HR0_r.

- **HR2_l and HR2_r:** Head radius two, in mm.

  These are the radii of the head arcs immediately to the left and right of the highest head arc, respectively (AHTL1 and AHTR1).

  These dimensions are computed in the same way as HR0_l, HR0_r, HR1_l and HR1_r.

- **HR3:** Head radius three, in mm.

  This is the radius of the highest head arc (AHT).

  It is computed in the same way as HR0_l, HR0_r, HR1_l, HR1_r, HR2_l and HR2_r.

- **HF1:** Head form with one arc, in mm.

  This is the sagitta of the highest head arc (AHT): the distance between its middle point and the center of its base (the segment that joins the ends of the arc).

In the case of the fit profile, this is computed as the distance between the center of the base of AHT in the model and the point where the line that joins the center of AHT in the model with its middle point meets the fitting circle.

- **HF3:** Head form with three arcs, in mm.

  This is the distance between the highest point of the highest head arc (AHT) and the segment between the lower ends of the two arcs immediately neighboring it (AHTL1 and AHTR1).

  In the case of the fit profile, this is computed as the distance between the center of the segment that joins the left and right ends of the model arcs AHTL1 and AHTR1, respectively, and the point where the line that joins the center of AHT in the model with its middle point meets the fitting circle.

- **HW:** Head width, in mm.

  This is the horizontal distance between the upper ends of the side head segments (SHL and SHR).

  When computing this dimension from fitting primitives, this is the distance between the points where the SHL and SHR fitting lines meet the line that joins the upper ends of the corresponding model segments.

- **HWLocal:** Local head width, in mm.

  This is the horizontal distance between the upper ends of the side head segments (SHL and SHR), except that only the upper half of the sides is taken into account when computing the segments.

  When computing this dimension from a rail section profile, lines are fit to the points associated with the upper half of SHL or SHR, and these lines are used in the same way as the fitting lines are used for HW. When computing the dimension for the rail model, its value is that of HW.

- **HWArema:** Head width according to the AREMA standard, in mm.

  This is the horizontal distance between the lower ends of the side head segments (SHL and SHR).

  When computing this dimension from fitting primitives, this is the distance between the points where the SHL and SHR fitting lines meet the line that joins the lower ends of the corresponding model segments.

- **HWAremaLocal:** Local head width according to the AREMA standard, in mm.

  This is the horizontal distance between the lower ends of the side head segments (SHL and SHR), except that only the lower half of the sides is taken into account when computing the segments.

When computing this dimension from a rail section profile, lines are fit to the points associated with the lower half of SHL or SHR, and these lines are used in the same way as the fitting lines are used for HWArema. When computing the dimension for the rail model, its value is that of HWArema.

- **WT:** Web thickness, in mm.

  This is the horizontal distance between the rightmost point of the left side of the web (SWL or AWL) and the leftmost point of the right side of the web (SWR or AWR).

  When computing this dimension from a fit profile, this is the distance between the points where the SWL and SWR fitting circles, or the AWL and AWR fitting arcs, meet the line that joins the middle points of their corresponding model primitives.

- **FW:** Foot width, in mm.

  This is the distance between the sides of the foot (SBL and SBR).

  When computing this dimension from a rail section profile, this is the horizontal distance between the leftmost of the points where the SBL fitting line meets the horizontal lines that go through the ends of the model SBL or SBR segments and the rightmost of the points where the SBR fitting line meets those lines.

- **FW_l and FW_r:** Left and right foot half-widths, in mm.

  These are the horizontal distances between the respective side of the foot and the middle of the rail, defined as the halfway point between the rightmost point of the left side of the web and the leftmost point of the right side of the web.

  When computing these dimensions from a rail section profile, they are the horizontal distance between, on one side, the center point of the line that joins the middle points of SWL and SWR or AWL and AWR and, on the other side, the outermost points where SBL and SBR, respectively, meet the horizontal lines used for the computation of FW.

- **FC:** Foot concavity, in mm.

  This is the distance between the highest point of the foot base and the segment that connects the lower ends of the foot sides. This is always zero for the rail model profiles, as their foot bases are completely flat, but actual rail sections may present a certain amount of concavity.

  This dimension is relatively complex. In order to compute it for a rail section profile, the base segment (SBB) is divided into 10 subsegments, and these subsegments are fit to the points associated with SBB. The lengths of each of the vertical segments that start on the SBB fitting line, pass through the middle

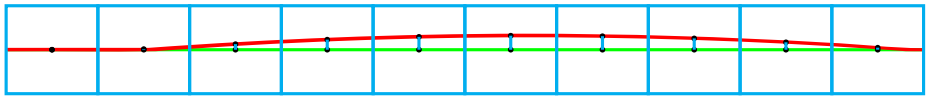

Figure 3.12: A concave rail foot.

point of one of the subsegments and end on the lines that result from fitting said subsegment to the points are calculated. Then, the maximum such length is used as a value for FC. This is shown in Figure 3.12.

- **FT_l and FT_r:** Foot thickness, in mm.

  These are the vertical distances between the foot top lines (SBTL and SBTR if they exist, otherwise SWTL and SWTR) and the line at the base of the foot (SBB), measured over the foot side line of the respective side (SBL and SBR).

- **FT14_l and FT14_r:** Foot thickness at 14 mm, in mm.

  These are the vertical distances between the foot top lines (SBTL and SBTR if they exist, otherwise SWTL and SWTR) and the line at the base of the foot (SBB), measured over a line parallel to the foot side line of the respective side (SBL and SBR), located at a distance of 14 mm of said foot side line, going toward the middle of the rail.

- **FTI_l and FTI_r:** Internal foot thickness, in mm.

  These are the vertical distances between the lower end of the foot top segment (SBTL and SBTR if they exist, otherwise SWTL and SWTR) of the respective side and the respective end of the base segment (SBB).

  When computing these dimensions from a rail section profile, the distances are measured over the vertical lines that pass through the ends of the model SBB segment.

- **H1_l and H1_r:** Flange one, in mm.

  These are the vertical distances between the upper end of the head base segment (SHBL or SHBR) and the lower end of the foot top segment (SWBL or SWBR) of the respective side.

  When computing these dimensions from a rail section profile, they are the vertical distances between, on one side, the points where the SHBL and SHBR fitting lines meet the vertical lines that pass through the upper ends of the SHBL and SHBR model primitives and, on the other side, the points where the SWBL and SWBR fitting lines meet the vertical lines that pass through the lower ends of the corresponding model primitives.

- **H2_l and H2_r:** Flange two, in mm.

  These are the vertical distances between the lower end of the head base segment (SHBL or SHBR) and the upper end of the foot top segment (SWBL or SWBR) of the respective side.

  When computing these dimensions from a rail section profile, they are the vertical distances between, on one side, the points where the SHBL and SHBR fitting lines meet the vertical lines that pass through the lower ends of the SHBL and SHBR model primitives and, on the other side, the points where the SWBL and SWBR fitting lines meet the vertical lines that pass through the upper ends of the corresponding model primitives.

- **AREA:** Area of the profile, in cm^2.

  A numerical approach is used to compute the area. First, a polygon is generated that represents the contour of the rail. Then, the following well-known formula* is used to determine the area of the polygon:

$$A = \frac{1}{2}\left(\begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \begin{vmatrix} x_2 & x_3 \\ y_2 & y_3 \end{vmatrix} + \ldots + \begin{vmatrix} x_n & x_1 \\ y_n & y_1 \end{vmatrix}\right)$$

  where x_i and y_i respectively designate the X and Y coordinates of the vertex in position i. Contiguous positions are occupied by contiguous vertices, and the vertex in the first position is also contiguous to the vertex in the last position.

  When computing the area of the rail model, the polygon is obtained by sampling the model with a distance of 0.5 mm between contiguous points.

  When computing the area of a rail section profile, the vertices of the polygon are the centroids of the points in the rail section point cloud that are at a maximum distance of 1 mm of a point in the sampled model point cloud. Points in the sampled model point cloud for which less than 3 rail section points are present within that distance are ignored. The area is not computed if the number of polygon vertices is less than 75 % of the number of points in the sampled model point cloud.

* This is sometimes called the shoelace formula or the surveyor's formula.

Chapter 4

Calibration

4.1 Background

In order to achieve highly accurate measurements, most machine vision techniques require the camera or cameras to be calibrated. This is especially true in the case of optical triangulation.

The output of the calibration process is a mapping between the **image coordinate system** and the **world coordinate system** (a coordinate system defined so that the plane $Z = 0$ is the measurement plane). This mapping can then be used to translate pixels in the images into points in the measurement plane, and vice versa.

This process should not be confused with other similarly named calibration processes, such as color calibration, which determines a mapping between the output of the camera sensors and the corresponding color values.

4.1.1 Mapping image coordinates and world coordinates

In order to adequately understand the calibration process, we must first understand the nature and usage of its results.

We will describe a possible mapping between the image coordinate system and the world coordinate system, using the OpenCV library as an example [OpenCV, 2017, Camera calibration and 3D reconstruction (calib3d module)]. While other computer vision frameworks* differ in their calibration models in small, at times subtle, ways**, they operate on the same principles.

4.1.1.1 The camera matrix

The simplest model for the relationship between a point in 3D space and its projection on an image plane (its coordinates in the image coordinate system) is the pinhole camera model. In this model, the camera has no lens, and its aperture is considered to be a point.

* Such as MVTec HALCON and the MATLAB Computer Vision System Toolbox.

** Most of these differences apply to the way distortion is modeled and represented.

By taking into account the focal length (as defined below), we can translate the image coordinates into a **camera coordinate system** centered on the camera aperture.

As our goal is to convert points in the image to points on the measurement plane, we must also take into account the location of the camera relative to the plane. This can be done by applying a rotation and a translation to the coordinates in the camera coordinate system.

Let (u, v) be the coordinates in the image coordinate system, given in pixels, and (X, Y, Z) the coordinates in the world coordinate system, given in world units (usually meters or millimeters), where the measurement plane is defined as $Z = 0$. Then:

$$w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R \mid T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The 3×4 matrix $P = K \begin{bmatrix} R \mid T \end{bmatrix}$ is usually called the **camera matrix**. Here, R is a 3×3 rotation matrix which represents the 3D rotation of the camera and T is a column matrix which expresses the translation of the camera relative to the world coordinate system. w is an arbitrary scale factor.

In order to define K, the camera intrinsic matrix, the following parameters must be described first:

- **Focal length:** The focal length is the distance between the camera aperture and the plane where the image is formed. If F is the focal length expressed in world units, then f_x and f_y designate the focal length in pixel widths and heights, respectively. As camera pixels are roughly square, f_x and f_y should be very similar.

- **Principal point:** c_x and c_y designate the pixel coordinates of the principal point of the image. The principal point, also called the optical center, is the point in the image plane that is closest to the camera aperture. This is roughly the geometric center of the image.

- **Skew:** γ is the skew coefficient of the camera, defined as $\gamma = f_y \tan(\alpha)$, where α is the angle between the y axis of the camera sensor and the perpendicular to the x axis. It follows that when the axes are perpendicular, then $\gamma = 0$. OpenCV calibration does not model skew, so this value is set to 0 in the K matrix.

Using these parameters, K can be defined as:

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

4.1.1.2 Lens distortion

As stated before, the pinhole camera model lacks a lens, which means that it cannot account for lens distortion. Therefore, lens distortion must be modeled separately.

Distortion can be thought of as anything that causes straight lines appear curved in the image. Distortion is most apparent in images taken with inexpensive lenses, such as the lenses of low-end webcams, and with lenses that are deliberately built to produce distortion, like fisheye lenses; however, all lenses present some degree of distortion, which must be taken into account when high accuracy is required.

The most prominent distortion patterns [Tsai, 1987; Heikkilä and Silvén, 1997], and the only ones OpenCV and other computer vision frameworks model, are:

- **Radial distortion:** Radially symmetric distortion is caused by the shape of the lens itself, and can be classified as barrel distortion (when magnification is higher at the optical center than at the edges) or pincushion distortion (when magnification is lower at the optical center).

  Radial distortion can be modeled as follows. Let (x, y) be a pair of original, distorted coordinates in the camera coordinate system:

  $$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  and $r = \sqrt{x^2 + y^2}$. Then:

  $$\delta_{r_x} = x(k_1 r^2 + k_2 r^4 + k_3 r^6 + ...)$$

  $$\delta_{r_y} = y(k_1 r^2 + k_2 r^4 + k_3 r^6 + ...)$$

  OpenCV and most commercial frameworks use an order-3 polynomial to model radial distortion, while others use just 2 coefficients. Employing more than 3 coefficients is not typically necessary to compensate for radial distortion [Heikkilä and Silvén, 1997].

- **Tangential distortion:** Tangential distortion appears because the lens and the image plane are not perfectly aligned. It is usually a secondary concern, as its influence is significantly smaller than that of radial distortion.

Tangential distortion can be modeled as follows. Again, let (x, y) be a pair of distorted coordinates in the camera coordinate system, in the same way as before, and $r = \sqrt{x^2 + y^2}$:

$$\delta_{t_x} = 2p_1xy + p_2(r^2 + 2x^2)$$

$$\delta_{t_y} = 2p_2xy + p_1(r^2 + 2y^2)$$

Thus, OpenCV's distortion coefficients are k_1, k_2, k_3, p_1 and p_2, and the relationship between distorted and corrected* camera coordinates can be given as:

$$x_{corrected} = x + \delta_{r_x} + \delta_{t_x}$$

$$y_{corrected} = y + \delta_{r_y} + \delta_{t_y}$$

with (x, y) defined as above.

These formulae are not analytically invertible. As such, the task of adding distortion (which is required when translating world coordinates for image coordinates) must be performed numerically**.

More complete distortion models exist that take into account additional distortion patterns, like prism distortion, caused by imperfections in the lens. However, no popular computer vision framework takes into account such complex models. Alternatives to the polynomial approach described here [Fitzgibbon, 2001] also exist.

4.1.2 Finding the calibration parameters

The calibration parameters described in Section 4.1.1 are usually separated into two categories:

- **Intrinsic parameters:** Intrinsic parameters are the parameters that depend exclusively on the camera and its lens. These parameters do not vary for a given camera unless its lens is replaced or certain settings are changed. In the case of OpenCV, these parameters are the distortion coefficients k_1, k_2, k_3, p_1 and p_2 along with the values contained in the camera intrinsic matrix K: f_x, f_y, c_x and c_y.
- **Extrinsic parameters:** Extrinsic parameters are the parameters that describe the location and bearing of the camera relative to the world coordinate system. They are also known as the camera pose. Extrinsic parameters must be found again if the camera or the world coordinate system are moved in any way. The

* 'Corrected' coordinates are usually called 'undistorted' in other works, but the OpenCV documentation employs the former term, which also eliminates a possible ambiguity.

** However, certain frameworks employ the opposite model, where distortion is added analytically and removed numerically.

extrinsic parameters are the translation vector and the rotation matrix (or an alternative representation, such as a Rodrigues vector).

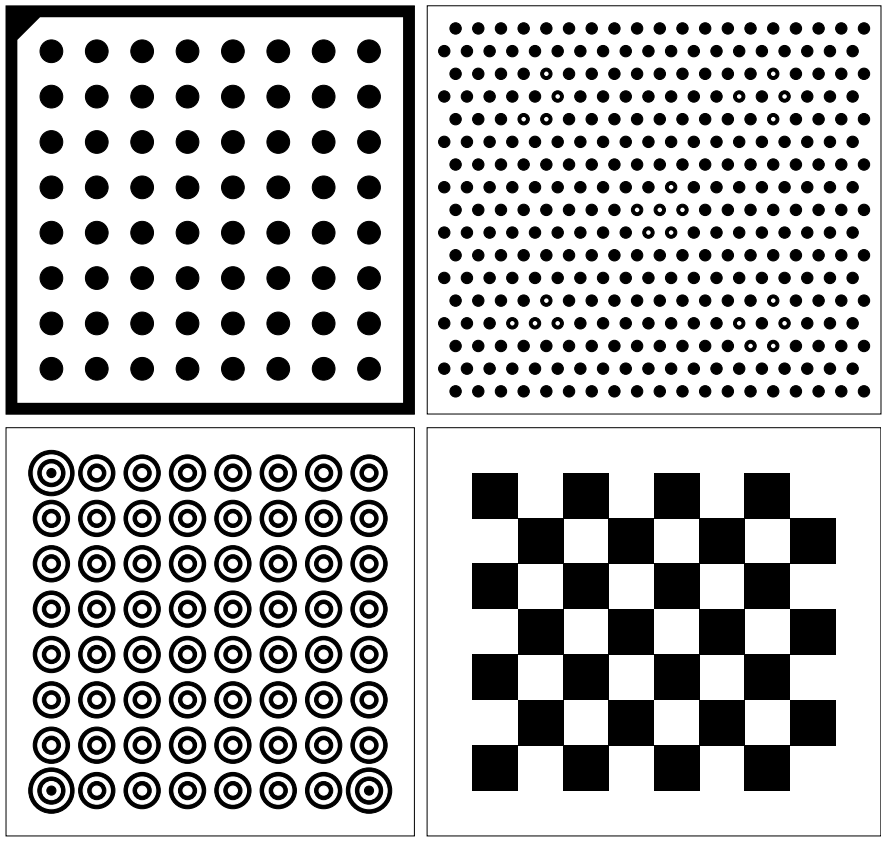

Figure 4.1: Four examples of calibration patterns. From left to right, then from top to bottom: two circle patterns, similar to the MVTec HALCON calibration targets; a concentric rings pattern; and the checkerboard, a common pattern used with frameworks such as OpenCV and the MATLAB Computer Vision System Toolbox.

Calibration is performed by capturing certain features of a known calibration object, and building a mapping between the location of these features as they appear in the image, or images, with their location on the mentioned calibration object.

This calibration object usually takes the form of a planar surface where a predetermined pattern of circles or squares is printed. Figure 4.1 depicts four such patterns.

It must be noted that the checkerboard shown there is not square, but an 8×7 rectangle. Checkerboards whose width and height are both even or both odd should not be used for this purpose because they are rotationally symmetric* and therefore ambiguous. The other depicted patterns avoid rotational symmetry by adding marks at different locations**.

The calibration process in OpenCV and other modern computer vision frameworks is based on Zhang's camera calibration algorithm [Zhang, 2000]:

1. Multiple images*** of the calibration object are taken, each of them from a different angle or in a different position.

2. Some features present on the calibration object (when a pattern of circles is used, these are typically their centers; when squares are used, their corners) as shown in the images are detected, and mappings between the locations of the features in the images and their locations on the calibration object are computed.

3. Using a closed form analytic solution, values for the camera intrinsic matrix K (a single set of values) and the extrinsic parameters (a set of values for each image) are estimated from the mappings. Lens distortion cannot be estimated at this point, as it cannot be solved analytically.

4. A maximum likelihood estimation for the full set of parameters is computed with the Levenberg-Marquardt algorithm, using the parameters that were found before as initial values.

As stated before, intrinsic parameters generally need only be computed once, while extrinsic parameters may have to be computed periodically. When only extrinsic parameters are needed, calibration can be performed in the following way:

1. A single image of the calibration object is captured.

2. The relevant features are found in the image.

3. The known intrinsic parameters are applied in order to translate the locations of the features from the image coordinate system into the camera coordinate system.

* Specifically, they remain the same when rotated by an angle of 180°; by an angle of 90° in the case of square checkerboards whose sides have an odd number of tiles.

** In the case of the top right circle pattern, these marks also serve the purpose of indicating which portion of the pattern is within the field of view as, unlike other patterns, it need not be fully visible to the camera for the calibration process to succeed.

*** The optimal number largely depends on the pattern and the quality of the images; 15 images are usually enough.

4. The extrinsic parameters that minimize the distance between the actual, predefined world coordinates of the features and their computed world coordinates are found. The Levenberg-Marquardt algorithm can be applied to this minimization problem, just as before.

This limited form of calibration can be called extrinsic calibration.

4.2 Calibrating the proposed system

In the case of the proposed system, the movement of rails through the structure causes the location of the cameras to suffer small, cumulative changes which quickly reduce the accuracy of the measurements, and eventually prevent the system from measuring altogether. Therefore, the system must include an extrinsic calibration procedure, in order to enable the users to periodically update the camera poses.

As for intrinsic calibration, it must be performed to find the intrinsic parameters of the cameras when first setting up the system and when replacing the cameras or their lenses. However, this can be done using existing tools, so a specific intrinsic calibration feature is not required.

4.2.1 The calibration target

General-purpose calibration patterns, such as those shown in Figure 4.1, are not suitable for calibrating this system, as that would require the calibration plates to be placed precisely on the measurement plane, so that the computed camera poses, relative to the plates, would match the camera poses relative to said plane.

Asking users to manually place the plates would not be practical, nor would it result in accurate poses. A robotic arm could be used in order to accurately place them and remove them when needed, but it would be prohibitively expensive, and itself require calibration.

Therefore, an alternative approach is required. As shown in Figure 4.2, a calibration target is used that is comprised of 13 cylinders that protrude from a surface. In order to perform extrinsic calibration, this surface must be placed roughly parallel to the measurement plane, and in such a way that the cylinders intersect said plane. In this way, the contour of each cylinder in the measurement plane is circular or near circular.

It should be noted that this calibration pattern is not an original contribution of this work, as similar patterns have been used for years in laser triangulation systems, although few references to them have been made in the literature.

Unlike the patterns from Figure 4.1, this specific pattern is rotationally symmetric, as it remains the same when rotated by an angle of 180°. This causes an ambiguity that must be taken into account when performing the extrinsic calibration.

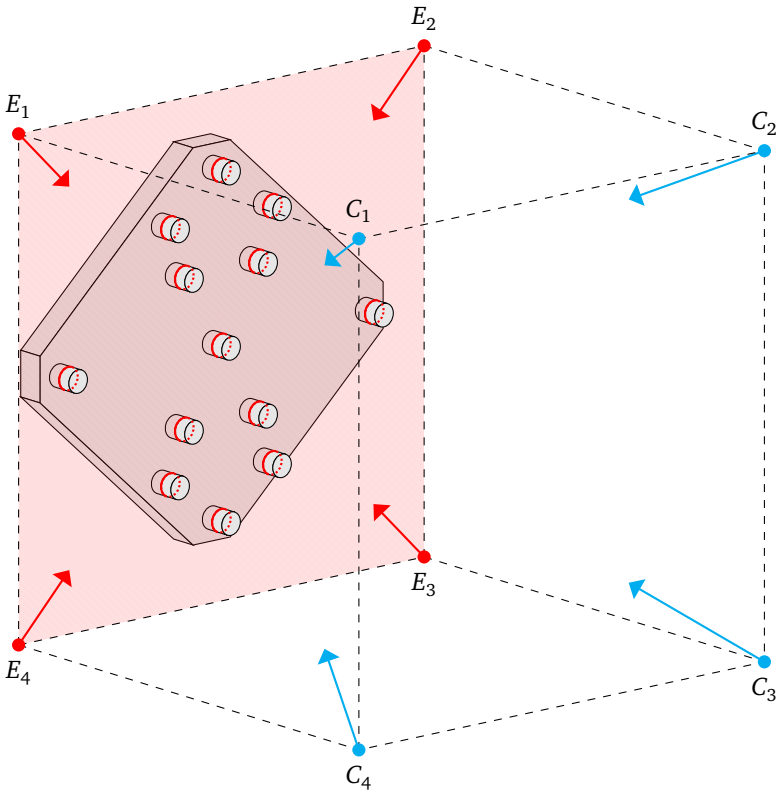

Figure 4.2: The calibration target, shown in its position relative to the cameras and the measurement plane. The depicted distances are not to scale.

