SEVENTH FRAMEWORK PROGRAMME

VIT
Vision for Innovative Transport

Project partly funded by the EC
Grant agreement no. *222199*

SP4-Capacities - Research for SMEs

WP4 – REPORT OF THE FEASIBILITY STUDY

Deliverable D4.1

Release date 7 July 2009

23 October 2009 - Reviewed after EU Technical Expert comments

Work package number WP4
Work package title 2D Train Scanning
Activity Type RTD

About the Document

This document reports the results of a feasibility study on 2D reconstruction of the train profile and is *Project Deliverable D4.1*.

The document has been produced by the collaboration of the workpackage WP4, the participants of the workpackage have all duly contributed to the activity of the workpackage and the production of this document and they endorse the final version as the conclusion of the workpackage.

Workpackage leader

Francesca Odone (DISI)

Document authors

Francesca Odone (DISI)
Alberto Lovato (IMA)
Emanuele Trucco (DUN)

Document reviewers

Guido Porta (ILOG)
Michele Molinari (MOL)

Table of contents

INTRODUCTION

This report describes the results of the feasibility study on 2D train scanning, i.e., the main RTD activity of WP4. The document addresses tasks T4.2 and T4.3 and is organized according to the following structure.

Structure of the report

The report is organized with the following structure:

1. Overview
2. Reconstruction of the train profile and automatic load verification
3. Localization and recognition of the ownership code
4. Possible improvements on the present state of the prototypes
5. Bibliography

AUDIENCE

The present deliverable is filed as Confidential, as it contains critical information for the VIT project and also for the Metrocargo system.

Therefore the audience of the document is restricted the project participants --- the SME's who will find the technical details following their user requirements and the RTD performers who will use the present report as a guideline of their research and development activity.

1. Overview

This section reports the advancements of WP4 at the state of month 12, with a reference to tasks as set in the project (Annex 1) and user requirements given by the SME during the first months (deliverable D2.1).

Relevant Workpackage Tasks

Task T4.2: Methods for train profile reconstruction and load verification

Stated purpose:
Study and design of image analysis methods for extracting the profile of the train, in terms of loaded or empty wagons. Study, design and evaluation of global and local vision-based methods for the verification of the train load.

Status Overview:
A pipeline for the extraction of the train profile has been determined. Single modules of the pipeline have been tested on static images, with positive results well below the objectives set by user requirements. The full pipeline is currently being tested on videos of trains entering and leaving stations. Geometrical methods for the verification of the correct placement of containers of wagons, to be used in redundancy of those performed as part of the tasks of WP3, have been studied, developed and evaluated. Experimental analysis shows a sensitivity to noise superior to the objectives set by the SMEs in user requirements. A more detailed report on the status of this task and on the technology that has been used is contained in section 2, “Reconstruction of the train profile and automatic load verification”, of the present document.

Task T4.3: Ownership code identification

Stated purpose:
Implementation of automatic localisation and character recognition methods.

Status Overview:
A custom character localization and recognition system has been built explicitly for the project. A verification strategy has been designed and implemented in order to allow for the possibility of partial OCR failure on a damaged character without compromising the overall performance of the system. The modules perform well both in good and adverse weather conditions, showing a performance well below the measurable objectives set by the SMEs in the user requirements A more detailed report on the status of this task and on the technology that has been used is contained in section 3, “Reconstruction of the train profile and automatic load verification”, of the present document.

User requirements

Deliverable D2.1 detailed a set of user requirements the feasibility of which had to be evaluated within WP4.

Some of the requirements listed in deliverable D2.1 refer to functionalities well within the grasp of current information technology, and require little discussion since their implementation can be carried out using standard methods and an appropriately designed hardware setup. They can be considered as feasible, once a full system integration is performed (an objective out of the scope of VIT). Such requirements are (excerpted from D2.1):

- **Generate a recording of the sequence of containers on the trains, including the information on their size (20’, 30’, 40’, 45’)**

This issue will be addressed in details in the forthcoming sections of the report.