Although alternate cylinder configurations could be used that would avoid this issue, finding (and then implementing and testing) an alternate configuration that does not present rotational symmetry and where cylinders do not occlude the laser emitters, while ensuring accuracy in the relevant areas of the measurement plane, would be a significantly complex task. As such, and because this pattern has a specific purpose and its general position relative to the cameras will always be known, compensating for this issue later is more feasible.

4.2.2 Extrinsic calibration

Two algorithms have been devised to carry out extrinsic calibration for the proposed system.

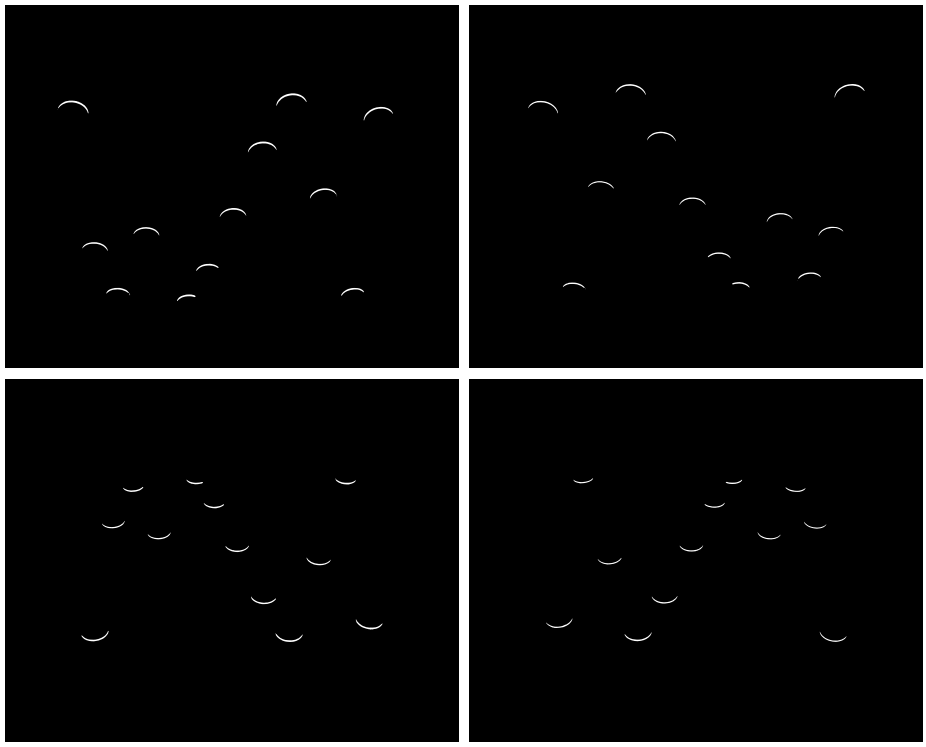

Figure 4.3: The calibration target as seen from the four cameras. In each of the images, one of the cylinders is occluded, but the remaining twelve can be seen clearly.

These algorithms require the following inputs:

- One image of the calibration target, as detailed in Section 4.2.1, from each of the cameras. Figure 4.3 shows an example set of images.
- The radius and center of every cylinder on the calibration target.
- An approximation for the angle by which the calibration pattern is rotated, as seen from each of the cameras. This is required in order to eliminate the ambiguity caused by rotational symmetry in the pattern, as outlined before.
- The set of intrinsic parameters for all the cameras.

Algorithm 4.1: *FitEllipses* calibration core.

```
1:  constants
2:      clipCount, the number of clipped line points to each side.
3:  end constants

4:  function CALIBRATEFITELLIPSES(img, intrinsic, cylCenters, angle)
5:      rawLines ← EXTRACTLINES(img)                                   ▷ Step 1.
6:      lines ← CLIPLINES(rawLines, clipCount)
7:      minLen ← 1/3 median([LINELENGTH(line) | line ∈ lines])         ▷ Step 2.
8:      usedLines ← [line ∈ lines : VALIDLINE(line, minLen)]
9:      ellipses ← [ELLIPSECENTER(FITELLIPSE(line)) | line ∈ validLines]
10:     usable ← [VALIDELLIPSE(ellipses[n], validLines[n]) | ellipses[n] ∈ ellipses]

11:     adEllipses ← ADIMENSIONALIZE(ellipses, angle)                  ▷ Step 3.
12:     adCylCenters ← ADIMENSIONALIZE(cylCenters, 0°)
13:     corr ← FINDCORRESPONDENCES(adEllipses, adCylCenters, usable)   ▷ Step 4.
14:     pose ← CALIBRATEEXTRINSIC(intrinsic, corr)                     ▷ Step 5.

15:     wEllipses ← IMAGETOWORLD(intrinsic, pose, ellipses)            ▷ Step 6.
16:     wCorr ← FINDCORRESPONDENCES(wEllipses, cylCenters, usable)
17:     corr ← [(ellipses[n], cyl) | (wEllipses[n], cyl) ∈ wCorr]
18:     pose ← CALIBRATEEXTRINSIC(intrinsic, corr)

19:     wLines ← IMAGETOWORLD(intrinsic, pose, usedLines)              ▷ Step 7.
20:     circles ← [CIRCLECENTER(FITCIRCLE(wLine)) | wLine ∈ wLines]
21:     iCircles ← WORLDTOIMAGE(intrinsic, pose, circles)
22:     corr ← [(iCircles[n], cyl) | (ellipses[n], cyl) ∈ corr]
23:     pose ← CALIBRATEEXTRINSIC(intrinsic, corr)

24:     return pose
25: end function
```

4.2.2.1 The *FitEllipses* algorithm

The first algorithm, called *FitEllipses* (depicted here as Algorithm 4.1, Algorithm 4.2 and Algorithm 4.3), generates a pose for each of the cameras as follows:

1. Laser lines are detected in the image and extracted using the algorithm described in Section 3.4.2. A predefined amount of points (*clipCount*) at the end of the lines is discarded, as extraction of the line ends is usually inaccurate.

2. Ellipses are fit to the lines using the direct least-square method.

   All lines for which this is not possible, or that are unlikely to correspond to actual calibration cylinders, are discarded. Specifically, a line is discarded for

Algorithm 4.2: *FitEllipses* calibration line and ellipse validity criteria.

```
1: constants
2:     maxExtraAngle, the maximum value for the sum of the absolute values of the extra
   angles.
3: end constants
4: function VALIDLINE(line, lengthThreshold)
5:     return LINELENGTH(line) ≥ lengthThreshold and FITELLIPSE(line) ≠ ∅
6: end function

7: function VALIDELLIPSE(ellipse, line)
8:     center ← ELLIPSECENTER(ellipse)
9:     (majorRadius, minorRadius) ← ELLIPSERADII(ellipse)
10:    startExtraAngle = ANGLELINEXAXIS(center, line[0])
11:    endExtraAngle = ANGLELINEXAXIS(center, line[last])
12:    absExtraAngle = |startExtraAngle| + |endExtraAngle|
13:    return majorRadius ≤ 2 minorRadius and absExtraAngle ≤ maxExtraAngle
14: end function
```

all purposes if and only if any of the following criteria applies:

- Its length is shorter than a third of the median length of all the lines.
- An ellipse cannot be fit to it.

Additionally, ellipses are marked as unusable for pose computation if their quality is low. These unusable ellipses cannot be discarded at this time, in order to prevent the cylinders with which they correspond from being associated with the wrong ellipse. An ellipse is marked as unusable for calibration if and only if any of the following applies:

- Its major radius is more than double its minor radius.
- The sum of the absolute values of the extra angles exceeds a predefined value (*maxExtraAngle*). The extra angles are defined as the angles between the horizontal line that goes through the center of the ellipse and each of the two lines that join the end of the laser line with the center of the ellipse. Figure 4.4 shows an example of this.

3. In order to be able to compare them to the actual cylinder coordinates, the coordinates of the ellipse centers are adimensionalized in the following way:
   3.1. The ellipse that corresponds to the centermost cylinder is found. This is done by computing the centroid of the set of ellipse centers, and then using the ellipse center that is closest to it.
   3.2. The coordinates of the center of the centermost ellipse are substracted from the coordinates of all ellipse centers.

Algorithm 4.3: *FitEllipses* calibration correspondences and adimensionalization.

```
1: constants
2:     threshold, the maximum distance between a cylinder and its associated ellipse.
3: end constants

4: function FINDCORRESPONDENCES(iCenters, wCenters, usable)
5:     closestCyls ← [CLOSESTPOINT(iCenter, wCenters) | iCenter ∈ iCenters]
6:     correspondences ← []
7:     for all wp ∈ wCenters do
8:         ip ← CLOSESTPOINT(wp, iCenters)
9:         n ← INDEXOF(ip, iCenters)
10:        if usable[n] and closestCyls[n] = wp and DISTANCE(wp, ip) < threshold then
11:            correspondences[last + 1] ← (ip, wp)
12:        end if
13:    end for
14:    return correspondences
15: end function

16: function ADIMENSIONALIZE(points, angle)
17:    (x_center, y_center) ← CLOSESTPOINT(CENTROID(points), points)
18:    centeredPoints ← [(x − x_center, y − y_center) | (x, y) ∈ points]
19:    rotatedPoints ← [ROTATEPOINT(p, angle) | p ∈ centeredPoints]
20:    xs ← [x | (x, y) ∈ rotatedPoints]
21:    ys ← [y | (x, y) ∈ rotatedPoints]
22:    adimPoints ← []
23:    for all (x, y) ∈ rotatedPoints do
24:        xScale ← |max(xs) if x > 0 else min(xs)|
25:        yScale ← |max(ys) if y > 0 else min(ys)|
26:        adimPoints[last + 1] ← (x/xScale, y/yScale)
27:    end for
28:    return adimPoints
29: end function
```

3.3. All the ellipse centers are rotated by the predefined amount described before, roughly compensating for camera rotation.

3.4. Ellipse coordinates are constrained to the range $[-1, 1]$ by applying the following rules:

- Positive X coordinates are divided by the value of the highest positive X coordinate.
- Negative X coordinates are divided by minus the value of the lowest (furthest from the center) negative X coordinate.
- Positive Y coordinates are divided by the value of the highest positive

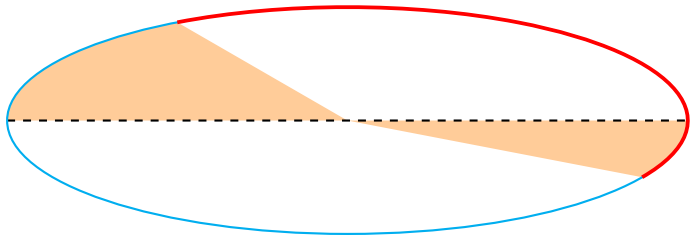

Figure 4.4: Extra angles for an example laser line. The laser line is shown in red, while the extra angles are represented by orange regions.

Y coordinate.

- Negative Y coordinates are divided by minus the value of the lowest (furthest from the center) negative Y coordinate.

Cylinder coordinates are constrained in a similar way.

4. Each ellipse is associated with the cylinder that is closest in Euclidean distance to its adimensional coordinates, unless the distance to the cylinder exceeds a predefined threshold (*threshold*), or a different ellipse is closer to the same cylinder. All the ellipses that cannot be associated with a cylinder in this way are discarded. Ellipses that were marked as unusable in Step 2 are also discarded, and the associated cylinder is not taken into account.

5. The extrinsic parameters that minimize the distance between the original ellipse centers and their corresponding cylinders are found, as described in Section 4.1.2.

6. Optionally, the results may be improved by applying the known intrinsic parameters and the extrinsic parameters that were found in Step 5 to the original ellipse centers from Step 2, in order to translate them into the world coordinate system, then repeating Step 4 and Step 5 using the translated ellipse center coordinates instead of the constrained coordinates (and the cylinder locations in the world coordinate system instead of their similarly constrained coordinates).

   This can be an improvement over the initial pose because the described cylinder association algorithm is very conservative: in order to prevent wrong correspondences from being used*, only the most obvious correspondences are chosen. When the translated coordinates are used, more correspondences are obviously correct, and therefore available.

* A single wrong correspondence would result in an extremely inaccurate pose.

As an alternative approach to cylinder and ellipse mapping [Usamentiaga et al., 2016], all the possible sets of cylinder associations can be tried, in order to choose the one that minimizes calibration error. While this would certainly be simpler than the algorithm described here, it is slower and does not scale well with the number of cylinders.

7. Also optionally, the results can be improved further by applying the pose obtained in the last step to the laser lines corresponding to the ellipses that were used, translating them to the world coordinate system, then fitting circles to them and projecting the circle centers back to image coordinates, then finally repeating Step 5 substituting the circle centers for the corresponding ellipse centers.

4.2.2.2 The *Iterate* algorithm

The second algorithm, called *Iterate* (depicted here as Algorithm 4.4), requires an initial pose in order to function, and uses it to generate more accurate poses for each of the cameras as follows:

1. A model of the calibration cylinders is built from arc primitives, using the known location and radius information.

2. Laser lines are extracted from the image as in Step 1 of the *FitEllipses* algorithm.

3. The initial pose is applied to the laser lines, translating them to the world coordinate system.

4. Correspondences between the points that constitute the translated laser lines and the model are found using an R-tree structure, as described in Section 3.5.2.

5. The original laser line points are substituted for the translated laser line points in the list of correspondences.

6. The extrinsic parameters that minimize the distance between the original laser line points and their corresponding line points are found, as described in Section 4.1.2.

7. Step 3 and all the steps after it are repeated, replacing the initial pose with the pose computed in Step 6. This is repeated until predefined convergence criteria (maximum number of iterations (*maxIterations*), minimum difference between successive poses (*minDiff*)) are reached.

Algorithm 4.4: *Iterate* calibration algorithm.

constants
    maxIterations, the maximum number of iterations.
    minDiff, the minimum difference between the new pose and the preceding one.
end constants

function CALIBRATEITERATE(*img*, *intrinsic*, *cyls*, *pose*)
    prims ← [ ] ▷ Step 1
    for all *cyl* ∈ *cyls* **do**
        prims[**last**] ← ARC(CIRCLECENTER(*cyl*), CIRCLERADIUS(*cyl*), 0°, 180°)
        prims[**last**] ← ARC(CIRCLECENTER(*cyl*), CIRCLERADIUS(*cyl*), 180°, 360°)
    end for
    rtree ← RTBUILD(*prims*)
    lines ← CLIPLINES(EXTRACTLINES(*img*), *clipCount*) ▷ Step 2.
    points ← [*p* ∈ [*l* ∈ *wLines*]]
    iterations ← 0
    do
        wPoints ← IMAGETOWORLD(*intrinsic*, *pose*, *points*) ▷ Step 3.
        corr ← [(*p*, *q*) | *p* ∈ *wPoints* : *q* = RTNEAREST(*rtree*, *p*) ≠ ∅] ▷ Step 4
        iCorr ← [(*points*[*i*], *q*) | (*wPoints*[*i*], *q*) ∈ *corr*] ▷ Step 5
        oldPose ← *pose*
        pose ← CALIBRATEEXTRINSIC(*intrinsic*, *iCorr*) ▷ Step 6
        diff ← $\sum$[|*pose*[*i*] − *oldPose*[*i*]| | *i* ∈ 0..|*pose*|]
    while *iterations*++ < *maxIterations* **and** *diff* ≥ *minDiff* ▷ Step 7.
    return *pose*
end function

4.2.2.3 Combining the algorithms

Due to the significantly higher number of correspondences it uses, *Iterate* generates more accurate poses than *FitEllipses*.

For this reason, both are employed to calibrate the proposed system. First, *FitEllipses* is applied to compute an initial pose, then *Iterate* refines this pose and generates the final pose that will be used in the system.

Step 6 and Step 7 of *FitEllipses* are not needed in this case, as high accuracy of the initial pose is not required.

4.2.3 Estimating the calibration error

The calibration error can be estimated in multiple ways, by applying the computed pose to the laser lines and comparing the result with the known cylinder pattern. Some of these ways (depicted in Figure 4.5) are:

- **Radius error:** After fitting circles to the translated laser lines, the radius error

(err_r) is computed as the absolute difference between the radius of each of the circles (r_{fit}) and the real radius of the corresponding cylinder (r_{model}). For each laser line:

$$err_r = |r_{fit} - r_{model}|$$

Failure to correctly fit circles to the translated lines may cause high error values to be returned, even if ellipses could be fit to the original lines. Therefore, this estimation cannot be relied upon by itself.

- **Center error:** After fitting circles to the translated laser lines, the center error (err_c) is computed as the Euclidean distance between the center of each of the circles (c_{fit}) and the center of the corresponding cylinder (c_{model}). For each laser line:

$$err_c = \sqrt{(c_{fit_x} - c_{model_x})^2 + (c_{fit_y} - c_{model_y})^2}$$

  This estimation presents the same problem as the radius error estimation, as detailed before.

- **Point error:** For each cylinder, the point error (err_p) is computed as the average absolute difference between the real radius of the cylinder and the Euclidean distance between each point (p) of the corresponding laser line (ps) and the real cylinder center. For each laser line:

$$err_p = \sum_{d \in ds} \frac{|d - r_{model}|}{|ds|}$$

  where $ds = \{\sqrt{(p_x - c_{model_x})^2 + (p_y - c_{model_y})^2} \mid p \in ps\}$.

- **Global point error:** The global point error (err_{gp}) is computed, for each of the laser line points, as the difference between the radius of the nearest real cylinder and the Euclidean distance between the line point and the center of said cylinder.

  Therefore, it is obtained in the same way as point error, except that ps is replaced with the set of all points in all laser lines, and the radius of the nearest cylinder to each given point is used instead of r_{model}.

  Global point error takes into account all laser lines, including those that were discarded because ellipses could not be fit to them. Although this may include lines that were not actually part of the calibration target, most such lines are actually partially occluded cylinders that would not be taken into account in any other way.

The radius, center and point error estimations operate on circles, whereas the global point error estimation operates on individual points. For this reason, all these

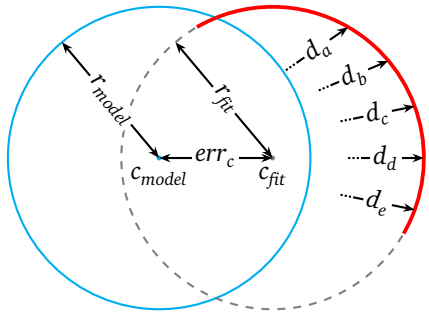

Figure 4.5: Calibration error estimations. The blue circle represents the model cylinder, whereas the gray dashed one represents the result of fitting a circle to the translated laser line. The distance between the circles has been exaggerated for illustration purposes. Here, $ds = \{d_a,\ d_b,\ d_c,\ d_d,\ d_e,\ \ldots\}$.

estimations can be used with the results of the *FitEllipses* algorithm, which also operates on circles.

Technically, all of them can also be applied to the result of the *Iterate* calibration algorithm, although the first three require each laser line to be associated with a specific cylinder, and the first two of them additionally require circles to be fit to the translated lines.

As the radius, center and point error estimations are not readily available for *Iterate*, and they also present the shortcomings detailed before, whereas global point error is robust and does not require correspondences to be computed, the latter is chosen as the main error estimation to be used when comparing poses.

4.2.4 Testing the calibration

In order to both ensure the quality of the computed poses and determine when the system should be re-calibrated, a simple testing procedure has been devised.

This testing procedure complements the general calibration error computations described in Section 4.2.3 by approximating the measurement error for different dimensions. The testing procedure is carried out in the following way:

1. Four images of the calibration target are captured*, one from each of the cameras.

2. Laser lines are extracted from the images, translated into world coordinates and combined into a single point cloud.

* Preferably, these should not be the same images that were used for extrinsic calibration.

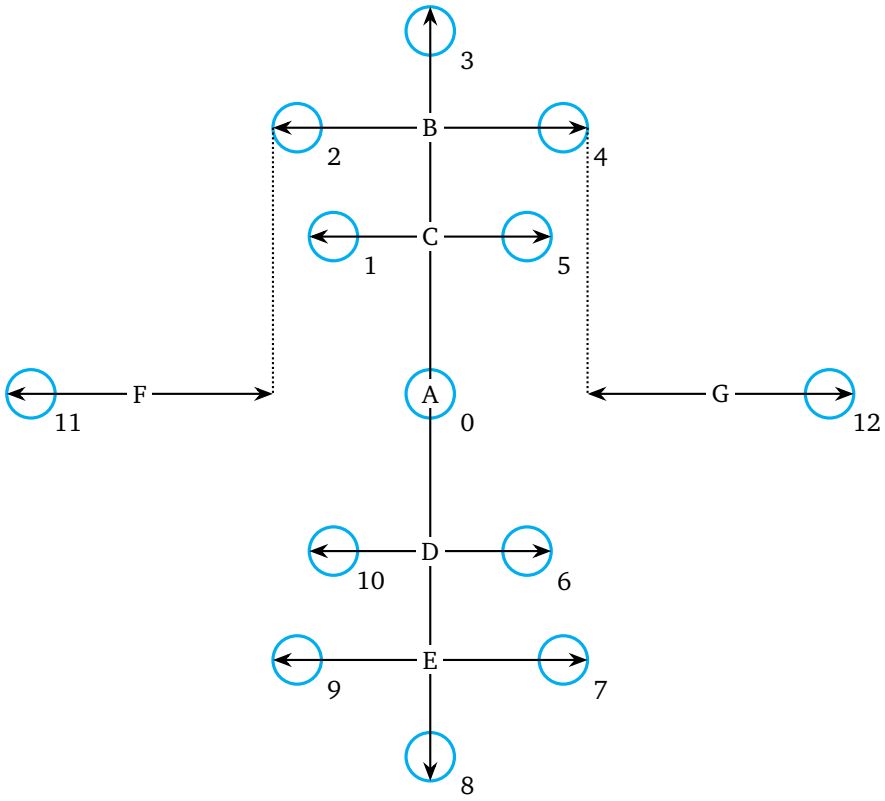

Figure 4.6: Distances considered when testing the camera poses. Lines A to G attempt to simulate rail height, head width, upper web thickness, lower web thickness, foot width, left asymmetry and right asymmetry, respectively.

3. The points that make up the line are associated with their corresponding calibration target cylinder.
4. A circle is fit to the set of points associated with each of the cylinders.
5. The distances between circle ends are measured and compared with the known distances between the corresponding cylinders, and the computed error is reported to the user.

Not all these distances can be displayed to the user, however, as the result would be unwieldy*. Therefore, only a small number of distances is shown: those that are

* The number of possible distances between cylinders is $\binom{13}{2} = \frac{13!}{2!11!} = 78$.

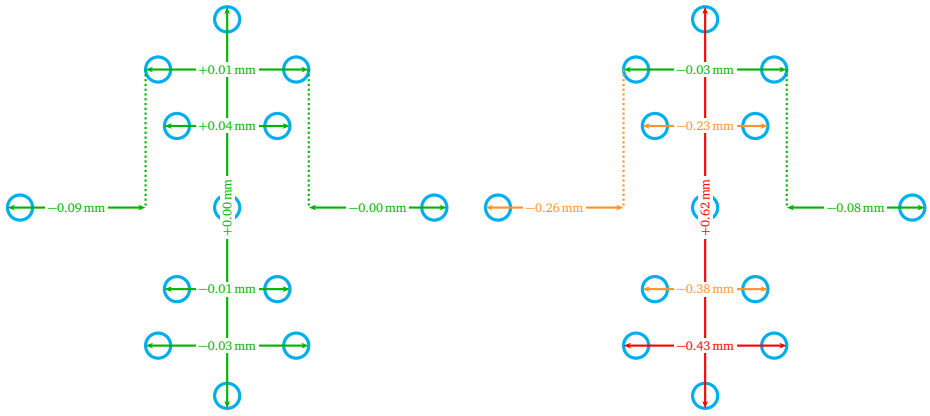

Figure 4.7: Two examples of calibration test results. The differences between measured and real distances are shown for a test taken immediately after calibration (left) and for a test taken two weeks afterwards (right).

considered to be representative of actual measurement error, based on analysis of the calibration target and the rail profiles. Specifically, the distances between cylinders that are located in regions of the measurement plane where relevant rail dimensions are computed during measurement.

Figure 4.6 shows the calibration target cylinders, and the distances that are considered when testing the camera poses. Figure 4.7 shows examples of calibration test results.

In order to define which distances are relevant, a simple description language was designed, where each distance is described as follows:

```
<label>: <starting cylinder>-<end cylinder>;
```

Here, '`<label>`' is a label for the distance and '`<starting cylinder>`' and '`<end cylinder>`' are the cylinders between which the distance is computed, as shown in the figure. By default, the distance between the circle points furthest from each other is considered. However, prepending '~' to a cylinder number causes the circle point closest to the other circle to be used instead. Adding 'x' or 'y' between the second cylinder number and the semicolon causes the horizontal or vertical distance to be reported, respectively, whereas Euclidean distance is reported by default.

In this language, the distances shown in Figure 4.6 and in Figure 4.7 can be described as follows:

```
Height: 3-8;
Head width: 2-4;
Upper web thickness: 1-5;
Lower web thickness: 10-6;
Foot width: 9-7;
Left asymmetry: 11-~2x;
Right asymmetry: 12-~4x;
```

4.3 Experiments on calibration procedures

The quality of the proposed calibration procedures, and their influence on the measurements made by the proposed system, must be studied experimentally in order to be able to determine the optimal settings for them.

4.3.1 Intrinsic calibration

In order to study the effect of multiple intrinsic calibration algorithms and patterns on the accuracy of the resulting measurements, an experiment is performed in the following way:

1. Images of five calibration targets are taken in different positions within the camera field of view and at different angles. The patterns depicted on the calibration targets are similar to the ones shown in Figure 4.1:
   - **WhiteCircle:** MVTec HALCON 11 circle pattern with a matte white background (similar to the top left pattern).
   - **ReflectiveCircle:** MVTec HALCON 11 circle pattern with a reflective background (otherwise similar to the top left pattern).
   - **SmallCircle:** MVTec HALCON 12 small circle pattern (similar to the top right pattern).
   - **Concentric:** Concentric ring pattern for use with certain algorithms described in the literature [Vo et al., 2011] (similar to the bottom left pattern).
   - **MATLAB:** Checkerboard pattern for the MATLAB Computer Vision System Toolbox (similar to the bottom right pattern).
2. Intrinsic parameters are generated for each of the image sets.
3. Five additional images of an 8×7 checkerboard pattern are taken.

4. Each of the checkerboard pattern images is used to generate a pose using every set of intrinsic parameters.

5. Each pose is applied to the checkerboard pattern images, other than the one that was used to generate it, along with the corresponding set of intrinsic parameters.

6. The absolute differences between square sides in the translated images and known square sides are averaged for each pose and pattern image, as an estimation of the calibration error.

As an alternative, reprojection error could have been used. **Reprojection error** is the distance between the point in the image and the reprojected point, that is: the result of translating the point to world coordinates and then back to image coordinates. It is often used as a measure of intrinsic calibration accuracy.

	WhiteCircle	*ReflectiveCircle*	*SmallCircle*	*Concentric*	*MATLAB*
Best	0.026	0.024	0.020	0.025	0.021
Median	0.045	0.039	0.031	0.035	0.036
Worst	0.072	0.058	0.046	0.055	0.085
Average	0.047	0.040	0.033	0.037	0.039
Std. dev.	0.014	0.008	0.008	0.009	0.018

Table 4.1: Comparison among sets of intrinsic parameters. Calibration error is given in mm. Each set of intrinsic parameters is labeled with the name of the calibration pattern that was used to generate it.

Table 4.1 compares the calibration error for each set of intrinsic parameters. It can be seen that calibration error is very similar even with different algorithms and patterns. Although specific intrinsic calibration algorithms might make more of a difference for other cameras that produce more distorted images, the differences are negligible in the case of the cameras used in the laboratory environment, for the purposes of this work.

4.3.2 Extrinsic calibration in a laboratory environment

In order to evaluate the extrinsic calibration procedures for the proposed system, as described in Section 4.2.2, they are compared with a well-known sheet-of-light calibration procedure in a laboratory environment*.

* This laboratory environment comprises four Genie Teledyne Dalsa HM1400 cameras, with Schneider-Kreuznach APO-XENOPLAN 2.0/24-0005 lenses and Coherent laser filters (635 CW - 20 BP for two of the cameras and 685 CW - 20 BP for the other two); as well as two Coherent StingRay-640 (640 nm) and two Coherent Stingray-685 (685 nm) laser emitters.

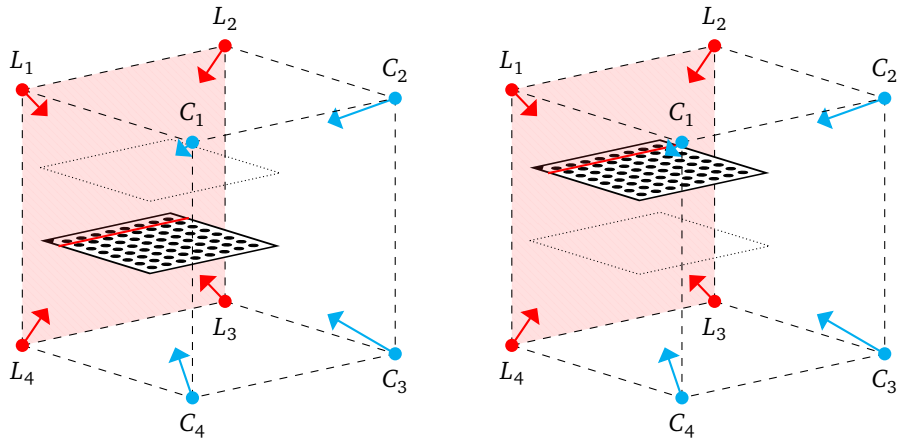

Figure 4.8: Sheet of light calibration with the MVTec HALCON circle pattern. The pattern is shown at two different heights. The red lines represent the intersection between the measurement plane and the pattern.

This well-known calibration procedure [MVTec Software GmbH, 2016, Section 6.3.1], depicted in Figure 4.8, entails capturing images of the calibration plates, such as those shown in Figure 4.1, roughly perpendicular to the measurement plane at two or more different heights. The poses of these calibration plates are computed as described before, the laser lines projected on them are extracted and translated into world coordinates relative to one of the plates and a plane is finally fit to these world coordinates. For the purposes of this work, this will be called the *SheetOfLight* algorithm.

Images for the experiment are captured in the following way:

1. The system is set up as depicted in Figure 4.2. The emitters are aligned so that their laser planes are near-coplanar to the naked eye.

2. Three sets of images of the calibration target are taken: with the target centered between the laser emitters and with the target displaced roughly 3 cm to the left and right.

3. The calibration target is removed from the system, and a circle pattern for MVTec HALCON is placed perpendicular to the measurement plane.

4. Images of the calibration target are taken from the upper cameras (1 and 2). A first set of images is captured with the laser emitters turned off and a high exposure time (between 400 ms and 1000 ms, depending on the camera, as they employ different wavelength filters), so that the pattern is clearly visible;

and three other sets are captured with the laser emitters turned on and three different low exposure times (specifically, 5 ms, 10 ms and 15 ms), so that the laser line can be seen. All parts of the laser line that appear in regions of the image other than the plate are manually removed with an image editor for the purposes of the experiment.

5. The plate with the circle pattern is elevated roughly 3.5 cm and 8 cm from its initial location, and Step 4 is carried out again for both new positions.

6. The plate is removed and a gray polyethylene test piece, similar to the one shown in Figure 4.9, is placed under the laser emitters.

7. Images of the laser lines projected on the test piece are taken from the upper cameras with different exposure times (2.5 ms to 15 ms, in increments of 2.5 ms). Again, all parts of the laser lines other than the upper region are manually removed with an image editor.

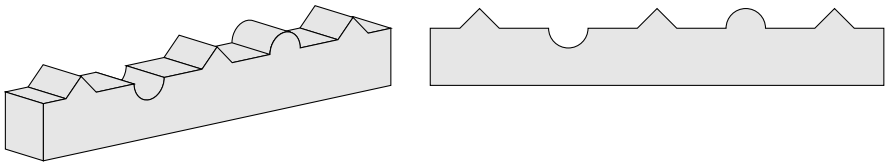

Figure 4.9: Calibration test piece. A perspective view of the piece (left) is depicted, as well as its cross-section (right).

After these images have been captured, poses are generated as follows:

- The *FitEllipses* and *Iterate* calibration algorithms are applied to the calibration target images. In the case of *FitEllipses*, both the initial pose and the poses that result from Step 6 and Step 7 are considered, called *FitEllipses1*, *FitEllipses2* and *FitEllipses3*, respectively. The pose that results from *FitEllipses3* is used as the initial pose for *Iterate*. As three sets of calibration target images are used, 3 poses result for each camera and algorithm or algorithm stage.

  In this experiment, two points are clipped from each end of the laser lines when calibrating with *FitEllipses*, and 10 points are clipped in the case of *Iterate*. The influence of different numbers of clipped points on calibration results will be explored in Section 4.3.3.

- The well-known *SheetOfLight* algorithm is applied to the plate and laser images. As three sets of plate images are used (each at a different level), three combinations of two images are possible. For each plate image, nine laser line images with different exposure times exist. In total, nine poses are generated for each camera using this algorithm.

For each of the six sets of two test piece images (one image from each of the two upper cameras):

1. Lines are extracted as described in Section 3.4.2. 20 points are clipped from each line end.
2. Each of the 21 sets of poses is applied to the extracted lines.
3. The translated lines for the two cameras are combined into a single test piece profile for each set of poses.
4. Each profile is aligned with the upper region of the model shown in Figure 4.9.
5. For each profile, average distances from each point to the model are computed.

The best, median and worst of these average distances are computed for each of the employed algorithms. The results are depicted in Table 4.2.

	FitEllipses1	*FitEllipses2*	*FitEllipses3*	*Iterate*	*SheetOfLight*
Best	0.052	0.049	0.047	0.041	0.037
Median	0.079	0.077	0.096	0.052	0.055
Worst	0.131	0.084	0.102	0.070	0.087
Average	0.087	0.071	0.082	0.056	0.064
Std. dev.	0.030	0.013	0.023	0.010	0.017

Table 4.2: Comparison between a well-known algorithm and the calibration algorithms for the proposed system. Best, median and worst profile errors (each computed as the average distance between profile points and the model) are shown (in mm) for each of the considered calibration algorithms.

It can be seen that the results of *FitEllipses2* are a small improvement over those of *FitEllipses1*, while the performance of *FitEllipses3* appears to be worse. *Iterate* supposes an important improvement over all of them, and its results are similar to, and slightly better than, those of *SheetOfLight*. As such, and taking into account that using *SheetOfLight* in the production environment is not feasible, it is clear that, among the available algorithms, *Iterate* should be used in the proposed system.

4.3.3 Extrinsic calibration in a production environment

The extrinsic calibration procedures for the proposed system are also evaluated by studying their performance on a set of real calibration plate images captured in the production environment*.

* This production environment comprises four Genie Teledyne Dalsa TS-M2560 cameras, with Goyo 16mm HR 1" F1.4 C lenses and MidOpt laser filters (Bi632 for two of the cameras and BP695 for the

Clipped points	*Failure* (%)	*Error* ≥ 1 mm (%)
0	0.00	0.48
1	12.14	0.48
2	0.00	0.24
3	0.00	0.24
4	0.00	0.71
5	0.71	0.95
6	2.14	1.19
7	2.86	0.95
8	3.57	1.67
9	7.14	1.67
10	11.43	0.95
11	17.14	0.24
12	24.29	0.24

Table 4.3: Percentages of invalid calibrations for the *FitEllipses* algorithm. Both the percentage of images for which the algorithm failed (left) and the percentage of images for which the global point error was 1 mm or greater (right) are shown, grouped by the number of clipped points to each end of the laser lines.

To this end, calibration plate images that were taken in a roughly two-year period (between September 2015 and October 2017) are retrieved. When multiple sets of calibration images (one for each of the four cameras) were taken on the same day, or on contiguous days, only the last such set is considered*. After applying this criterion, 35 sets of images can be used.

It must be noted that cases where the calibration algorithms failed to return a valid result, or where the operators decided to discard the result without saving, are not considered here. Therefore, this experiment is not suitable for the goal of studying the reliability of the algorithms in conditions other than optimal. Only their best-case error values are studied in this way.

For each set of calibration images, and for each possible number of clipped line points from 0 to 12, poses for all three *FitEllipses* stages (*FitEllipses1*, *FitEllipses2* and *FitEllipses3*, as described in Section 4.3.2) are computed, when possible.

The *Iterate* algorithm is also applied to every set of calibration images, using the *FitEllipses3* pose with two clipped points** for a given set as an initial pose. *Iterate* is

other two); as well as two Coherent StingRay-640 (640 nm) and two Coherent Stingray-685 (685 nm) laser emitters.

* As they were so quickly replaced, the other sets are assumed to be 'bad' in some way, and thus unusable for this purpose.

** Clipping two points from the line ends appears to be optimal or close to optimal when applying

FitEllipses clipped points	*FitEllipses1+Iterate*	*FitEllipses2+Iterate*	*FitEllipses3+Iterate*
0	0.052	0.052	0.052
1	0.052	0.052	0.052
2	0.052	0.052	0.052
3	0.052	0.052	0.052
4	0.052	0.052	0.052
5	0.052	0.052	0.052
6	0.052	0.052	0.052
7	0.052	0.052	0.052
8	0.052	0.052	0.052
9	0.051	0.051	0.051
10	0.051	0.051	0.051
11	0.051	0.051	0.051
12	0.051	0.051	0.051

Table 4.4: Comparison between the results of the *Iterate* algorithm using different initial poses. Global point error (in mm) is shown for the results of *Iterate* with 10 clipped points, using initial poses originating from different *FitEllipses* stages with different numbers of clipped points.

also applied for all possible numbers of clipped line points from 0 to 12.

Error is measured as described in Section 4.2.3. The original (unclipped) laser lines are used in the case of the global point error, while the other error estimations use the same (clipped) lines as the calibration algorithms.

Apart from the radius error, which is clearly a poor error estimation, all the other estimations show the error values for *FitEllipses2* as being slightly lower than *FitEllipses1*, and those of *FitEllipses3* as being consistently better than both of them, despite the opposite being true for the laboratory environment, as seen in Section 4.3.2. Figure 4.13 also shows that the performance of *Iterate* is significantly better than those of the *FitEllipses* stages.

As for the number of clipped points, in the case of *FitEllipses* the optimal value appears to be two, when considering center and point error. For the global point error, the optimal value seems to be zero, according to the graph. However, taking into account the facts that no points were clipped when computing it for this experiment, and that the same images were used for calibration and for error estimation (as only one set of images is available for each calibration), it follows that the slightly lower error values when no clipped points were used for calibration might be caused by

FitEllipses to these images. *FitEllipses3* also appears to have better performance than the other *FitEllipses* stages on this set of images. All of this will be explained below.

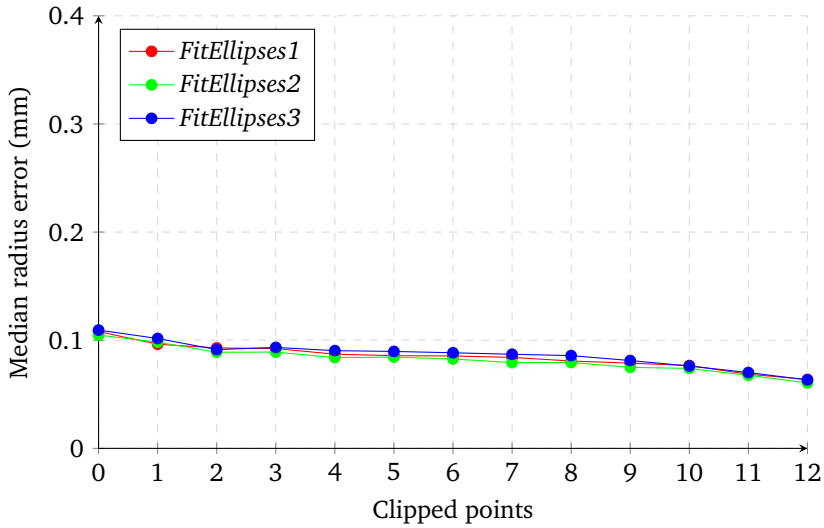

Figure 4.10: Median radius error for different numbers of clipped points.

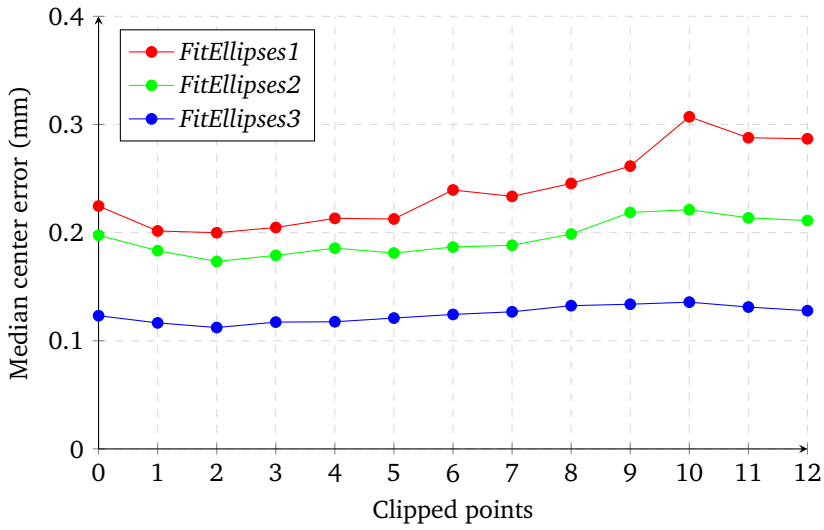

Figure 4.11: Median center error for different numbers of clipped points.

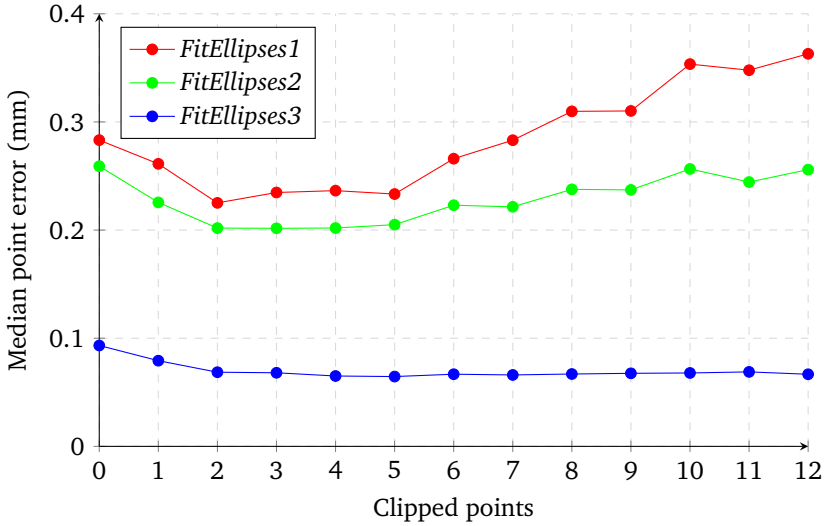

Figure 4.12: Median point error for different numbers of clipped points.

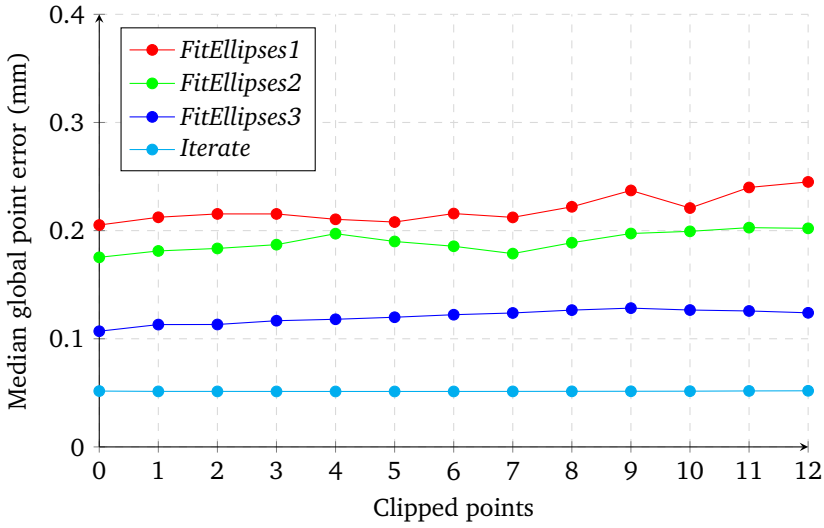

Figure 4.13: Median global point error for different numbers of clipped points.

overfitting of the poses. Therefore, a value of two is finally chosen* for *FitEllipses*.