- **Save pictures of the lateral sides of the containers, with readable ownership codes. We suppose this could be done during the train 2D scanning.**

- **Save images of the front and back sides of all containers, with readable ownership codes and sufficient detail to manually detect damages and the presence of the seal on the lock. We suppose this can be done by cameras located in the pairs of columns standing between the train and the buffer bays.**
- **After completion of loading, send to a manned control room images of each container on the train showing ownership code and the two corner fittings of each side. Record and save such images.**

These user requirements can be addressed with a careful positioning of additional cameras on the load/unload area. As suggested in the user requirements, the presence of columns between the train area and the buffer bays may be exploited to this purpose.
Assuming this configuration, images of the back side of the container will be acquired before load (or after unload); viceversa, pictures of the front side will be acquired before unload (or after load). Pictures of the sides will be captured during load or unload operations when the container sides are free from occlusions.
In case this ad hoc installation will not be possible, at the time the full prototype is set up, an alternative solution based on the use of wide range video surveillance camera. Megapixel cameras will be the ideal technical solution, as they allow for a quality high resolution recording in the areas of interest (container ownership codes, and locking). Once the images have been acquired, saving them is a simple task. These issues and requirements have already been addressed in the Technical Specification document issued by the RTD performers (deliverable D2.2).
In conclusion the requirements are clearly feasible, but a specification of the solution depends on the final layout of the prototype plant.

The other requirements listed in D2.1 have been grouped for experimental analysis in two sets, according to the kind of functionality required, and the project task they refer to. We have chosen to assess the feasibility of the required tasks not only theoretically and at the best of our technological knowledge, but by experiment – by building software prototypes and testing them on sample input.

The same image processing techniques (edge and line detection) are involved with **reconstruction of the train profile and automatic load verification (see section 2 of the present document).** So we have grouped under that title the analyses of the following tasks (as in D2.1 – *User requirements* and D2.2 – *Technical Specifications*):

- **Reconstruct the train profile (as a sequence of empty or filled wagons).**
- **Verifying that each container or swap body is correctly loaded, i.e. that all retaining blocks fit correctly the relevant corner fitting and that the container has been fully lowered on the block**
- **Generate a recording of the sequence of containers on the trains, including the information on their size (20', 30', 40', 45')**
- **Reconstruction can be done either whilst the train is coming to a stop in the station (in this case an initial speed of 60 km/h with 1 m/sec2 deceleration should be considered) or when it is standing in the station.**

The remaining tasks concern a single pipeline, **localization and recognition of the ownership code (see section 3 of the present document).** This objective was broken into the following tasks in D2.1:

- **Read and check against the foreseen load plan the ownership codes (alphanumeric strings) painted on the side of each container. The painting may be faded and damaged, per sample pictures supplied by SMEs**
- **Checking the ID and position of load units versus the anticipated load plan**
- **Verifying the correct loading of the containers in the foreseen positions of outgoing trains, checking their ID by 2D train scanning. Due to the high level of accuracy of control on the container location within the Metrocargo plant (practically zero error),**

this verification is actually redundant and it is acceptable that it be done while train is leaving the station. Any mistake thus detected will be dealt with at the following stop of the train.

All above tasks are required to be accomplished under the following constraints:

- **The international limit profile for trains foresees a maximum profile width of 3150 mm. The Italian railways adopt a maximum width of 3200 mm. A safety margin of 150 mm is to be added on each side, that becomes 180 mm in case of soft suspensions. Therefore while train is in motion all equipment must stay 3200/2 + 180 = 1780 mm clear of the geometrical center of the track. As containers are 2438 mm wide, distance from container must be at least 561 mm**
- **The devised hardware setup (including cameras, illuminators, and additional sensors) should not alter the layout of the plant.**
- **Operate in daylight and at night, preferably using its own illumination although artificial lighting can be provided**
- **Work in extreme weather conditions: fog, rain, snow, wind up to 100 km/h, temperatures between -30° and +50°C.**

Such constraints are already covered as hardware requirements. The robustness of the software when working on data taken by compliant hardware in such extreme conditions will be tested on the field in the last months of the project.