For *Iterate*, no value appears clearly better for these sets of images. However, lower quality images may benefit from potentially noisy laser line ends being clipped. As this does not appear to affect pose quality in any significant way, a value of 10 clipped points for each line end is chosen.

Table 4.3 shows the percentage of invalid calibrations depending on the number of clipped points for the *FitEllipses* algorithm**. It must be noted that two points were clipped from each end in the production environment, and calibration failures were discarded, so the failure percentage for two clipped points could not be anything other than 0 %. However, it can also be seen that no value improves the percentage of calibrations with error higher than 1 mm over that seen in the case of two clipped points, which reinforces the above conclusions.

Additionally, Table 4.4 shows error values for *Iterate* (with 10 clipped points) depending on the initial pose. It can be seen that the quality of the original poses is largely (but not completely) irrelevant when employing the *Iterate* algorithm.

* For this camera and image resolution. Different image resolutions may require different numbers of clipped points.

** The *Iterate* algorithm succeeded for all the images for which *FitEllipses* did.

Chapter 5

Autonomic computing

5.1 Background

Autonomic computing refers to the self-management of the resources of a computing system [Huebscher and McCann, 2008]. Self-management implies:

- Self-configuration of the system for a certain usage or a specific platform or user.
- Self-optimization of the system and its use of resources, both reactively and proactively.
- Self-healing of the system, which includes problem detection, diagnostic and, when possible, repair.
- Self-protection of the system against security threats, both malicious and not.

In order to achieve this, policies must be defined that describe the desired system conditions. These policies determine how the effectors of the system should be used in order to correct its functioning. Such decisions must be based on the properties of the system, which are measured by a series of sensors. The basic structure of an autonomic system can be seen in Figure 5.1.

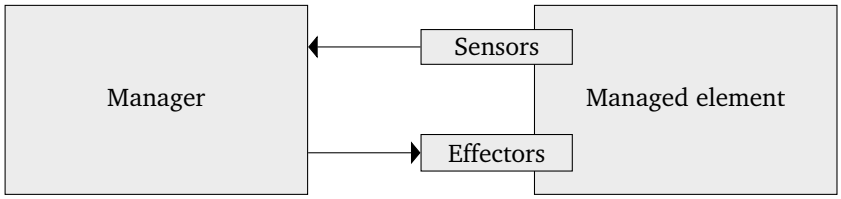

Figure 5.1: Basic structure of an autonomic system.

Thus, a managed element (or a set or managed elements) exposes sensors and effectors to an autonomic manager, which implements the autonomic features. The

system conformed by the managed element (or elements) and the manager is called an autonomic system.

In an ideal implementation of autonomic principles, an autonomic manager is supposed to implement the following loop, usually called the MAPE-K loop [Huebscher and McCann, 2008]:

1. **Monitor** the managed element (or elements), collect and store sensor data.
2. **Analyze** the stored data, identify problems and opportunities for improvement.
3. **Plan** an action or sequence of actions for the managed element to take, using a repository of **knowledge** which stores models for system behavior. This knowledge may have been acquired by the system from previous analysis, or it may have been set by the user or programmer.
4. **Execute** the plan.

Depending on the position of the manager relative to the managed element, there are three possibilities:

1. the manager may be presented as a component of the managed system, designed to be a part of the system and integrated with it;
2. it may be an independent system, as a sort of middleware;
3. or a mixed approach may be chosen, where every managed element has its own manager, but this manager reports to, and is managed by, an overarching manager, forming a hierarchy of autonomic managers [IBM, 2005]. In this approach, the local manager acts as a 'local loop' which only takes into account the environment of its managed element, while the overarching manager, or 'global loop', has knowledge on the whole system and supervises its overall configuration, performance and security [Parashar and Hariri, 2005].

Approach 2 allows for the implementation of autonomic features in existing systems [Khalid et al., 2009], although it still requires the managed system to expose its sensors and effectors to the manager. Approaches 2 and 3 are also especially suitable when managing distributed systems, where a single manager must supervise multiple managed elements.

The expression 'autonomic computing' was coined by IBM in 2001 [Horn, 2001]. The term 'autonomic' was inspired by the autonomic nervous system, which manages unconscious bodily functions in animals. This was not the first time biological metaphors were employed to describe self-management features in computer systems: in the 1990s, a distributed system (also by IBM) that analyzed malware samples (and which incidentally implemented self-protection features that predate autonomic computing) was termed an 'immune system' [White et al., 1999].

A long-term goal of autonomic computing as a research field is the development of systems able to seek high-level objectives, and even negotiate and enforce agreements with other autonomic systems in order to achieve those objectives [Nami and Bertels, 2007].

While autonomic computing frameworks are best suited for the management of computer networks and distributed systems, with a large amount of hardware and software elements to manage, autonomic computing may also apply to a single computer system or a single piece of software.

Depending on the degree of application of the principles of autonomic computing, and on the complexity of the system, different types of policies may be implemented [Kephart, 2005]. These policies may:

- describe the actions that must be taken when certain conditions are met (rule-based policies);
- just describe the values some properties should have and let the system decide on specific actions (goal-based policies);
- or specify an utility function that must be maximized or minimized (utility function-based policies).

As such, autonomic computing techniques range from the comparison between the measured values, according to criteria selected by the programmer or user (in which case the manager can be implemented as a rule engine or an expert system of sorts [Agrawal et al., 2005]) to the use of machine learning techniques to analyze the measured values during the entire life of the system and apply the results in order to reach high-level goals [Kephart and Chess, 2003].

Such advanced techniques include neural networks, probabilistic models and combinatorial search [Menascé and Bennani, 2003], depending on the specific self-management feature that is to be implemented.

Apart from classifying autonomic systems according to their architecture and their degree of compliance with IBM's autonomic ideals, a quantitative evaluation of autonomic capabilities may be performed. Self-healing and self-protection capabilities may be measured by fault injection [Lau and Shum, 2009], while self-optimization capabilities may be checked by subjecting the system to a quickly varying workload under different conditions.

Implementation of autonomic features in a machine vision system entails analyzing the captured images in order to determine the state of the cameras (or other capture devices), of the light sources and of other elements that may affect the system. The resulting information may then be used to detect problems, and to optimize the settings of the machine vision software (also the cameras and other elements, to the extent that they may expose their configuration to the software).

The managed vision system may expose a set of image properties, so that only those properties may be taken into account for self-management purposes, or it may expose the raw images. In the latter case, the manager would operate as a secondary machine vision system, and consider all the elements that may appear in the images.

Figure 5.2 shows the basic structure of a machine vision system.

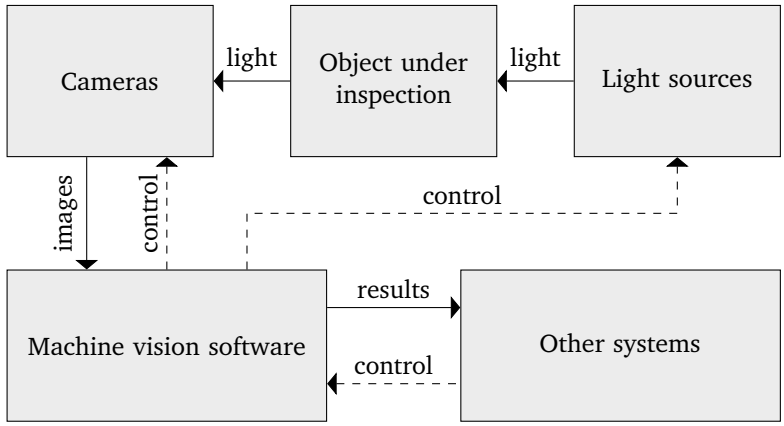

Figure 5.2: Basic structure of a machine vision system.

5.2 Designing autonomic computing functionality

In the case of the system described in this work, autonomic features may be used in order to detect issues related to the working environment and the operating conditions, and more specifically those related to:

- The laser emitters and the camera lenses.
- The position of the cameras.
- Other system components.

Whenever working conditions indicate that the inspection system is about to provide incorrect measurements, human operators should be warned, and advised to perform specific manual tasks such as replacing or cleaning the laser emitters.

5.2.1 Baselines

The output of sensors based on the laser lines may be hard to correlate with the system health, barring extreme values that would indicate a complete or near complete

inability to operate. For this reason, there is no way for the system to determine the acceptable range for every sensor by itself. This means that feedback must be provided by human operators. After maintenance is performed, the system should be in a good working condition. If the fact that maintenance has been performed is known, then the sensor values at the time (or rather, the averaged values for a given period of time after maintenance) may be used as a baseline. After the baseline is known, an acceptable value range may be computed. As an example, Figure 5.3 shows laser intensity (described later, in Section 5.3.1) for all cameras over three months. Reference lines are shown, marking the average laser intensity of the first 250 rails for every camera.

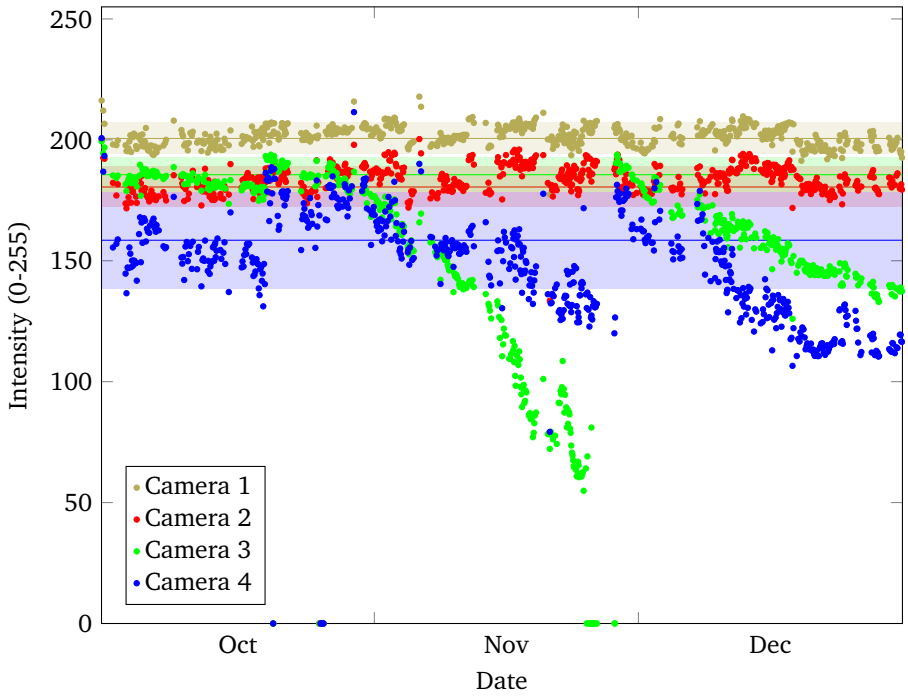

Figure 5.3: Laser intensity for all cameras, with baselines, for rails measured from October 2017 to December 2017.

This way, the system can be provided with updated health criteria every time a maintenance procedure is carried out, while minimal operator oversight is required.

5.2.2 Output

Different strategies may be employed to make human operators aware of the working conditions at a given time. These strategies include:

Alarm events which may be triggered in order to warn the operator about serious system problems that require immediate action. These include:

- Any individual sensor being out of range to such an extent that current measurement results may have no value.
- The complete inability to measure rails for any other reason, such as camera trigger failure.

System reports which may be issued periodically. These reports may include:

- Recent sensor values: average and other statistics.
- Overall system health, derived from said values.
- Number of alarms and other events during the period covered by the report.
- Estimated time until sensor values get out of range.

System reports are intended to provide a quick summary in order for operators to check that the system is healthy, and allow them to gauge the need for maintenance action.

A **sensor dashboard** would be displayed on demand, and show:

- All the information that would appear in a system report.
- Current and historical sensor data.
- All the recorded alarms and other events, including maintenance events.
- All the generated system reports.

The sensor dashboard is intended to expose all the available sensor information in order to allow operators to investigate system failures or predict future issues, and for any other purpose.

5.3 Relevant features

In order to implement self-assessment functionality, the first step is to determine one or more features that may serve as indicators for system health. It can be seen that, to estimate the health of the triangulation units, relevant features must be derived

from the resulting images. However, to give an estimate of the health of the overall system, measurement results must also be taken into account. Figure 5.4 shows a simplified view of the system, where sensors to compute such features take either the images or the measurement results as inputs.

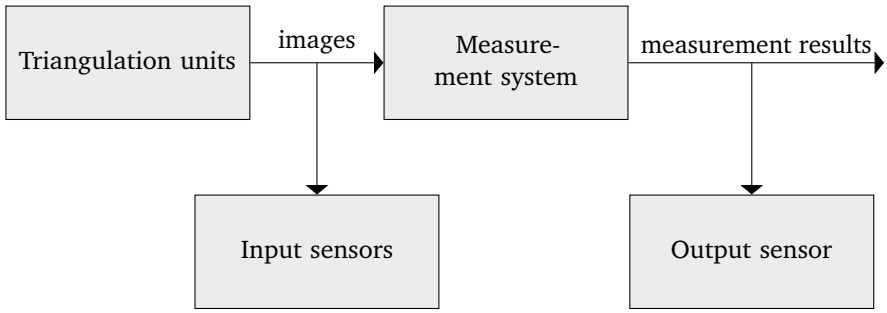

Figure 5.4: Role of autonomic sensors with respect to other components.

The following features are proposed as indicators of system health. For most features, a definition is given, along with an algorithm for computing its value in order to design and implement a sensor. The results of applying such sensors to a set of preexisting rail images are also supplied. In order to obtain these results, the raw images were used, unless stated otherwise. When the laser lines were needed, they were extracted in the same way as they are when normally operating the proposed system during manufacturing.

5.3.1 Laser intensity

Laser intensity indicates the average gray value of the points in the laser line. As the images are 8-bit grayscale, laser intensity is a value between 0 and 255. Both too low and too high laser intensity values may be undesirable. Low laser intensities may make it harder to detect a laser line, whereas high laser intensities may saturate the sensor, and thus reduce the accuracy of subpixel line extraction.

Laser intensity is computed as follows: The normal of the line at all the extracted points is computed. All the image pixels that are crossed by any of the normal lines and which are not further than $\frac{linewidth}{2}$ from the intersection between the normal and the extracted line are considered to be laser line pixels. Figure 5.5 shows blue crosses on the pixels that are considered to laser line pixels for this purpose. The average gray value of the pixels in the laser line, for all the lines in all the images of the same rail, is computed for every camera.

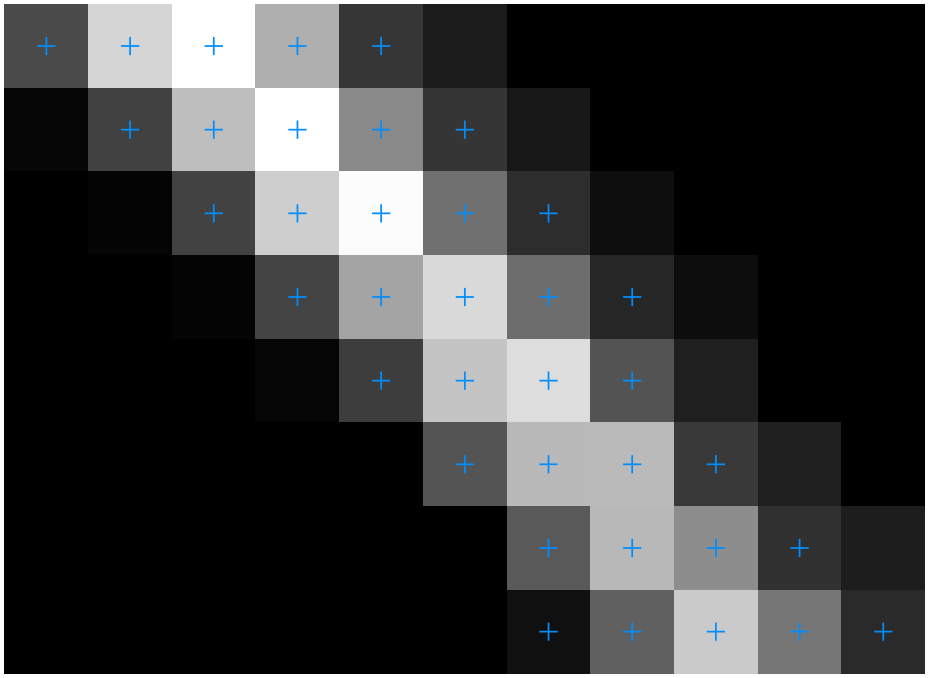

Figure 5.5: Pixels in the line

5.3.1.1 Results

Figure 5.6 shows the average laser intensity for the first rail that was measured every day from January 1, 2017 to December 31, 2017.

Vertical lines denote that an event took place at the time, as follows:

- Red lines: Production stopped for at least 48 hours. Cleaning of the cameras and laser emitters and other maintenance activities may or may not have happened during this period.
- Green lines: The system was calibrated.

Additionally, the laser emitters were replaced in June. As the exposure times were then changed accordingly, this is marked with a cyan line.

Laser intensity can be seen not to degrade at all for the upper cameras. The only change (both in the average intensity, which increased slightly, and in the deviation, which decreased) that can be identified occurs in June, when the laser emitters were replaced and exposure time was adjusted accordingly. However, for the lower cameras the intensity degrades continuously, and it only improves when the

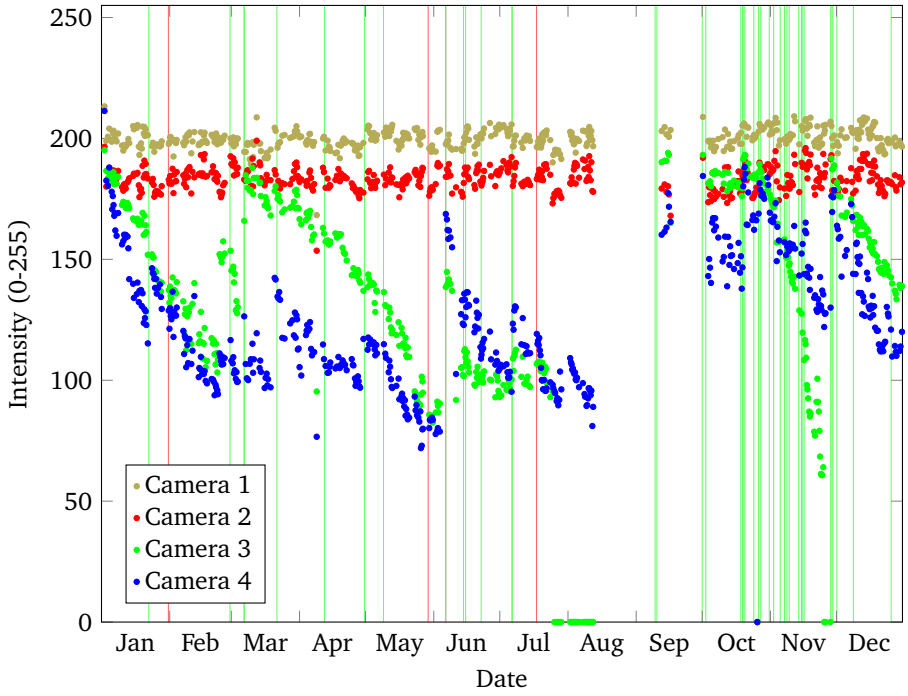

Figure 5.6: Laser intensity for all cameras, for rails measured from January 2017 to December 2017.

lenses are cleaned, the laser emitters are replaced or the exposure time is increased. Therefore, it appears that laser line degradation in this system is mostly caused by dirt accumulated on the lenses, which would naturally affect the lower cameras, which point upwards, more strongly than the upper cameras, which point downwards.

Figure 5.7 depicts the behavior of the laser line intensity for one of the cameras (triangulation unit L_4) during a single month (May 2017). The plot is jagged, with sudden, very steep variations of up to about 10 intensity units, which contrast with a slower downward trend. Although they are not pictured here, plots for the other cameras (especially triangulation unit L_3) are similarly jagged. These upward slopes came as a surprise: intensity was expected to degrade almost monotonically, and seldom increase except for maintenance events.

One possible conjecture is that the laser intensity follows a daily cycle, even though lighting conditions are controlled inside the facility where the system is located, and there is no access to natural light. For all the rails that were measured during May 2017, Figure 5.8 depicts the difference between their average laser line intensity and

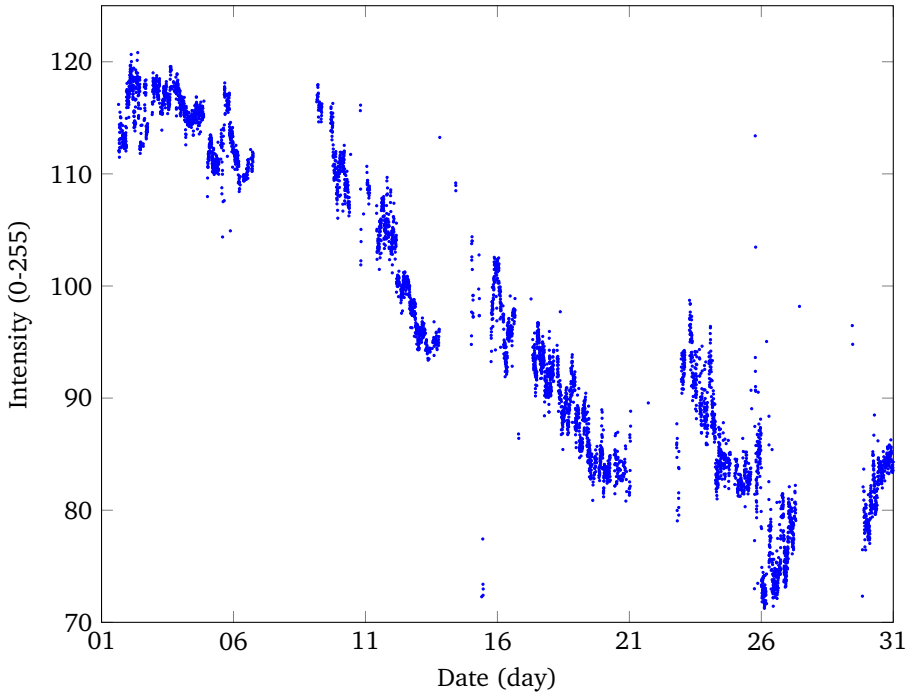

Figure 5.7: Average laser line intensity for triangulation unit L_4, for each of the 5233 rails that were measured during May 2017.

the daily average. No patterns are apparent. Additionally, the median difference (as described before) for each 15 minute interval was computed for the rails measured during May 2017 and for all the rails measured during the year. No correlation was found. Therefore, laser line intensity does not appear to follow a daily cycle, and the reasons for its behavior are still unknown.

5.3.2 Line width

The laser line should be wide enough for it to be extracted all along the rail surface, but a thick line may also indicate overexposition. Therefore, the line width should also be considered as a feature.

The line width is computed for all points in the extracted line using Steger's Gaussian line model [Steger, 2013]. Then, the widths at all line points in all the images of the same rail are averaged for every camera. Figure 5.9 depicts how the line width is computed.

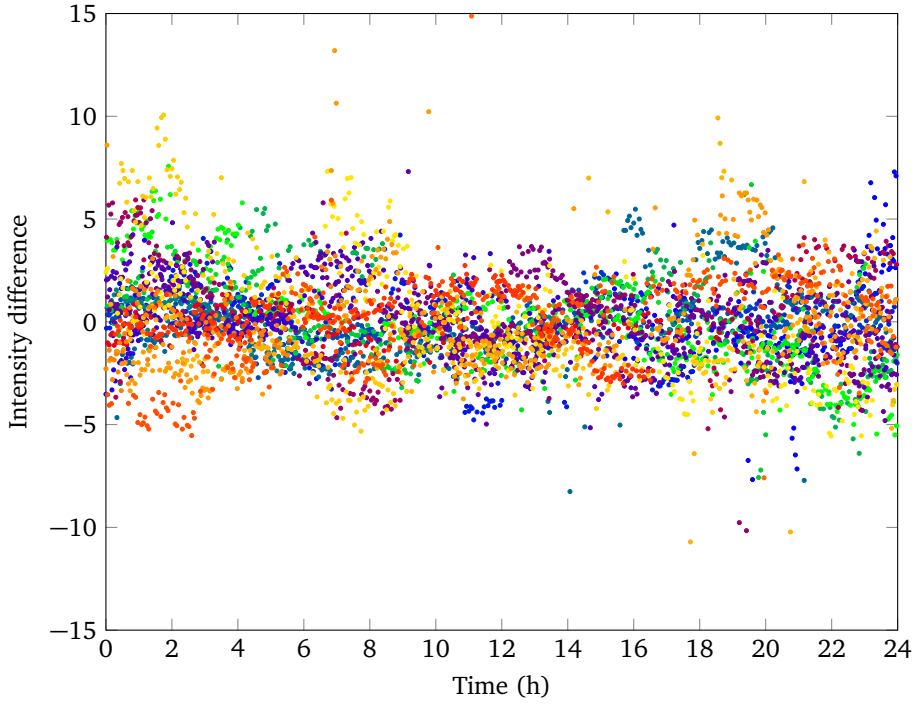

Figure 5.8: Evolution of the line intensity for triangulation unit L_4 during the day. For each of the 5233 rails that were measured during May 2017, the difference between its average intensity and the daily average is shown. Rails that were measured during the same day are depicted in the same color.

5.3.2.1 Results

Figure 5.10 shows the average line width in millimeters for all the rails that were measured every day from January 1, 2017 to December 31, 2017.

The vertical lines have the same meaning as stated before. Line widths seem to degrade in a monotonic manner for Camera 3, except for some time between May and June. However, Camera 4 appears to have thicker lines than all the other cameras, as well as a higher deviation.

5.3.3 Coverage ratio

The coverage ratio measures how much of the rail perimeter is actually covered by the laser stripe. Even if the laser stripe is wide and bright enough in most areas, it

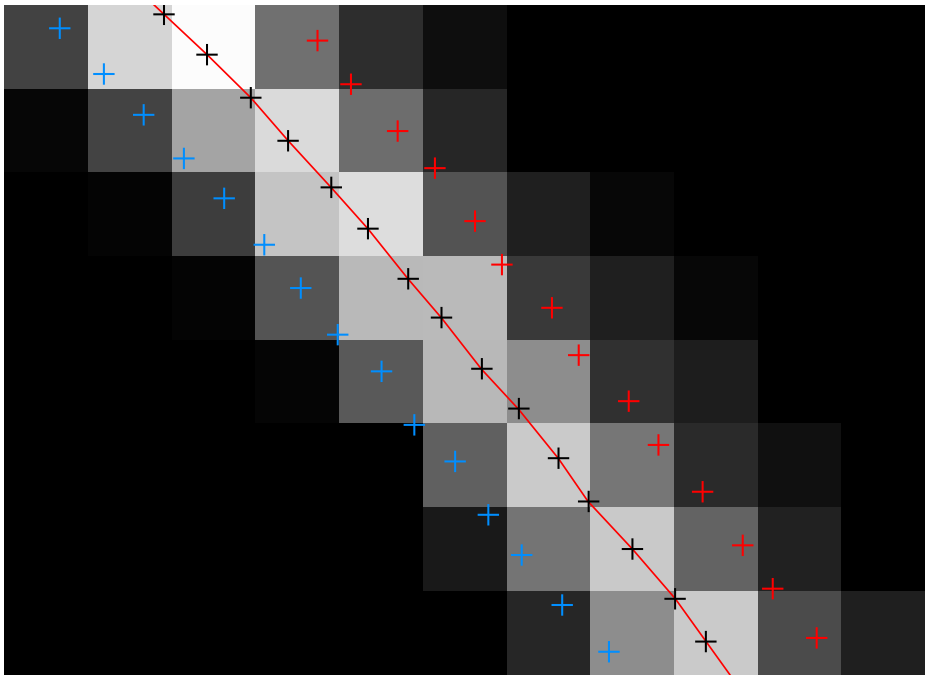

Figure 5.9: Line width. The laser line (red) and the extracted points are shown (black crosses), superimposed to the raw image. The points where the line is considered to end ($|x| = w$ in Steger's model) are also depicted (blue and red crosses).

may be weaker in certain regions.

For every image, the coverage ratio can be computed as follows:

A model point cloud is built by sampling the primitives (arcs and segments) from certain regions of the rail that are expected to be completely visible to the camera for all rail models. The sampling distance, defined as the distance between two consecutive points, measured on the given primitive, is 0.5 mm. A laser line point cloud is built from the extracted line after it is undistorted, translated to world coordinates and aligned with the model.

A point in the model point cloud, p_m, is considered to be covered if there is a point in the laser line point cloud, p_l, such that $d(p_m, p_l) \leq 1\,\text{mm}$, where d is the Euclidean distance between two points. The value of 1 mm and the sampling distance were chosen so that only actual gaps in the laser lines are detected, and smaller faults in the profile due to noise or miscalibration are not.

The coverage ratio is computed as the ratio between the number of covered points and the total number of points in the model point cloud. Then, the average coverage

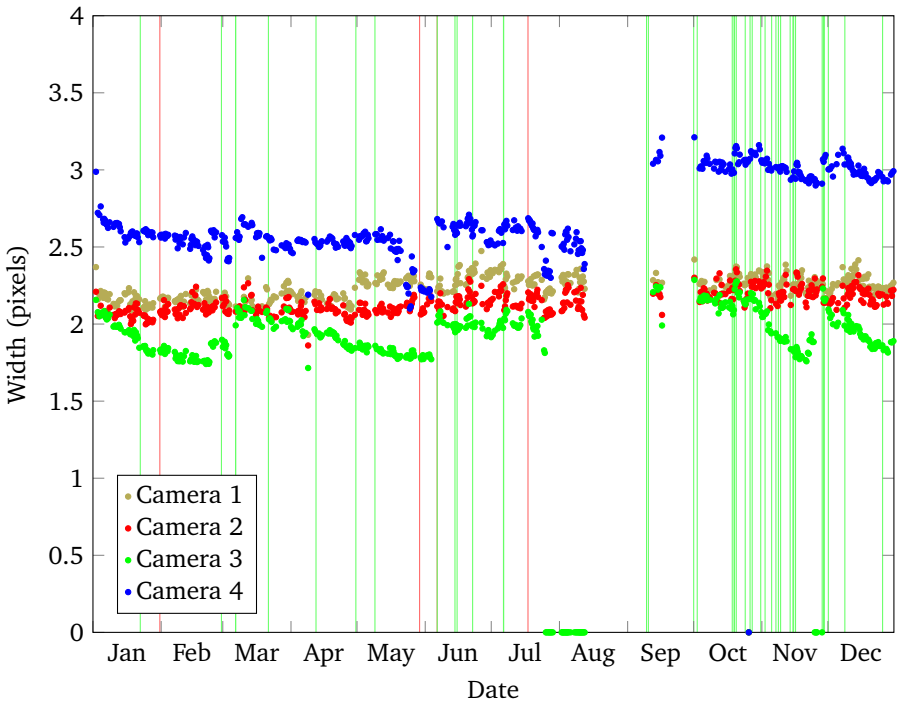

Figure 5.10: Line width for all cameras, for rails measured from January 2017 to December 2017.

ratio is computed for the set of all the images of the same rail that were captured by the same camera. Figure 5.11 depicts the model primitives that were considered, along with a sample laser line.

5.3.3.1 Results

Figure 5.12 shows the average coverage ratio for all the rails that were measured from January 1, 2017 to December 31, 2017.

Cameras 1 and 2 (the upper cameras) can be seen to have a stable coverage ratio. Cameras 3 and 4 (the lower cameras), however, can be seen to degrade very quickly. As such, the observed degradation may be attributed to the accumulation of oil and steel particles, which would mainly affect the lower cameras and laser emitters.

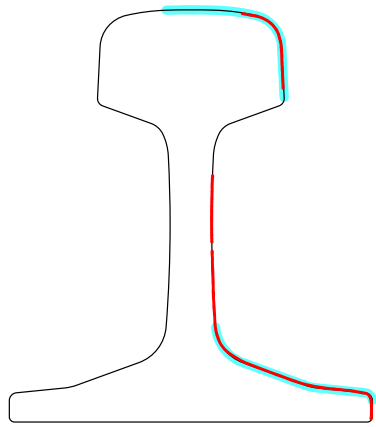

Figure 5.11: An example of coverage for Camera 2. The model primitives that were considered are shown in cyan, while the actual laser line is colored red.

5.3.4 Line distance

Over time, the position of the cameras and laser emitters is expected to change, mostly due to the vibrations of the structure caused by the movement of the rails. This means that calibrations must be performed regularly in order to compensate for such changes. The distance between overlapping laser lines as seen by multiple cameras can be used to determine whether the calibration data is still valid. Unlike the previous features, the line distance is not merely a feature of the images, although it is still caused by changes in the triangulation units.

From the physical description of the system, as described in Chapter 3, and especially Figure 3.2, laser lines overlap at four different regions:

1. The lines for cameras 1 and 2 overlap at the head arc.
2. The lines for cameras 1 and 4 overlap at the rail web.
3. The lines for cameras 2 and 3 also overlap at the rail web.
4. The lines for cameras 3 and 4 overlap at the foot.

Empirically, overlaps at the rail web have been identified as the most unreliable, as laser lines are often weak there, and that region is prone to occlusion. Therefore, only overlaps at the head (‘upper double line’) and the foot (‘lower double line’) will be considered (as seen in Figure 5.13).

However, these overlaps are not completely reliable when used by themselves, as some parts of the laser lines may be missing. This means that distance between

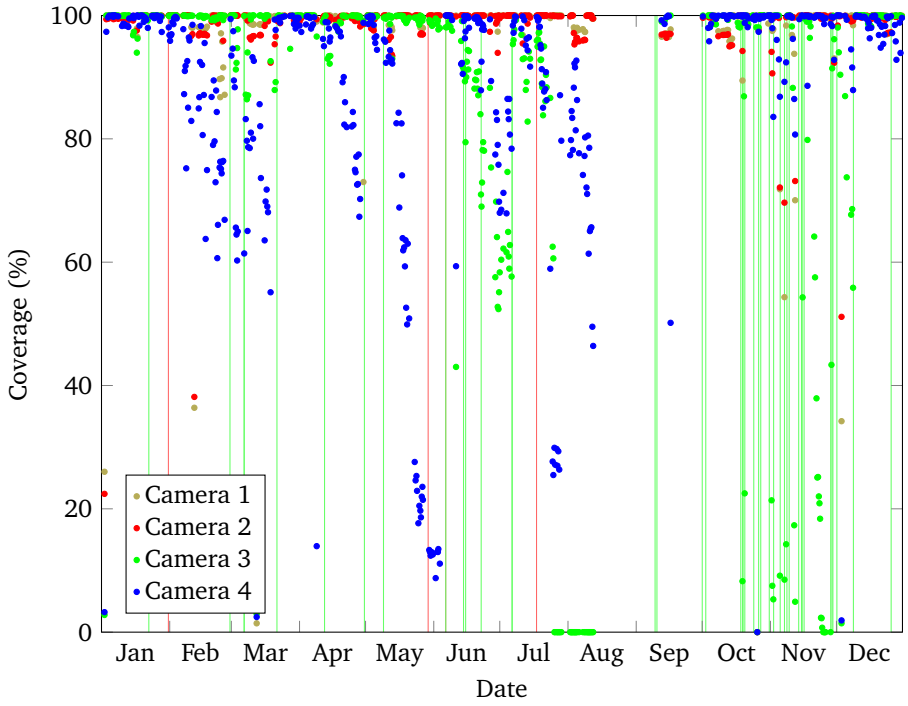

Figure 5.12: Coverage for all cameras, for rails measured from January 2017 to December 2017.

lines cannot be computed for all images. This condition should be detected by the other sensors, and it will not be considered here: rail sections where line distance cannot be computed will be ignored for the purposes of this sensor. This should not introduce any bias, as the quality of the laser line is independent from the quality of the calibration.

As seen in Figure 5.14, line distance for a given set of triangulation units (lower or upper) can be computed as follows:

1. The points from both laser lines are fit to an arc or a line, depending on the corresponding rail model primitive.
2. Normal vectors of the model every 1 mm are computed.
3. The distances between the points where the arcs or lines intersect each of the normal vectors are computed.
4. The computed distances are averaged.

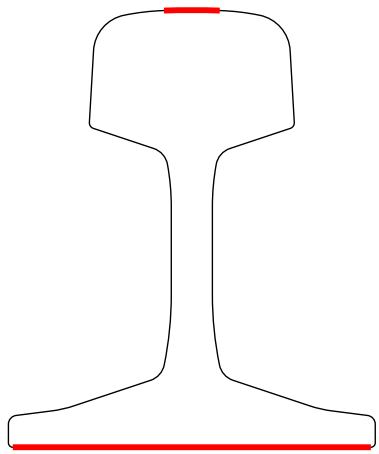

Figure 5.13: Regions that are considered for the line distance computation.

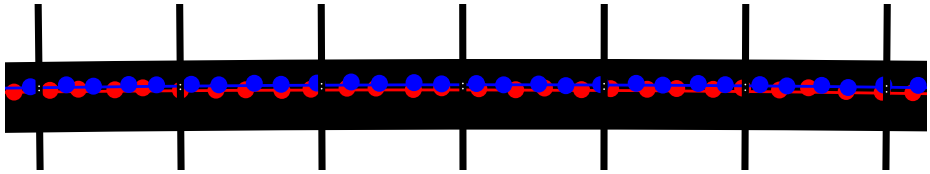

Figure 5.14: Line distance computation. Part of the head of the rail model is shown (black), along with the normal vectors (also black, vertical), and both laser lines (red and blue). The points of intersection of the laser lines with the vectors are also depicted (small white dots).

5.3.4.1 Results

Figure 5.15 shows the average line distances for all the rails that were measured in 2017. Each of the green lines denotes that the system was calibrated at the time, as stated before. The rails for which line distances could not be measured are not shown here.

Line distances were expected to slowly increase after every calibration. However, they seem to be mostly stable, excluding the outliers, especially for the upper cameras. More surprisingly, they show a slight tendency to decrease at some points.

Figure 5.16 shows the difference between the average line distance for each rail and the line distance immediately after the last calibration event before that rail. For this purpose, the line distance immediately after the last calibration event is defined

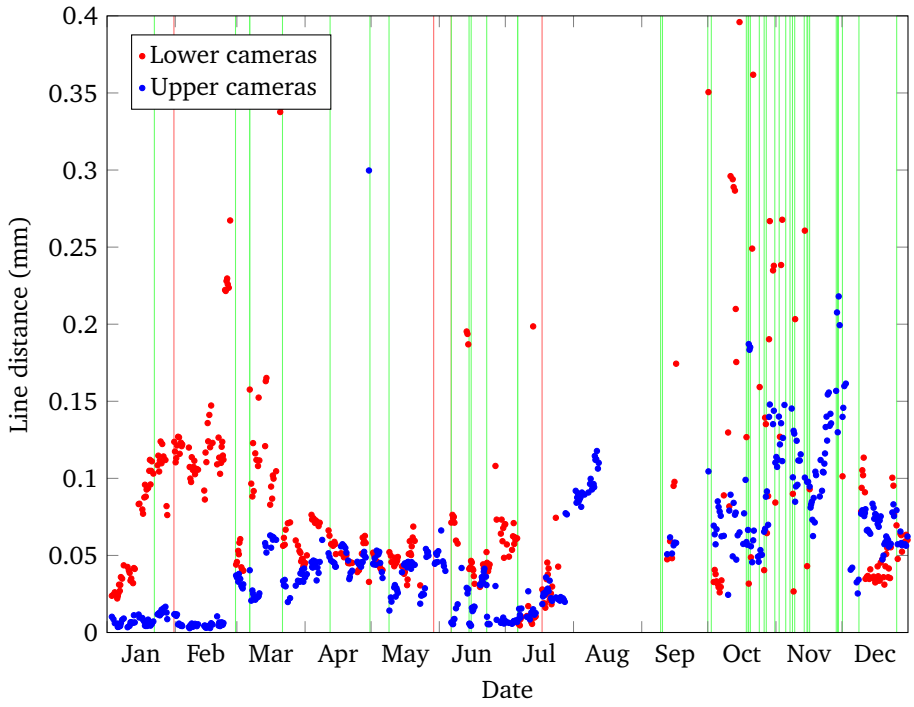

Figure 5.15: Line distance for all cameras, for rails measured from January 2017 to December 2017.

as the average line distance for the first 100 rails that were measured immediately after said event. It is clear from this figure that, most of the time, line distance experiences few changes between calibration events for the upper cameras. The lower cameras seem to be slightly more susceptible to degradation.

5.3.5 Missing dimensions

The number of sections for which some dimensions cannot be computed is a good indicator for the overall health of the system. Although the measurement system is robust enough to operate in degraded conditions to a certain extent, it will fail to compute certain dimensions when the relevant parts of the rail profile cannot be found.

Unlike the previous features, this relates to the output of the measurement system, rather than its input, as seen in Figure 5.4. Therefore, it can be used to check whether a given issue in a triangulation unit affects the output in a major way. However, it

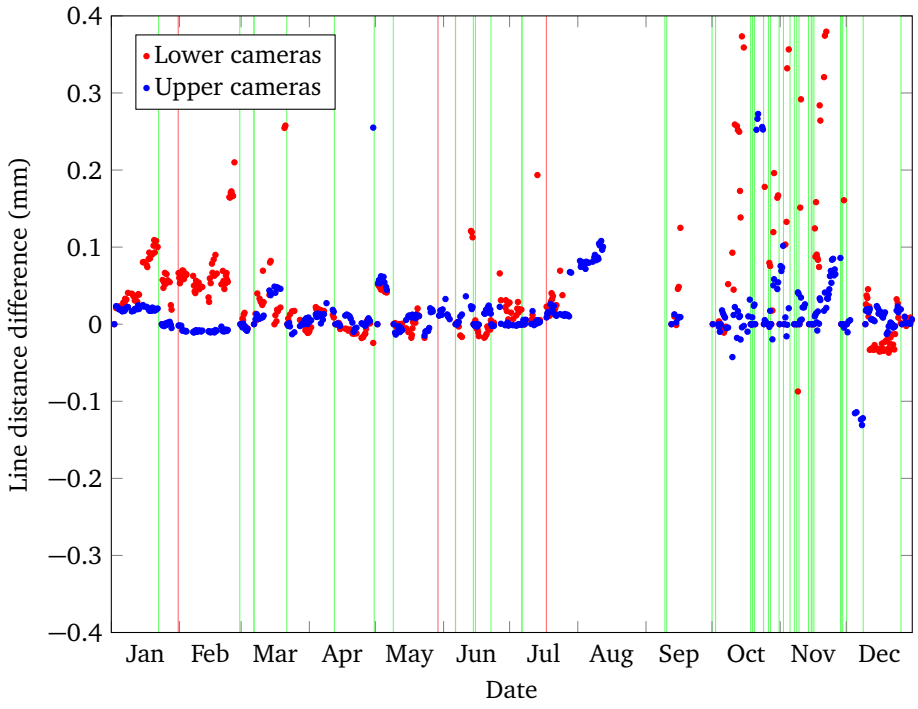

Figure 5.16: Line distance for all cameras, for rails measured from January 2017 to December 2017.

must be noted that significant accuracy losses may occur before a signle dimension is lost, so this feature should not be used by itself as the single basis for a self-assessment feature.

Dimensions are considered missing if the computed value for a given section is stored as *NaN* (not a number) and the expected value for the rail type is not *NaN* (as some dimensions do not apply to certain rail types). From this, a list of which dimensions are missing for a given section can easily be computed. Table 5.1 shows an example of such a list.

Although computing the ratio of missing dimensions for a given rail*, or the ratio of sections for which any dimension is missing, or the ratio of sections for which a given dimension is missing, would be trivial, such raw values would be of little use for self-assessment. Therefore, in order to make use of this feature, additional processing will be required. This will be covered in Section 5.4.

* Defined as $\frac{Num.missing\ dimensions}{Num.dimensions \cdot Num.sections}$.

Sections	D_1	D_2	...	D_n	Any	All
S_1	0	1	...	0	1	0
S_2	0	0	...	0	0	0
...	...	...	...	...	...	...
S_m	1	1	...	1	1	1
Overall	10%	25%	...	15%	75%	5%

Table 5.1: Example of a table of missing dimensions. Here, 0 means that a given dimension could be computed for a specific section, and 1 means it could not.

5.3.5.1 Results

Figure 5.17 shows the percentage of rail sections for which at least one dimension was missing, and the percentage of rail sections for which every single dimension was missing, for all the rails measured in 2017.

While such percentages are of little practical use by themselves, as stated before, this figure helps understand how missing dimensions are distributed: it can be seen that rails with high percentages of sections with missing dimensions usually appear in groups: relatively short bursts of rails with missing dimensions that typically last no longer than three days, appearing in the figure as almost vertical lines. Additionally, when no dimensions can be computed for a given rail section, typically no dimensions can be computed for any other sections in the same rail. This may be because issues that cause no dimensions to be computed, such as a wrong rail model, or unpowered laser emitters, usually affect a whole rail, rather than individual sections.