Measurable objectives

Deliverable D2.1 – User Requirements contains a list of objectives indicating quantitative, measurable requirements on the performance of tasks in WP4, to be used as a guideline to assess whether a given operation is feasible or not:

- **Maximum error percentage - reconstruction of the train profile: 1 position per train of 22 wagons equals 4.5% over the total number of observed wagons in standard conditions, 2 positions equals 9% in extreme conditions. An error is observed either if an empty slot of a wagon is missed, or a container size is wrongly associated.**

This objective refers to T4.2 the reconstruction of the train profile. Experiments done in order to validate it will be addressed in section 2 of the present document. Obtained results are compatible with the objective.

- **Maximum error percentage on incoming train – automatic code reading: 1 container out of 50 equal 2% in standard conditions, 2 containers equal 4% in extreme conditions. A code is erroneously read if it will not match with the anticipated load plan, leading to a remote human intervention, and the load plan is proven correct. A direct human interventions is admitted if the visible ownership code is damaged to be unreadable (it is not measurable).**
- **Maximum error percentage on outgoing train --- automatic code reading: 1 container out of 50 equal 2% in standard conditions, 2 containers equals 4%in extreme conditions. A code is erroneously read if it will not match with the load plan of the Metrocargo plant, leading to notification to the general control system, to human verification and subsequent action at the following station.**

Those other two objectives refer instead to T4.3, Ownership Code Verification. Thus, experimental tests performed in order to validate the objectives will be addressed in section 3 of the present document. Obtained results are well below the objectives.

2. Reconstruction of the train profile and automatic load verification

In this section we report our experiments in localizing containers along the train profile and verifying their correct positioning on the wagons. We first describe the software modules that are common to both tasks and then report an experimental analysis of their performance for each task.

Edge detection and line detection

Two basic computer vision tools that are used in both modules under analysis in this section are edge and line detection. Line detection is a very natural choice since both tasks addressed here (localizing containers in the image and verifying that they are correctly positioned) are particularly suited to be performed by geometrical analysis: containers are parallelepipeds which can only have a fixed number of standardized sizes, and will appear as rectangles when looked from the side; analysing the sizes and slopes of rectangles present in the image can yield some information about the container positions along the train profile and the correctness of their loading.

Edge detection

Edge detection filters aim at highlighting what we intuitively perceive as edges of objects. The edge detection algorithm we have used is Canny's [TruVer98][Canny86] in the OpenCV implementation, which performs automatically image smoothing by estimating the gradient via Sobel's filter [TruVer98][Sobel68][OpenCV]. We are also in possession of an internal implementation of the classic Canny algorithm, which allows us a more fine-grained control on the smoothing phase and of edge connectivity tracing, but its use has not been necessary for a successful application of the techniques described below.

Line detection

If edge detection reduces an image to a map of its edges, line detection aims at collecting the straight lines present in an image. For images of material objects, that usually amounts to detecting straight edges: this is why line detection is usually applied in cascade to edge detection. In our experiments we have used the very classical Hough line detector [TruVer98][Hough59][DH72], which works by discretizing (at a resolution chosen by the user) the parameter space of lines in polar representation and counting the number of edge points that fit each particular choice of the parameter. Parameter value corresponding to local maxima above a certain threshold of said count are taken as describing lines in the image; such parameters, in the polar representation used, are the slope of the line and its distance from the image origin (0,0).

Train profile reconstruction

Following the solutions detailed in D2.2 (Technical Specifications) an inspection of the Vado Ligure plant was carried out to evaluate cameras positioning. Considering the fact that the plant layout is not yet definitive a solution based on the use of a single still camera seemed to be more appropriate.