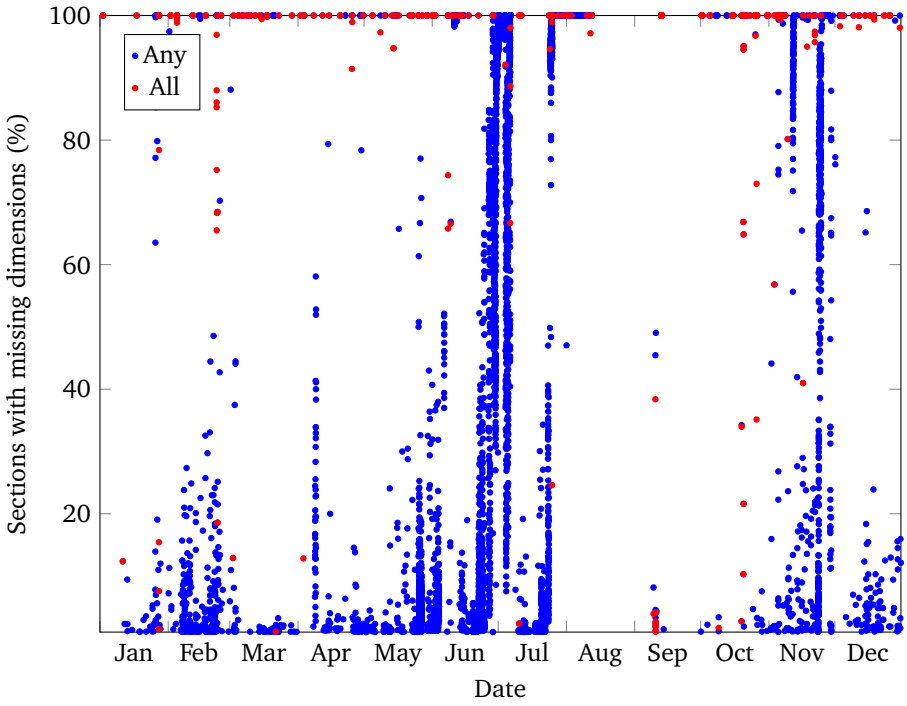

Figure 5.17: Percentage of sections for which any dimensions were missing, for all the rails measured in 2017. Sections for which all dimensions were missing are also shown. Here, every dot represents a rail. Rails with no missing dimensions in any section are not shown.

5.4 Determining the cause of missing dimensions

As explained in Section 5.3.5, the percentages of missing dimensions cannot be used by themselves. Further processing is required in order to determine their cause. In this section, different possibilities to associate one or more missing dimensions with issues with one or more triangulation units will be discussed. Figure 5.18 shows the inputs and outputs of a result analysis module that would perform the required processing, with respect to the simplified system shown in Figure 5.4.

5.4.1 Grouping missing dimensions

In order to identify different faults depending on which dimensions are missing at a given time, one possible approach is to determine which other dimensions are

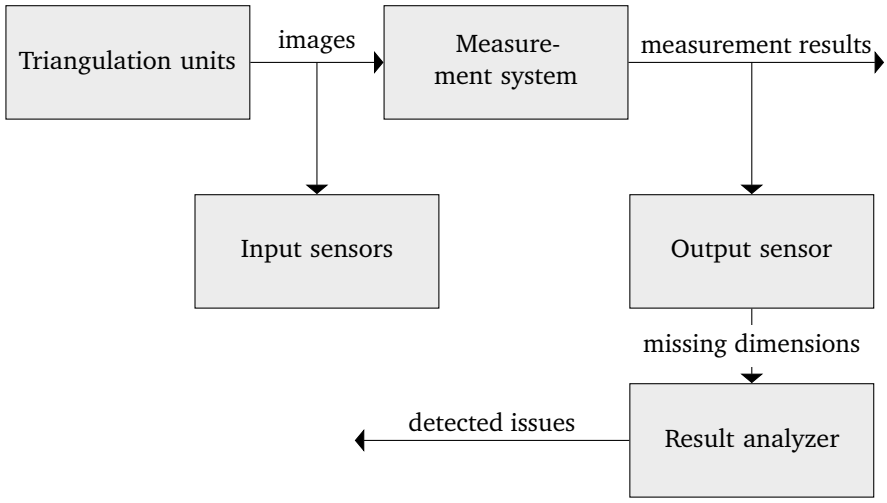

Figure 5.18: Role of the result analyzer with respect to other components.

typically missing when a given dimension is. This way, the correlation between dimension failures may be studied.

However, this approach is somewhat naïve, in that unrelated dimensions may be combined just because they are missing in the same section. A more robust approach could be considered which would consider whether the first and last invalid sections for all the grouped dimensions are the same, although this has not been implemented.

A second alternative approach is proposed, where only a set of predefined combinations of dimensions is considered, thereby removing most of the complexity of the previous approach. In this case, for a given combination of dimensions, a section is invalid if the values of all the dimensions in the combination are missing. In order to determine which dimensions should be grouped, experiments were carried out as follows.

5.4.1.1 Missing laser line in the production environment

1. The images for a single rail section are taken from a rail that was measured in the production environment.

2. The rail section is measured four times, and every time a blank image is substituted for that of one of the cameras.

3. The rail section is measured four times, and every time blank images are substituted for those of two of the cameras.

Figure 5.19 and Figure 5.20 show the rail profiles that were built from these measurements. The rail model is shown in cyan, while the actual profiles are shown in red. For certain combinations of blank images (respectively: missing upper laser lines, missing right laser lines, missing left laser lines), the system was unable to compute the correct alignment, as shown in Figure 5.21. The correct alignment was computed for all the other profiles.

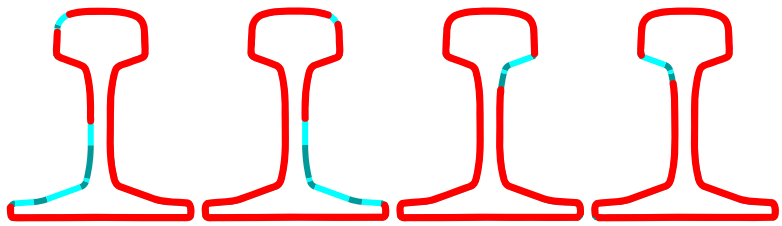

Figure 5.19: Profiles with a single missing laser line (from left to right: blank C_1, C_2, C_3, and C_4).

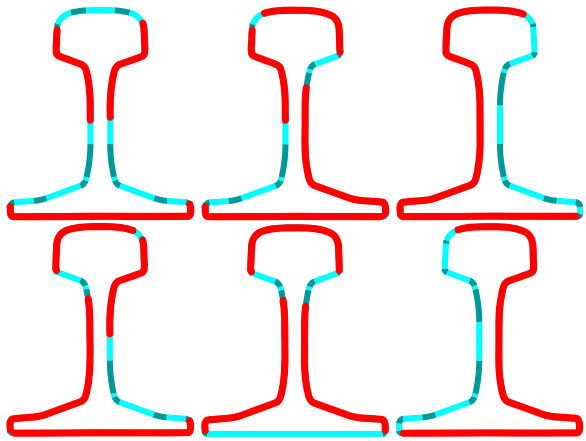

Figure 5.20: Profiles with two missing laser lines (from left to right, first row: blank C_1 and C_2, C_1 and C_3, C_2 and C_3; second row: blank C_2 and C_4, C_3 and C_4, C_4 and C_1).

The following dimensions were missing:

- **Unaltered:** None.
- **Blank C_1:** RFH, FT_l, FTI_l, H1_l, H2_l.

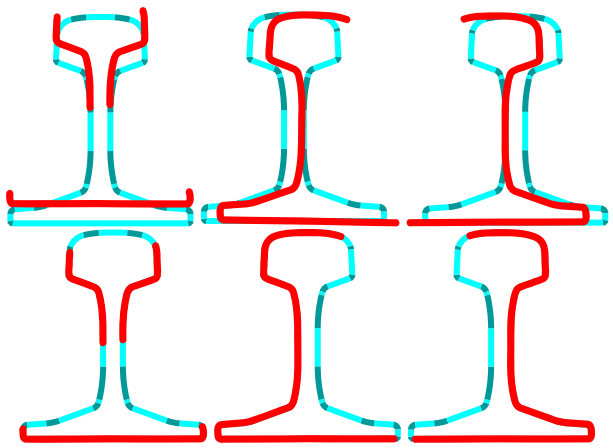

Figure 5.21: Bad alignment due to missing laser lines (first row: computed alignment, second row: correct alignment; from left to right, blank C_1 and C_2, C_2 and C_3, C_4 and C_1).

- **Blank C_2:** RFH, FT_r, FTI_r, H1_r, H2_r.
- **Blank C_3:** RFH, H1_r, H2_r.
- **Blank C_4:** RFH, H1_l, H2_l.
- **Blank C_1 and C_2:** All (as stated before, this is due to bad alignment).
- **Blank C_3 and C_4:** RH, RAS_Arcelor, RFH, FW, FC, FT_l, FTI_l, FT_r, FTI_r, H1_l, H1_r, H2_l, H2_r.
- **Blank C_2 and C_3:** RAS_Arcelor, RAS_l, RAS_r, RFH, HR1_l, HW, HWArema, WT, FW, FT_l, FT_r, FTI_r, H1_r, H2_r (due to bad alignment, as above).
- **Blank C_4 and C_1:** RAS_Arcelor, RAS_l, RFH, HR1_l, HW, HWArema, WT, FW, FT_l, FTI_l, FT_r, H1_l, H2_l (same as above).
- **Blank C_1 and C_3:** RFH, FT_l, FTI_l, H1_l, H1_r, H2_l, H2_r.
- **Blank C_2 and C_4:** RFH, FT_r, FTI_r, H1_l, H1_r, H2_l, H2_r.

5.4.1.2 Missing laser line in the laboratory prototype

1. The laboratory prototype is calibrated.

2. A rail section is measured where all the relevant dimensions can be computed (BaseCalib).

3. The rail section is captured four times, and every time one of the laser emitters is completely occluded.

The following dimensions were missing:

- **Unaltered:** None.
- **Occluded E_1:** RFH, FT_l, FTI_l, H1_l, H2_l.
- **Occluded E_2:** RFH, FT_r, FTI_r, H1_r, H2_r.
- **Occluded E_3:** RFH, H1_r, H2_r.
- **Occluded E_4:** RFH, H1_l, H2_l.

5.4.1.3 Moved cameras in the laboratory prototype

1. The laboratory prototype is calibrated.

2. A rail section is measured where all the relevant dimensions can be computed.

3. The four cameras are turned inwards by roughly 10° and the rail section is measured again.

No dimensions were missing.

5.4.1.4 Moved laser emitters in laboratory prototype

1. The laboratory prototype is calibrated.

2. A rail section is measured where all the relevant dimensions can be computed.

3. The four laser emitters are misaligned in such a way that the longitudinal distance between the laser lines is no less than 3 mm and no more than 10 mm at all points of the rail profile, and the rail section is measured again.

No dimensions were missing.

5.4.2 Defining rules

Using the information gathered as described in the previous section, a set of rules may be established that will associate patterns in dimension failures with issues with one or more cameras or lasers.

In this section, the proposed rules will be expressed with the following notation:

$$\overline{D_{m_a}} \wedge \overline{D_{m_b}} \wedge ... \wedge D_{p_a} \wedge D_{p_b} \wedge ... \rightarrow L_a \wedge L_b \wedge ...$$

Which means that when all the dimensions $D_{m_a}, D_{m_b}, ...$ are missing, and all the dimensions $D_{p_a}, D_{p_b}, ...$ are present, issues with all of $L_a, L_b, ...$ exist. Here, L_n refers to the triangulation unit formed by the laser emitter numbered n and the camera numbered n. This means that issues with a given unit may be caused by either the laser or the camera.

All the rail dimensions described in Section 3.6, must be considered, and those that unambiguously indicate issues with one or more specific triangulation units will be used to define the rules. In this way, the following rules can be tentatively suggested:

- **RH:** Failure to compute rail height means that either the top of the rail head is missing (issues with both L_1 and L_2) or the base of the foot is (issues with both L_3 and L_4). As such, this dimension is ambiguous for this purpose and cannot be used by itself.

- **HW and its variants:** Failure to compute head width means that one or both sides of the rail head are missing. A missing left side denotes issues with both L_1 and L_4, while a missing right side denotes issues with both L_2 and L_3. As such, these dimensions are ambiguous and cannot be used by themselves.

- **HF1 and HF3:** Failure to compute head form means that the top of the rail head is missing. This denotes issues with both L_1 and L_2. Therefore, the following rules may be defined:

  $$\overline{HF1} \rightarrow L_1 \wedge L_2$$

  $$\overline{HF3} \rightarrow L_1 \wedge L_2$$

  As the presence of these dimensions denotes that the top of the rail head is not missing, the presence of either of them combined with the absence of RH indicates that the base of the foot is lost. Therefore, we may define the rule:

  $$\overline{RH} \wedge \overline{HF1} \rightarrow L_3 \wedge L_4$$

- **HR0_l, HR0_r, HR1_l, HR1_r, HR2_l, HR2_r and HR3:** Failure to compute one or more of the head arcs means that L_1 presents issues, in the cases of HR0_l, HR1_l and HR2_l, that L_2 presents issues, in the cases of HR0_r, HR1_r

and HR2_r, or that both L_1 and L_2 present issues, in the case of HR3. Therefore, the following rules may be defined:

$$\overline{HR0\_l} \rightarrow L_1 \qquad \overline{HR0\_r} \rightarrow L_2$$

$$\overline{HR1\_l} \rightarrow L_1 \qquad \overline{HR1\_r} \rightarrow L_2$$

$$\overline{HR2\_l} \rightarrow L_1 \qquad \overline{HR2\_r} \rightarrow L_2$$

$$\overline{HR3} \rightarrow L_1 \wedge L_2$$

- **RAS_l and RAS_r:** Failure to compute rail asymmetry denotes issues with the camera-laser sets of the corresponding side (L_1 and L_4 for RAS_l, L_2 and L_3 for RAS_r). These issues may be located either on the foot or on the side of the head, or both. Therefore, we may define the following rules:

$$\overline{RAS\_l} \rightarrow L_1 \wedge L_4 \qquad \overline{RAS\_r} \rightarrow L_2 \wedge L_3$$

- **RAS_Arcelor:** This dimension presents the same issues as stated in the case of HW.

- **RFH:** Failure to compute rail flange height denotes issues with either L_1, L_2, L_3 or L_4. As seen before, this dimension cannot be computed if any of the laser lines is missing. Therefore, it conveys no useful information for the purpose of this sensor.

- **H1_l, H1_r, H2_l and H2_r:** Failure to compute any of these dimensions denotes issues with the corresponding upper laser on the rail foot, or with the corresponding lower laser on the underside of the rail head. Therefore, these dimensions cannot be used by themselves.

- **WT:** Failure to compute web thickness denotes the complete loss of either side of the web. Which side is missing cannot by determined in this way, so this dimension cannot be used for this purpose.

- **FT_l, FT_r and their variants:** Failure to compute the foot thickness denotes issues with L_1 (for FT_l) or L_2 (for FT_r) on the upper part of the foot, or with L_3 (for FT_r) or L_4 (for FT_l) on the sides of the foot. Failure to compute FT_l or FT_r may also be caused by a poor alignment, due to the loss of most of the laser line on the opposite side. FTI_l and FT_r appear to be more resilient in this case. Whether the issues are located on the upper part or on the side cannot be determined, and so these dimensions cannot be used by themselves.

  The presence of the foot thickness denotes that the side and the upper part of the foot are not missing. This means that if H1_l is missing but FT_l is present, then the issue is related to L_4, and located on the underside of the rail head;

and, analogously, if H1_r is missing when FT_r is present, the issue is related to L_3. Therefore, we can define the following rules:

$$\overline{H1\_l} \wedge FT\_l \rightarrow L_4 \qquad \overline{H1\_r} \wedge FT\_r \rightarrow L_3$$

- **FW and its variants:** Failure to compute foot width denotes the complete loss of one of the sides of the foot. Which of the sides is lost cannot be determined from this, so these dimensions cannot be used by themselves.

- **FC:** Failure to compute foot concavity denotes issues with both L_3 and L_4, and the loss of parts of the base of the foot. Therefore, the following rule may be defined:

  $$\overline{FC} \rightarrow L_3 \wedge L_4$$

  In order to compute this dimension, the base of the foot is split into many segments, and each segment is considered separately, and all of them are expected be present. This means that this dimension is very sensitive to the loss of small line segments.

  As the presence of this dimension denotes that the base of the rail foot is not missing, its presence combined with the absence of RH indicates that the top of the head is lost. Therefore, we may define the rule:

  $$\overline{RH} \wedge FC \rightarrow L_1 \wedge L_2$$

  Similarly, the presence of FC combined with the absence of the foot thickness means that the loss of the foot thickness was not caused by one of the lower camera-laser sets, but by one of the upper ones. Thus, we can state the following rules:

  $$\overline{FTI\_l} \wedge FC \rightarrow L_1 \qquad \overline{FTI\_r} \wedge FC \rightarrow L_2$$

  FTI is used here, rather than FT, because it is more resilient to bad alignment, as stated before, and to actual faults in the shape of the sides of the rail foot. This is expected to reduce the chance of false positives for these rules.

  As stated before, when FW is missing, which side of the foot was lost cannot be immediately determined. However, if FC is present and FT for a given side is also present, then we know that the other side is the one that was lost. Thus, we can define the rules:

  $$\overline{FW} \wedge FT\_l \wedge FC \rightarrow L_2 \wedge L_3 \qquad \overline{FW} \wedge FT\_r \wedge FC \rightarrow L_1 \wedge L_4$$

- **AREA:** This dimension is not useful for this purpose.

5.4.3 Validating the rules

For each of the rules defined in Section 5.4.2, all the rail sections that were measured in 2016 and that contain missing dimensions that fulfill the rule prerequisites were recorded.

Among the recorded rails, all the sections that fulfill the prerequisites for $\overline{HF1} \rightarrow L_1 \wedge L_2$, $\overline{HF3} \rightarrow L_1 \wedge L_2$ and $\overline{RH} \wedge FC \rightarrow L_1 \wedge L_2$ also fulfill the prerequisites for $\overline{HR3} \rightarrow L_1 \wedge L_2$. Thus, we may say these rules are redundant. Similarly, $\overline{RH} \wedge HF1 \rightarrow L_3 \wedge L_4$ is redundant because all the sections that fulfill it also fulfill $\overline{FC} \rightarrow L_3 \wedge L_4$. $\overline{FW} \wedge FT\_l \rightarrow L_F C2$, 3 and $\overline{FW} \wedge FT\_r \wedge FC \rightarrow L_1 \wedge L_4$ are also redundant because all the sections that fulfill them also fulfill $\overline{RAS\_r} \rightarrow L_2 \wedge L_3$ and $\overline{RAS\_l} \rightarrow L_1 \wedge L_4$, respectively. These redundant rules will not be considered.

All the sections that fulfill $\overline{HR0\_r} \rightarrow L_2$ also fulfill $\overline{HR1\_r} \rightarrow L_2$. However, this is not true for $\overline{HR0\_l} \rightarrow L_1$ and $\overline{HR1\_r} \rightarrow L_1$, which means that a section that fulfills $\overline{HR0\_r} \rightarrow L_2$ and does not fulfill $\overline{HR1\_r} \rightarrow L_2$ can potentially exist, so $\overline{HR0\_r} \rightarrow L_2$ will not be removed. No other redundant rules were observed.

Sections where few or no dimensions can be computed must be treated differently, because the loss of so many dimensions is usually caused by a system failure, bad alignment or rail type mismatch, rather than by issues with the camera-laser sets. Therefore, sections where half or more of the total number of dimensions are missing are grouped as *All* and are ignored by all other rules.

Sections where one or more dimensions are missing, but which do not match the prerequisites for any other rule, are grouped as *Unknown*.

In order to validate the remaining rules, an experiment is carried out as follows. For each of the non-redundant rules (and $\overline{HR0\_r} \rightarrow L_2$, as explained), 30 samples (with replacement) are taken from all the sections that fulfilled its prerequisites. These samples are visually checked so as to ensure the validity of the rules:

- $\overline{HR0\_l} \rightarrow L_1$: In 30 out of 30 samples part of the rail head is missing from the image.
- $\overline{HR1\_l} \rightarrow L_1$: In 2 out of 30 samples part of the rail head is missing (in one of them, it is apparently missing from the actual rail). In the other 28 samples, the rail type does not match the stated type.
- $\overline{HR0\_2} \rightarrow L_1$: In 19 out of 30 samples part of the rail head is missing or the laser line is too wide. In the remaining 11 samples, the rail type does not match the stated type.
- $\overline{HR0\_r} \rightarrow L_2$: In 30 out of 30 samples part of the rail head is missing.
- $\overline{HR1\_r} \rightarrow L_2$: In 30 out of 30 samples part of the rail head is missing from the image or the laser line was too wide.

- $\overline{HR2\_r} \rightarrow L_2$: In 30 out of 30 samples part of the rail head is missing from the image or the laser line is too wide.
- $\overline{HR3} \rightarrow L_1 \wedge L_2$: In 30 out of 30 samples part of the rail head is missing from the images.
- $\overline{H1\_l} \wedge FT\_l \rightarrow L_4$: In 30 out of 30 samples the base of the head is missing.
- $\overline{H1\_r} \wedge FT\_r \rightarrow L_4$: In 28 out of 30 samples the base of the head is missing. In one of the other samples, most of the upper part of the rail foot is missing (issue with L_2, rather than L_3). In the other remaining sample, the actual rail type does not match the stated type.
- $\overline{RAS\_l} \rightarrow L_1 \wedge L_4$: In 25 out of 30 samples either the base of the head, the top of the foot or both are missing from the images of both L_1 and L_4. In the remaining 5 samples, parts of the left side of the foot are missing from the actual rail.
- $\overline{RAS\_r} \rightarrow L_2 \wedge L_3$: In 26 out of 30 samples either the base of the head, the top of the foot or both are missing from the images of both L_2 and L_3. In 3 of the remaining samples, parts of the right side of the foot are missing from the actual rail. In the remaining sample, the actual rail type does not match the stated type.
- $\overline{FTI\_l} \wedge FC \rightarrow L_1$: In 30 out of 30 samples part of the top of the rail foot is missing. At least 11 of the samples present an obvious occlusion due to an unidentified object over the foot of the rail.
- $\overline{FTI\_r} \wedge FC \rightarrow L_2$: In 28 out of 30 samples part of the top of the rail foot is missing. In the remaining samples, the rail profile is not correctly aligned.
- $\overline{FC} \rightarrow L_3 \wedge L_4$: In 30 out of 30 samples the base of the rail foot is either weak, with one or more holes, or missing altogether.
- All: 30 out of 30 samples showed a rail section that do not match the stated rail type. By design, the system relies on rail type information from external sources. Nothing can be done here about the issue.
- $Unknown$: Most of these samples present either calibration issues or missing laser lines on the rail web.

In this way, it can be seen that the described set of predefined rules can be used in order to accurately associate the failure to compute certain dimensions with issues with given triangulation units. Missing dimension results can be shown as a ratio for every triangulation unit and rail: the amount of sections that trigger one or more of

the rules associated with a given triangulation unit, divided by the total number of sections in the rail.

For example, a rule such as $\overline{D_m} \wedge D_{p_b} \rightarrow L_a$ will be triggered for a given rail section if dimension D_m is missing in that section and dimension D_{p_b} is not. In that case, such section will be counted as an issue for triangulation unit L_a. The ratio between the number of sections which trigger one or more rules for L_a and the total number of sections in the rail will be considered the missing dimensions ratio for L_a.

While this approach omits some information that could be useful (particularly, no information is given on the distribution of invalid sections along the rail), enough information is given to estimate the severity of the relevant issues, and to pinpoint their cause, and it is remarkably simpler than other discussed approaches. Therefore, this is the approach that will be used in the following sections.

5.4.4 Results

For all the rails that were measured in 2017, Figure 5.22 shows the percentage of rail sections for which the predefined rules associated with a given triangulation unit, among those described in Section 5.4.2, were triggered. Hereafter, such rail sections will be referred to as "missing sections" for the relevant triangulation unit.

It can be seen that most of the cases where any dimension is missing (as seen in Figure 5.17) trigger at least one of the predefined rules. The pattern of short bursts of rails with missing dimensions, lasting only a few days, is still apparent in this figure. Additionally, most, but not all, instances of sections with missing dimensions appear to indicate issues with either both the upper or both the lower cameras. Finally, missing sections appear to be much more common for the lower triangulation units (L_3, L_4) than for the upper ones (L_1, L_2). This is similar to the results of image features such as the coverage ratio (Section 5.3.3).

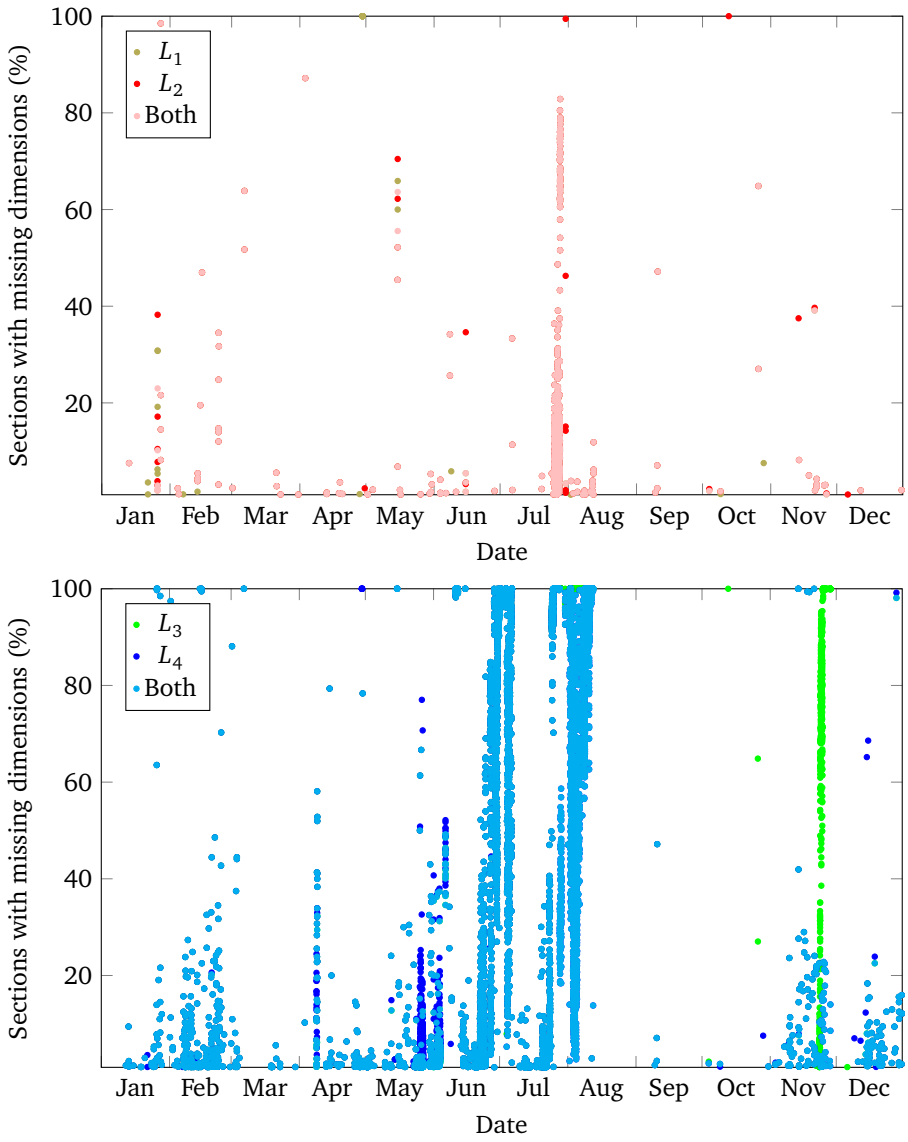

Figure 5.22: Percentage of sections for which the missing dimension rules were triggered, for all the rails measured in 2017. Here, every dot represents a rail. Rails with no missing dimensions in any section are not shown. L_n means that a predefined rule was triggered that indicated failure for triangulation unit L_n, but at least one dimension could be computed.

5.5 Self-assessment subsystem

After describing a set of features relevant to the health of the system, and techniques to compute values for such features (called "sensors" here), along with a module to determine the triangulation unit responsible for missing dimensions, a final module must be designed that translates such values into feedback for human operators. Figure 5.23 depicts the inputs and outputs of the profile measurement system along with the self-assessment module and other components described in the previous sections.

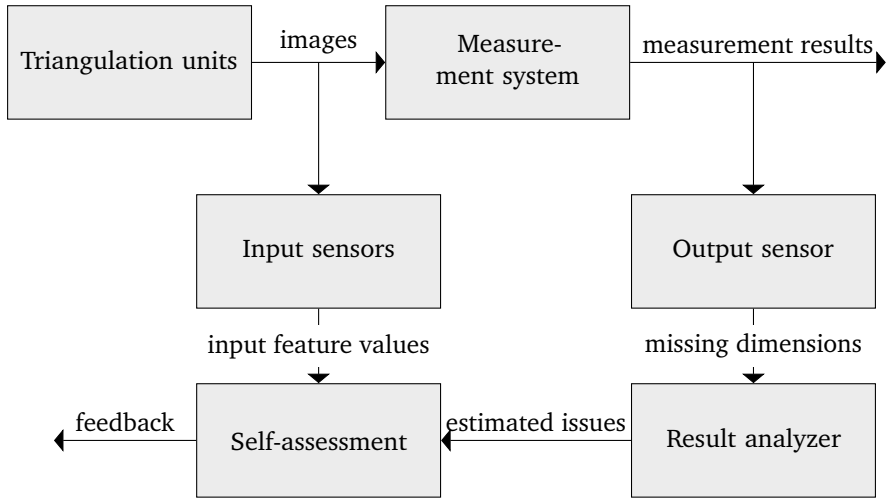

Figure 5.23: Role of the self-assessment module with respect to other components.

A proper autonomic system would close the loop between the self-assessment module and the triangulation units, by allowing the self-assessment system to act on the triangulation units and make changes if necessary. In the case of this system, however, human operators are required to act on the provided feedback, mostly by cleaning or replacing cameras or laser emitters as needed.

5.5.1 Correlation between laser stripe features and missing dimensions

If missing dimensions, as described in Section 5.3.5, can be used to detect system health issues, and most such issues are caused by problems with the laser stripes, then a correlation should exist between missing dimensions and the laser features described before. Finding such a correlation would help determine the validity of the described sensors.

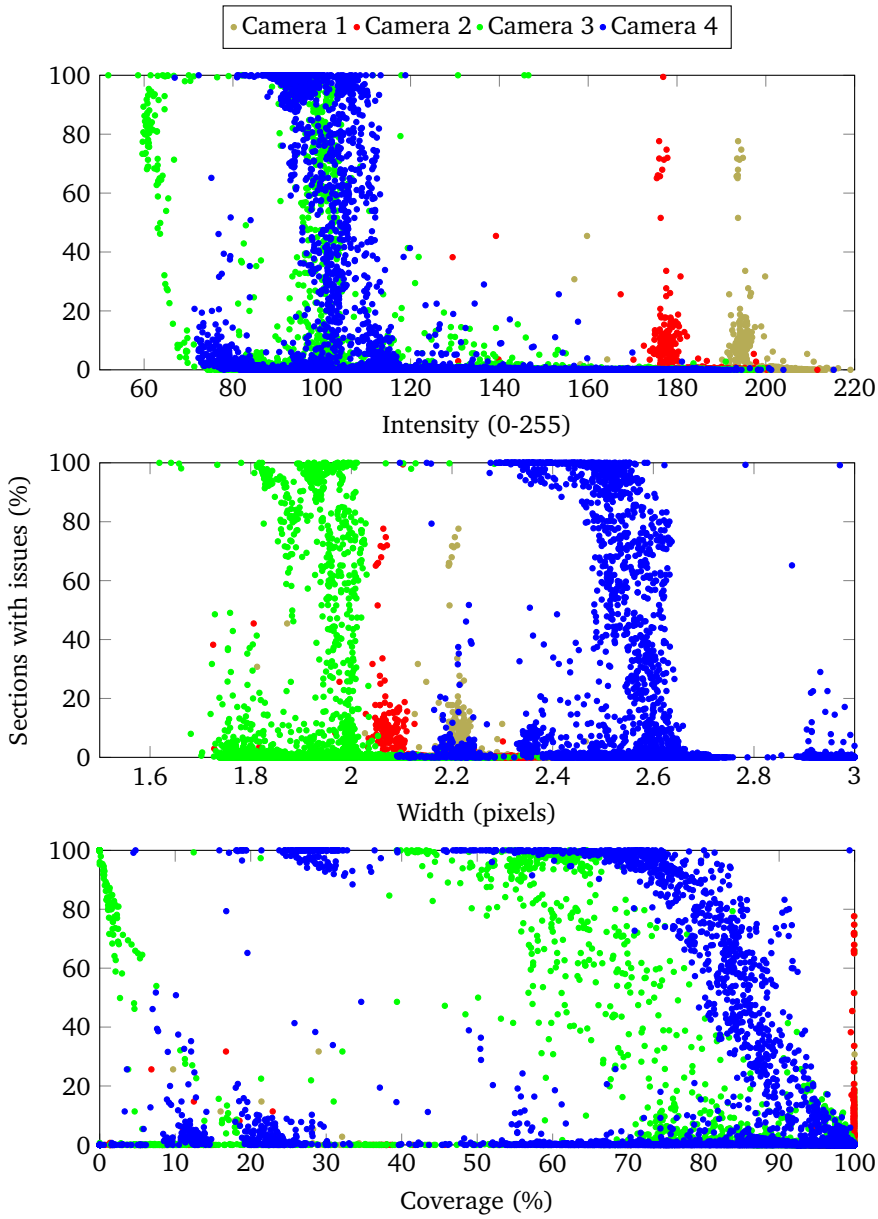

Figure 5.24: Percentage of missing sections as a function of image feature values, for rails measured from January 2017 to December 2017.

Figure 5.24 shows the correlation between the percentage of missing sections and, from top to bottom, laser intensity, line width and coverage, for all the rails measured in 2017. It can be seen how sensor values for the lower cameras are consistently worse than values for the upper cameras, as discussed before. Additionally, and more relevantly, little or no correlation is shown between sensor values and missing dimensions for the upper cameras, whereas correlation is apparent for the lower cameras, especially in the cases of intensity and coverage. The lack of correlation for the upper cameras may be related to the fact that they present little degradation of the considered features over time.

In order to offer a clearer representation, Figure 5.25 shows two histograms for every laser stripe feature and triangulation unit. In blue, the distribution of rails for which less than 5% of the measured sections are missing dimensions related to a given triangulation unit (as determined by the rules described in Section 5.4.2). In red, the remaining rails. A threshold of 5% is chosen in order to separate sporadic, unpredictable failures from long-term issues that more clearly signal a problem. Figure 5.26 shows the same data as a set of cumulative histograms.

In this way, the percentage of missing sections can be used both as an independent indicator of system health and as a tool to validate other sensors and determine reasonable limits for sensor values.

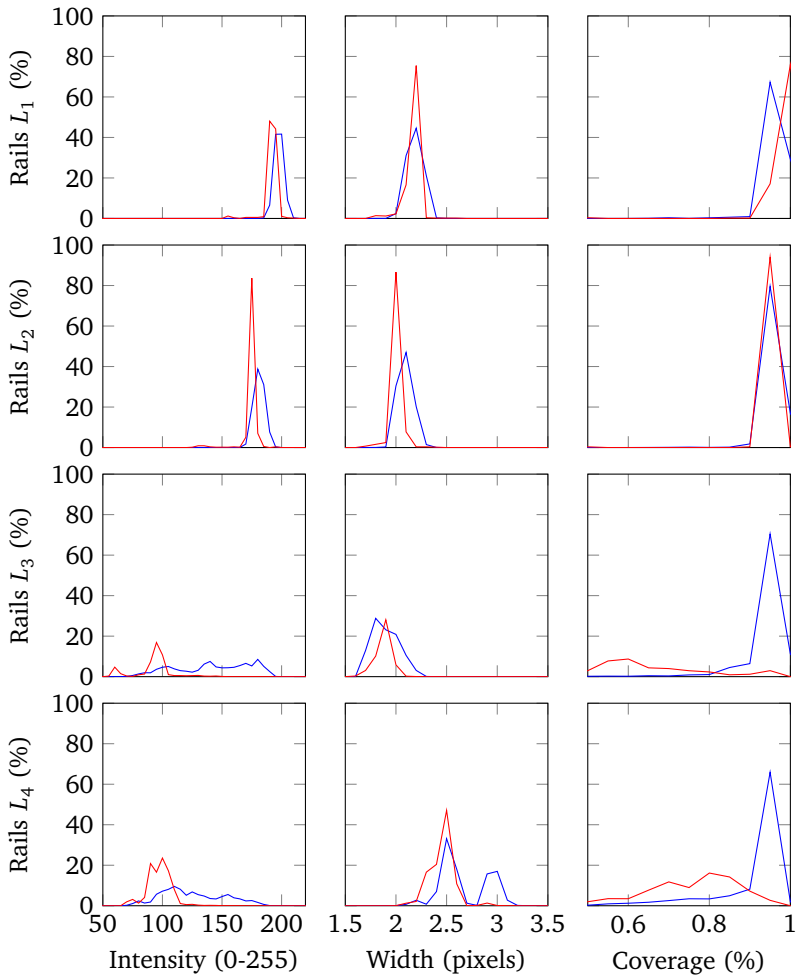

Figure 5.25: Laser feature histograms, for rails measured from January 2017 to December 2017. In blue, rails for which less than 5% of all sections had missing dimensions associated with a given triangulation unit. In red, the remaining sections.

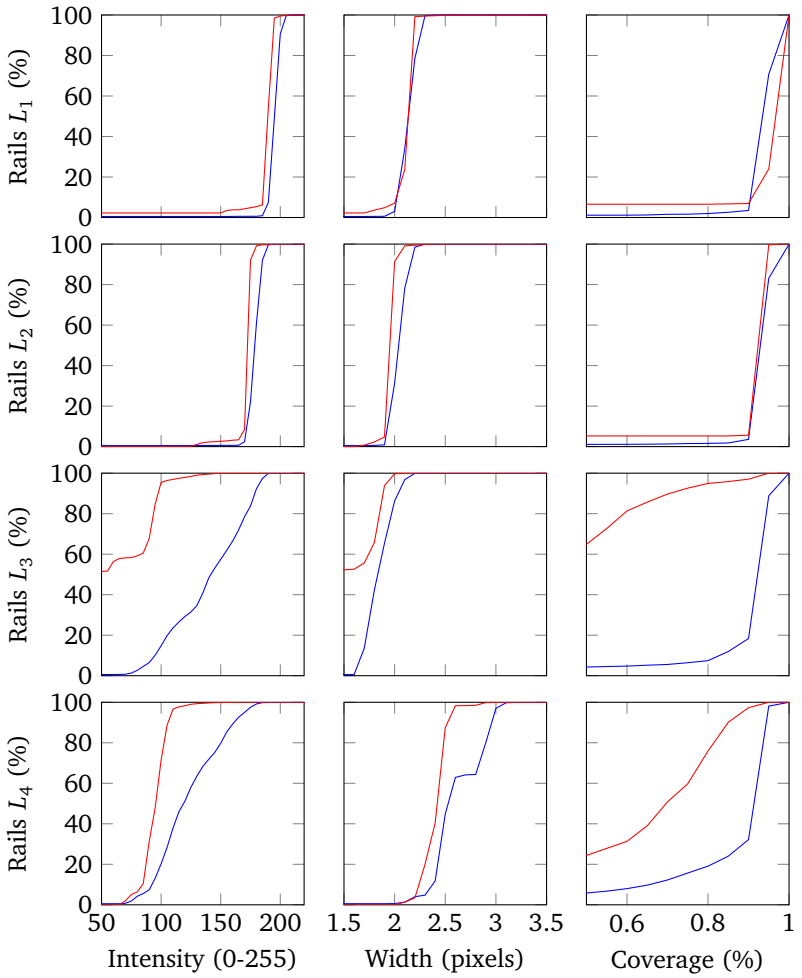

Figure 5.26: Cumulative laser feature histograms, for rails measured from January 2017 to December 2017. In blue, rails for which less than 5% of all sections had missing dimensions associated with a given triangulation unit. In red, the remaining sections.

5.5.2 Possible alarm rules

As stated in Section 5.2.2, alarm events should be triggered whenever serious system problems are detected, so that operators can take immediate action if needed. In the case of sensors that detect image or laser line features, determining which camera-laser emitter pair is at fault is easy in most circumstances. Problem detection should take into account the results for each specific rail, as well as the average of the recent results and the current trends. A possible, tentative, set of rules for problem detection is discussed here:

Each of the features that are described in Section 5.3 should be computed during rail measurement, and an average value should be obtained. Then, average values should first be compared with predefined, per feature maximum and minimum 'hard' limits. If any of the values falls outside these limits, an alarm event should be issued immediately, specifying which feature poses a problem and for which component. Additionally, a moving average of the values should be computed, and compared with a different set of predefined limits. These limits may be defined as percentiles of a set of recent values for the same feature.

Then, linear regression should be applied to the computed values, along with the corresponding values for the last days. The resulting line would then be used to predict when the values would reach their limit.

Whenever the system is calibrated, the stored values should be reset for the purposes of linear regression and moving average computation. Moving averages and linear regression would not be computed until a predefined number of rails had been measured.

Alarm rules can be built from a combination of **limits** and **value computations**. Different limit types are used to compute the acceptable range for a given sensor at a given time, whereas value computations are the strategies to determine the values that will be compared with said range.

The following kinds of limit types have been considered:

- **Hard limits:** Values must fall between two predefined numbers.
- **Percentile limits:** Values must fall between two predefined percentiles of the last N sensor values.
- **Relative limits:** Values must be within a predefined distance of the average of the first N values since the last calibration.

Possible value computation strategies are as follows:

- **Last value:** The last value returned by the sensor is used.
- **Moving average:** The average of the last N values returned by the sensor is used. If less than N values are available, no value is computed.

- **Prediction:** A line is fit to the last N sensor values. The value at a given point in time, extrapolated from this line, is used. If less than N values are available, no value is computed.

As depicted in Algorithm 5.5, a possible set of rules would be as follows. The percentages of missing sections, along with the correlations shown in Figure 5.25, have been taken into account in order to decide where to place the limits, but, for simplicity, they are not directly used here as sensors. Their influence will be discussed in Section 5.5.3.

1. Raise an alert if, for any of the cameras:
   - **Width:**
     - The last value falls outside the range $[1.75, 3]$ (pixels).
     - The average of the last 250 values is below percentile 10 or above percentile 95 of the last 5000 values.
     - The value obtained by fitting a line to the last 250 sensor values, then extrapolating the value it will reach 7 days from now is below 1.75 pixels.
   - **Intensity:**
     - The last value falls outside the range $[50, 240]$.
     - The average of the last 250 values is below percentile 10 or above percentile 90 of the last 5000 values.
     - The value obtained by fitting a line to the last 250 sensor values, then extrapolating the value it will reach 7 days from now is below 50.
   - **Coverage:**
     - The average of the last 250 values is below 90 %.
2. Raise an alert if, for either the lower or the upper laser lines:
   - **Double line:**
     - The average of the last 250 values is above 0.25 mm.
     - The average of the last 50 values, reset after every calibration, is more than 0.05 mm over the average of the first 50 values after the last calibration.

Periodical (or on-demand) system reports should include the values of each feature for all computed rails, the moving average, applicable percentile-based limits, the predicted time at which the values will reach the limits and all issued alarm events.

5.5.2.1 Results

Table 5.2 shows the results of running the tentative rules described above on all the rails that were measured in 2017.

Rules	L_1	L_2	L_3	L_4	Overall
W.1	0.53	0.52	8.26	18.03	6.83
W.2	9.67	7.93	23.62	14.63	13.96
W.3	5.53	10.86	32.00	2.16	12.64
Any width rule	13.83	17.24	45.46	31.41	26.99
I.1	0.53	0.50	7.31	0.46	2.20
I.2	12.50	9.03	37.04	23.61	20.55
I.3	0.80	0.74	12.74	7.19	5.37
Any intensity rule	12.56	9.10	44.94	28.79	23.85
Coverage rule	3.91	2.72	27.23	35.65	17.38
O.1		1.68		16.96	9.32
O.2		7.80		31.38	19.59
Any overlap rule		8.31		33.34	20.83

Table 5.2: Percentage of rails that triggered alert events.

It can be seen that most of the alert events are triggered for the lower triangulation units, L_3 and L_4. This is true for all the considered sensors, including the overlap distance sensors, even though the kinds of issues this sensor is intended to detect (mostly, the need for recalibration) are different from those detected by the other ones (degraded laser emitters and dirty camera lenses, for the most part).

Clearly, the percentage of rails that trigger alarm rules is high. This may be ascribed, in part, to the tentative nature of the rules, but also to the fact that image quality was indeed relatively bad at many times in 2017.