We started building a prototype container segmentation module, partly inspired on [KHAVNJ07] and strongly based on the geometrical features of the containers – i.e. their being parallelepipeds with standardized dimensions. These are the assumptions that have been made in our study of the problem:

1. The camera observes the incoming train laterally.
2. The distance between camera and rails, as well as lens characteristics, are known: this allows us to convert container lenghts and heights from pixels to feet and vice-versa.
3. The container baseline is horizontal with respect to camera view.
4. The background removal module works well.
5. The containers we are looking for are wholly included in the image.

Assumptions 1-3 reduce to correct camera calibration; this is well in the reach of current technology, and quite standard practice in setting up a computer vision system. The prototype vision system developed in the last part of the project will follow appropriate specifications on these topics, as outlined in the technical specifications deliverable (cfr D2.2).

Assumption 4 is due to the fact that, even if robust modules for background computation and update are well within the know how of the involved RTD performed, a proper evaluation of such models on video sequences of moving trains has not been done yet. The reason for this is that we are at the moment only in possession of still images, taken at trains standing in a station. In order to properly apply fully automatic background subtraction without any manual intervention one requires building a model of the background; this can be done only when a reasonably long video of the environment is available, so that one can build a convincing model of the background scenario (i.e., a picture of still objects in the scene). This model should be built, or updated, prior train arrival, to minimize the changes due to illumination variations particularly troublesome in outdoor scenarios. This operation can be done real-time, with no delay to the rest of the processing pipeline. From an experimental view-point this would require video acquisitions including a time span (e.g., a minute) before train arrival and the full sequence of the train entering the station, acquired by a still camera. However, Italian legislation requires an explicit written authorization from the national Rail company for filming trains from a standing camera. We have applied for such an authorization months ago, and we are still waiting for it.

In the absence of an appropriate dataset, we have resorted to simulating the effects of background subtraction. Anyway, as far as feasibility goes, this technique has been successfully implemented in order to achieve the same final goal in [KAHVNJ07].

Assumption 5 can be met in two different ways: a) by positioning the camera and choosing a lens such that the field of view is sufficiently wide to include the biggest container the system is supposed to handle b) by mosaicing the movie frames in order to obtain one single, wider picture containing the whole convoy. [KAHVNJ07] successfully uses technique b); moreover, we as a research group are already proficient with mosaicing technology, so we are confident that this assumption can be met independently on the construction specifications of the METROCARGO system. In the technical specifications document (deliverable D2.2) both methods were listed as possibilities; altough we are still unable to gather data corresponding to the different kinds of visual input, we point out that the algorithm tested below is largely independent of the data acquisition method per se, depending instead on the quality of the supplied images.

Figures 1 and *figure 2* show two sample images on which we tested the system, after manual background subtraction.

Figure 1 – Still image of a convoy segment, showing one 20' container fully and other two 20' containers partially. Background has been removed by hand.