Figure 5.27 depicts the laser intensity for all the rails that were measured during 2017, as seen by one of the cameras (triangulation unit L_4), along with the alert events that were triggered. As shown in Table 5.2, percentile-based triggers (rule I.2, green) are the most common, followed by relative triggers (rule I.3, blue). The line intensity seldom reaches the hard limits (rule I.1, red). Additionally, Rule I.2 events often occur in longer bursts.

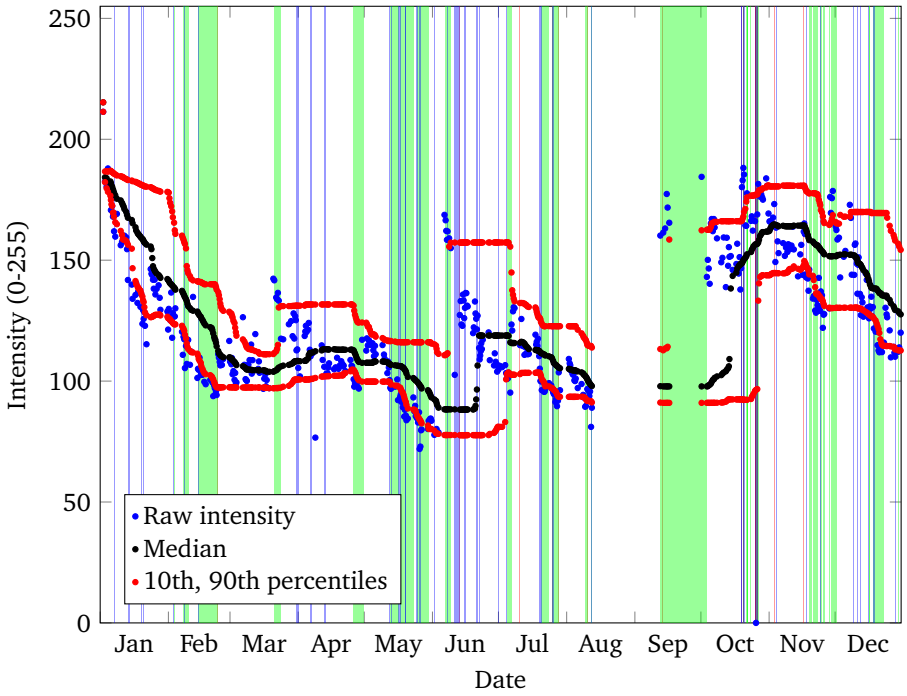

Figure 5.27: Laser intensity and alarm events for L_4, for rails measured from January 2017 to December 2017. Hard limit alarms are shown in red, percentile-based alarms in green and relative alarms in blue.

Algorithm 5.5: Rule system. *intensity*, *width*, *coverage* and *overlap* lists contain the average values of each sensor for all triangulation units and all measured rails. The function described here should be called every time a new rail is measured.

```
 1: function CHECKHEALTH(intensity, width, coverage, overlap)
 2:     for all c ∈ 1..4 do
 3:         if not (1.75 ≤ width_c[last] ≤ 3) then                          ▷ Rule W.1.
 4:             ALERT(width_c, "hardLimit")
 5:         end if
 6:         recent ← width_c[last − 4999..last]
 7:         if not (recent_10% ≤ avg(width_c[last − 249..last]) ≤ recent_95%) then   ▷ Rule W.2.
 8:             ALERT(width_c, "movingAverage")
 9:         end if
10:         line ← LINEARINTERPOLATION(width_c[last − 249..last])
11:         if not (1.75 ≤ line(now + 7 days)) then                         ▷ Rule W.3.
12:             ALERT(width_c, "prediction")
13:         end if
14:         if not (50 ≤ intensity_c[last] ≤ 240) then                      ▷ Rule I.1.
15:             ALERT(intensity_c, "hardLimit")
16:         end if
17:         recent ← intensity_c[last − 4999..last]
18:         if not (recent_10% ≤ avg(intensity_c[last − 249..last]) ≤ recent_90%) then   ▷ Rule I.2.
19:             ALERT(intensity_c, "movingAverage")
20:         end if
21:         line ← LINEARINTERPOLATION(intensity_c[last − 249..last])
22:         if not (50 ≤ line(now + 7 days)) then                           ▷ Rule I.3.
23:             ALERT(intensity_c, "prediction")
24:         end if
25:         if not (0.9 ≤ avg(coverage_c[last − 249..last])) then           ▷ Rule C.1.
26:             ALERT(coverage_c, "movingAverage")
27:         end if
28:     end for
29:     for all d ∈ 1..2 do
30:         if not (avg(overlap_d[last − 249..last]) ≤ 0.25) then            ▷ Rule O.1.
31:             ALERT(overlap_d, "movingAverage")
32:         end if
33:         lastCalib ← LASTCALIBRATION(overlap_d)
34:         avgLastCalib ← avg(overlap_d[lastCalib..lastCalib + 249])
35:         avgValue ← avg(overlap_d[last − 249..last])
36:         len ← count(overlap_d)
37:         if lastCalib + 249 ≤ len and not (avgValue ≤ avgLastCalib + 0.05) then   ▷ Rule O.2.
38:             ALERT(overlap_d, "movingAverageDeviation")
39:         end if
40:     end for
41: end function
```

5.5.3 Report and alarm system

In the initial alarm rule system described in Section 5.5.2, the percentages of missing sections are not directly used. It can also be seen that the number of alarm events, as shown in Section 5.5.2.1, is unacceptably high, at least for the tentative rules that were considered there. Furthermore, said rules raise alarm events for both issues that require immediate action by human operators and issues that can be expected to require action by maintenance crews in the following days. This is not ideal, as human operators and maintenance crews have different concerns and operate on different time frames. Using that rule system as a foundation, along with the ideas discussed in Section 5.2.2, a more complete system can be devised.

First, after each rail has been measured, a rail average of each feature (for every camera, when applicable) should be computed, and a **single rail report** should be issued. This report would include:

- The percentage of sections for which one or more dimensions were lost, for every camera.
- The average laser stripe intensity, width and coverage ratio, for every camera.
- The average distance between laser stripes, for both the lower and the upper cameras.

Along with the report, in the case of laser stripe intensity, width, coverage ratio and distance between laser stripes, a monthly histogram of the values for each feature and triangulation unit should be shown to human operators on demand. The location in the histogram of the last rail to be measured should be marked.

In addition to this report, an **alarm** system should be used to warn operators of likely issues. Alarms should be issued if, for any given rail, at least one of the following is true:

1. The average value of one or more image features (that is, the input features except for the line distance: laser stripe intensity, width or coverage) falls below or above predefined hard limits, such as those described in Section 5.5.2. This rule is intended as a fallback.

   As some features had no hard limits in Section 5.5.2, such limits must be defined. For instance, the (upper) hard limit on line distance can be set to 0.2 mm, in order to detect only situations where the need for calibration is clear.

2. The average value of all of the image features is an outlier among all the values recorded in the last month. Values are considered outliers if they fall below the first quartile of the values recorded minus 1.5 times the interquartile range, or above the third quartile plus 1.5 times the interquartile range; except that

no upper limit is considered for coverage, where higher values are always desirable.

Outlier detection is used here, rather than percentile limits, in order to prevent relatively low or high, but still close to typical, values from triggering alarms. The values of all image features are required to be outliers, rather than just one of them, in order to reduce the rate of false alarms. As an alternative, or in addition to this, a different multiplier, larger than the standard 1.5 typically used in outlier detection, could be considered.

3. The average value of the line distance is above a given hard limit.

4. The percentage of sections with missing dimensions attributed to any triangulation unit exceeds a predefined threshold, tentatively set here to 5% of the number of sections in the rail.

In order to provide as much information as possible about the issue, alarms are associated with a given triangulation unit (or, in the case of the line distance, with either the upper or lower triangulation units).

If two or more of these conditions are met for the same rail and triangulation unit, a single alarm should be issued. Additionally, the issues that cause alarms can be separated into different categories:

- **Missing:** Issues that cause dimension loss and affect any of the image features. That is, the fourth condition above is met, and additionally either the first or the second condition is met*.

- **Not missing:** Issues that do not cause dimension loss and affect any of the image features. That is, the first or second conditions above are met for input features other than the line distance, but the fourth one is not.

- **Calibration:** Calibration issues, where the third alarm condition is met. It must be noted that all the other categories exclude each other, whereas Calibration is independent from them.

- **Unknown:** Issues that cause dimension loss, and cannot be associated with any image feature. That is, all issues that meet the fourth condition but do not meet the first nor the second conditions.

Alarms should also be assigned a criticity value. An alarm should be considered critical if the percentage of sections missing dimensions exceeds a second threshold, tentatively defined here as 20%. Otherwise, the alarms should be considered non-critical.

* The line distance is not considered here because it has little or no impact on missing dimensions, as shown in Section 5.4.1.

Alarms should be shown as part of the rail report, and they should also be notified to human operators whenever they arise. For all alarms, the criticity, characterization and affected triangulation units should be clearly stated.

In addition to the single rail report and alarm system, intended to immediately inform human operators of conditions that require their attention, a **long-term report**, intended for maintenance crews, would be displayed on demand. Such report would show:

- Single rail reports for all measured rails.
- Date plots and histograms for all sensors.
- All the recorded alarms and events.
- An estimate of the time remaining until the values for each of the sensors reach the hard limits, and the estimated value of such sensors after a given amount of time (for example, three days or a week). This is similar to the predictive limits suggested in Section 5.5.2, except that, in this case, predictive information is not presented as an alarm that might be interpreted as requiring immediate action, but as a detailed report for maintenance crews.

5.5.3.1 Results

The alternative alarm rule system described in this section was applied to all the rails that were measured in 2017.

Category	Dims.	Laser	L_1	L_2	L_3	L_4	Average
No issues	0	0	97.67	97.81	82.37	82.77	90.16
Not missing	0	1	1.23	1.11	4.42	5.04	2.95
Unknown	1	0	1.03	1.02	4.90	9.46	4.10
Missing	1	1	0.07	0.06	8.31	2.73	2.79

Table 5.3: Percentages of rails by triggered condition. Here, a value of 1 in the Dims. column means that the third condition was triggered (missing dimensions). A value of 1 in the Laser column means that the first or second conditions were triggered for one of the image features.

Table 5.3 and Table 5.4 show the percentage of rails that meet the discussed conditions. It can be seen that many of the detected issues cause dimension loss, but do not affect image features in such a way that any of the hard limits are reached or all image feature values are considered outliers. This could be improved by ascribing dimension loss to a given image feature if its value is considered an outlier, even if the values of other dimension features are not (and therefore the second alarm condition is not met).

Category	L_1 and L_2	L_3 and L_4	Average
Calibration	2.09	14.32	8.21

Table 5.4: Percentage of rails that triggered calibration-related alert events.

The number of triggered alarms has been greatly reduced with respect to the initial alarm rules described in Section 5.5.2.1, mostly owing to the fact that this approach does not consider long-term issues (such as the prediction rules discussed there), but only those issues that human operators should be able to fix, and probably need to fix, in a short amount of time.

Chapter 6

Conclusions and future work

In this final chapter, the main results of this work, which were presented in the previous chapters, are briefly discussed. Additionally, different possible goals are considered that could guide future work.

6.1 Conclusions

Rail profile measurement systems in rolling mills are crucial inspection equipment to guarantee compliance with rail standards prior to rail track construction. In this work, a profile measurement system has been designed that can measure rail dimensions under degraded conditions, such as the inability to capture a laser stripe from one of the triangulation units.

Additionally, this work contributes a calibration technique which allows for the computation of extrinsic parameters in industrial environments with an accuracy similar to that of well-known approaches that cannot be used effectively in an industrial context. Different methods to calibrate a system for dimensional quality inspection of rails are described. Using the best such method, an accuracy better than ± 0.1 mm is achieved for most of the considered dimensions. Rail standards usually allow for tolerance ranges of approximately 1 mm. As such, the stated accuracy can be considered sufficient for the purpose of dimensional quality inspection for rails. The usage of higher-resolution cameras in the rail profile measurement system would improve this accuracy, if needed.

Different techniques to compute calibration error have been considered and evaluated, and a way to visually estimate the accuracy of a triangulation system, to be used after calibration, has also been shown and discussed.

It has been found that, although specific intrinsic calibration algorithms might make more of a difference for other cameras that produce more distorted images, the differences are negligible in the case of the cameras used in the laboratory environment for the purposes of this work. This may be attributed to the high quality of the employed cameras and lenses.

As for system health evaluation, it has been found that for a system such as the one considered here, degradation of hardware components has a negligible effect

on the quality of the output of the laser triangulation units, and most of the quality losses over a year are caused by dirt accumulation on lenses and laser emitters, due to hostile environmental conditions inherent to industrial environments.

Finally, different sensors have been designed to measure system health, and the inability to compute different dimensions has been associated to issues with specific laser emitters. A correlation has been found between sensor output and these dimension failure issues. These has been used to define a set of rules that can detect and, in some cases, predict, accuracy loss and system failure.

The results of applying these rules can be used to set up a system to warn process computers and operators of the rail rolling mill. Some techniques that can be applied in order to implement a self-aware rail profile measurement system have also been discussed.

6.2 Future work

The work performed here could benefit from rigorous uncertainty quantification, in order to identify which parts of the measurement pipeline affect measurement error the most, and then identify strategies to effectively deal with such measurement error.

As for the calibration procedure, alternative calibration target setups should be considered in order to determine which setup can minimize calibration error. In order to systematically check a range of such setups, the possibility to build synthetic calibration target images has been considered. Alternatively, a different calibration procedure could be devised that would use a target that could be permanently deployed in a structure such as the one considered here. In this way, calibration could be performed periodically with no operator involvement.

Future work should also involve designing more refined autonomic computing features, by changing the behavior of the profile measurement system using the output provided by considering the new proposed rules, in order to minimize the need for maintenance.

References

D. Agrawal, S. Calo, J. Giles, K. W. Lee, and D. Verma. Policy management for networked systems and applications. *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, 2005.

J. Ahlström and B. Karlsson. Fatigue behaviour of rail steel–a comparison between strain and stress controlled loading. *Wear*, 258:1187–1193, 2004. doi: 10.1016/j.wear.2004.03.030.

H. Akanuma. The Significance of Early Bronze Age Iron Objects from Kaman-Kalehöyük, Turkey. *Anatolian Archaeological Studies*, 2008.

AREMA. AREMA Manual for Railway Engineering — Chapter 4: Rails, 2011.

R. Bernhardt, V. Piwek, F. Woletz, K. Jacobs, and W. Leitenberger. Coordinate measuring apparatus having a bridge configuration, 2000. US Patent 6,161,298.

P. Besl and N. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1992.

H. Bessemer. Improvement in the manufacture of iron and steel, 1856. US Patent 16,082.

K. L. Boyer and A. C. Kak. Color-encoded structured light for rapid active ranging. *IEEE transactions on pattern analysis and machine intelligence*, 1987.

D. F. Cannon, K.-O. Edel, S.L. Grassie, and K. Sawley. Rail defects: an overview. *Fatigue & Fracture of Engineering Materials & Structures*, 26:865–886, 2003. doi: 10.1046/j.1460-2695.2003.00693.x.

Centro de formación ArcelorMittal Asturias. *El proceso siderúrgico*. ArcelorMittal, second edition, 2007. Depósito legal AS-96/2007.

A. Chapman and P. Kent. *Robert Hooke and the English Renaissance*. Gracewing, first edition, 2005. ISBN 978-0852445877.

N. Deshpande. *Artificial intelligence*. Technical Publications, second edition, 2008.

E. Deutschl, C. Gasser, A. Niel, and J. Werschonig. Defect detection on rail surfaces by a vision based system. *IEEE Intelligent Vehicles Symposium*, 2004.

A. T. Douglas and H. J. Keen. Digitally compensated hydraulic scale system, 1997. US Patent 5,606,516.

J.-C. Emery. Simplifying the electronic balance load cell. *Sensors Magazine*, 2002.

European Committee for Standardization. EN-13674 — Railway applications - Track - Rail, 2011.

F. A. Firestone. Flaw detecting device and measuring instrument, 1942. US Patent 2,280,226.

A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

J. García-Martín, J. Gómez-Gil, and E. Vásquez-Sánchez. Non-destructive techniques based on eddy current testing. *Sensors*, 2011.

Gosstandart of Russia. GOST R 51685-2000 — Railway rails - General Specifications, 2001.

Greenwood Engineering A/S. MiniProf Digital Profile Measuring: Your way to valid and reliable data. Commercial brochure, 2010. URL `http://bakerailservices.co.uk/files/2010/07/MINIPROF_brochure040314-2-small.pdf`. Retrieved on 2017-07-26.

A. Guttman. R-trees - a dynamic index structure for spatial searching. *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, 1984.

J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

P. Horn. Autonomic computing: IBM's perspective on the state of information technology. Technical report, IBM Research, 2001.

M. C. Huebscher and J. A. McCann. A survey of autonomic computing — degrees, models, and applications. *ACM Computing Surveys (CSUR)*, 2008.

S. Hussmann, B. Hagebeuker, and T. Ringbeck. *A Performance Review of 3D TOF Vision Systems in Comparison to Stereo Vision Systems*. INTECH Open Access Publisher, 2008.

IBM. An architectural blueprint for autonomic computing. Technical report, IBM, 2005.

International Union of Railways. UIC Code 860 — Technical specification for the supply of rails, 2008.

ISO. *ISO 9000:2015: Quality management systems — Fundamentals and vocabulary*. International Organization for Standarization, 2015.

J. O. Kephart. Research challenges of autonomic computing. *Proceedings of the 27th international conference on Software engineering*, 2005.

J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Comp*, 2003.

A. Khalid, M. A. Haye, M. J. Khan, and S. Shamail. Survey of frameworks, architectures and techniques in autonomic computing. *Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on*, 2009.

T. Lau and P. K. L. Shum. Quantitative measurement of the autonomic capabilities of computing systems, 2009. US Patent 7,539,904.

G. Li, C. Wang, J. Liu, and W. Jin. Dynamic rail-wear inspecting system based on machine vision. *IEEE Conference on Industrial Electronics and Applications*, 2007.

Z. Liu, J. Sun, H. Wang, and G. Zhang. Simple and fast rail wear measurement method based on structured light. *Optics and Lasers in Engineering*, 49, 2011.

P. Manso, D. F. García, and R. Usamentiaga. Rail flatness measurement method based on virtual rules. *IEEE Transactions on Industry Applications*, 2017.

H. Men, B. Gebre, and K. Pochiraju. Color point cloud registration with 4D ICP algorithm. *IEEE International Conference on Robotics and Automation*, 2011.

D. A. Menascé and M. N. Bennani. On the use of performance models to design self-managing computer systems. *Proceedings of the Computer Measurement Group*, 2003.

O. Molina. Composition for ultrasonic inspection of objects and method for employing same, 1974. US Patent 3,826,127.

R. L. Moncrief, E. J. Durfey, and M. L. Behensky. Strain gauge pressure-sensitive video game control, 1992. US Patent 5,116,051.

MVTec Software GmbH. HALCON 12.0.2 Solution Guide III-C 3D Vision, 2016. URL `http://download.mvtec.com/halcon-12.0-solution-guide-iii-c-3d-vision.pdf`. Retrieved on 2017-11-14.

M. R. Nami and K. Bertels. A survey of autonomic computing systems. *Third International Conference on Autonomic and Autonomous Systems*, 2007.

A. Nieder. Stereoscopic vision: Solving the correspondence problem. *Current Biology*, 2003.

J. R. O'Leary. A computer simulation of a true position feature pattern gage. *J. Vib., Acoust., Stress, and Reliab*, 1983.

OpenCV. OpenCV 3.3.0 documentation, 2017. URL `http://docs.opencv.org/3.3.0/`. Retrieved on 2017-09-13.

M. Papaelias, C. Roberts, and C. L. Davis. A review on non-destructive evaluation of rails: State-of-the-art and future development. *Proceedings of The Institution of Mechanical Engineers Part F-journal of Rail and Rapid Transit - PROC INST MECH ENG F-J RAIL R*, 222, 12 2008.

M. Parashar and S. Hariri. Autonomic computing: An overview. In J. P. Banâtre, P. Fradet, J. L. Giavitto, and O. Michel, editors, *Unconventional Programming Paradigms. Lecture Notes in Computer Science, vol 3566*. Springer, Berlin, Heidelberg, 2005.

S. Sahay, G. Mohapatra, and G. Totten. Overview of pearlitic rail steel: Accelerated cooling, quenching, microstructure, and mechanical properties. *Journal of ASTM International*, 6, 2009.

G. Sansoni, M. Trebeschi, and F. Docchio. State-of-the-art and applications of 3d imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors*, 2009.

C. Steger. Extracting lines using differential geometry and gaussian smoothing. *International Archives of Photogrammetry and Remote Sensing, Vol. XXXI, Part 3*, 1996.

C. Steger. Unbiased extraction of lines with parabolic and gaussian profiles. *Computer Vision and Image Understanding*, 2013.

A. C. C. Tan, M. Kaphle, and D. Thambiratnam. Structural health monitoring of bridges using acoustic emission technology. *International Conference on Reliability, Maintainability and Safety*, 2009.

V. Tarquinee and R. J. Scovill, Jr. Method and apparatus for the continuous casting of metal, 1955. US Patent 2,698,467.

R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 1987.

K. Tzanakakis. *The Railway Track and Its Long Term Behaviour: A Handbook for a Railway Track of High Quality*. Springer Tracts on Transportation and Traffic. Springer, 2013.

R. Usamentiaga, D. F. García, and J. Molleda. Efficient registration of 2D points to CAD models for real-time applications. *Journal of Real-Time Image Processing*, 2015.

R. Usamentiaga, D. F. García, and F. J. delaCalle. Real-time inspection of long steel products using 3D sensors: calibration and registration. *IEEE Transactions on Industry Applications*, 2016.

M. Vo, Z. Wang, L. Luu, and J. Ma. Advanced geometric camera calibration for machine vision. *Optical Engineering*, 2011.

H. N. G. Wadley and R. Mehrabian. Acoustic emission for materials processing: a review. *Materials Science and Engineering*, 1984.

C. Wang, Z. Ma, Y. Li, J. Zeng, T. Jin, and H. Liu. Distortion calibrating method of measuring rail profile based on local affine invariant feature descriptor. *Measurement*, 2017.

S. R. White, M. Swimmer, E. J. Pring, W. C. Arnold, D. M. Chess, and J. F. Morar. Anatomy of a commercial-grade immune system. Technical report, IBM Research, 1999.

Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

S. Zheng, X. Chai, X. An, and L. Li. Railway track gauge inspection method based on computer vision. *IEEE International Conference on Mechatronics and Automation*, 2012.