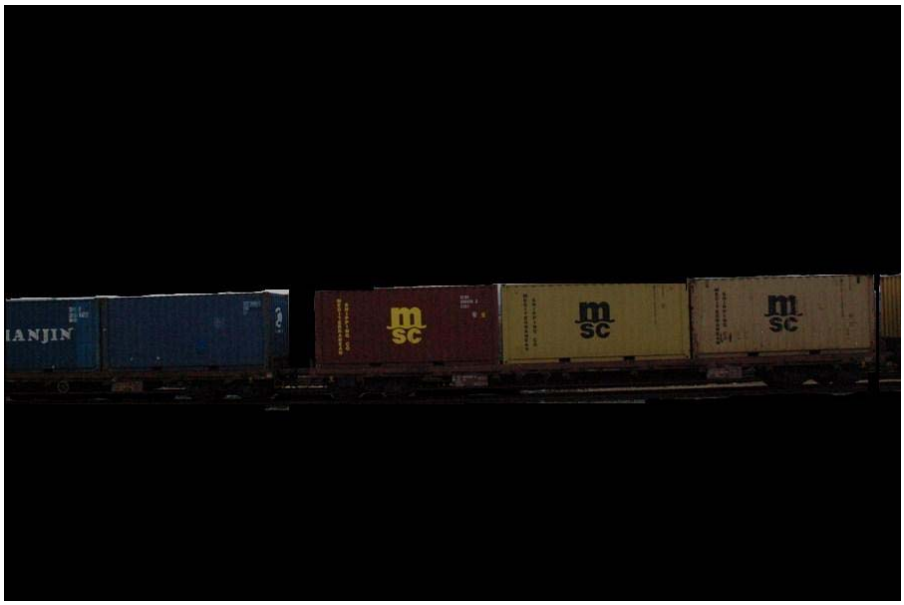

Figure 2 – Still image of a convoy segment, showing four 20' containers fully and a portion of other two. Background again removed by hand. This image is more "difficult" than the previous one, being not horizontally aligned and showing some containers which are adjacent to one another.

On this kind of images, we have performed experiments using a prototypical implementation of the following detection scheme:

1. Extract lines from the image using Canny Edge Detection and Hough Line Detection in cascade
2. Filter out all lines which are neither approximately vertical nor horizontal. "Approximately" here means we have given the algorithm a tolerance of a couple of sessagesimal degrees, in order to compensate for the fact that the image were taken by a hand-held camera.
3. Mark in a separate list all vertical/horizontal lines that come from the edge between the foreground and the subtracted background.
4. Keep in the line list only those lines which are a number of pixel approximately equal to a multiple of the container height (for horizontal lines) or width (for vertical lines) apart from a parallel line on the foreground/background border.
5. Distances between parallel vertical lines are potential container lengths. Distances between parallel horizontal lines are potential container heights. Calculate them and build lists of the line pairs with the respective distance.
6. Consider quartets made up of two parallel horizontal and two parallel vertical lines: they are a superset of all possible rectangles in the image. Keep only those whose aspect ratio and size are compatible (within a small tolerance) with the standard container dimensions.
7. Filter out the rectangles which are intersecting the background
8. For any cluster of heavily intersecting rectangles, choose the representative having the closest aspect ratio to the ideal one.

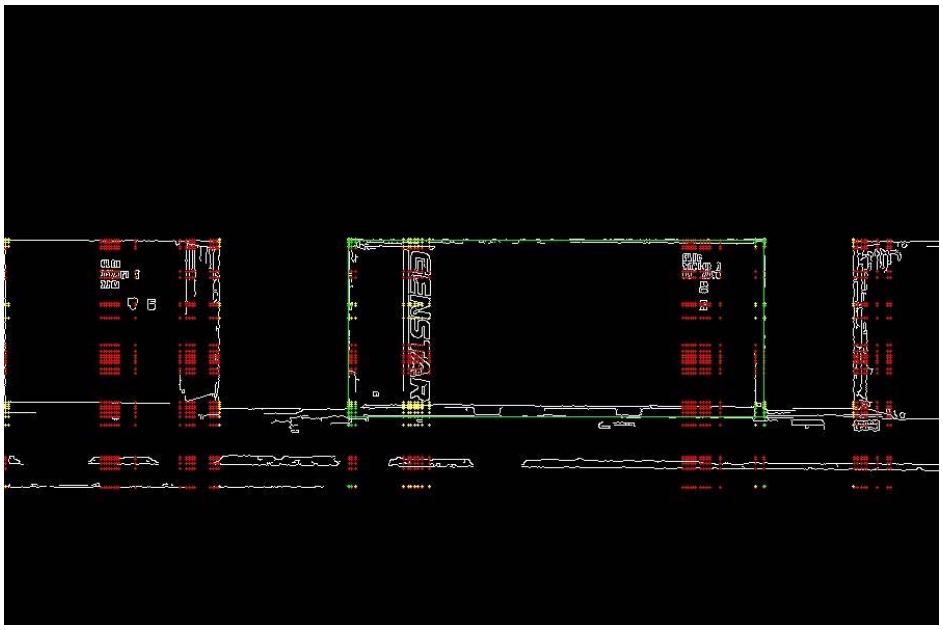

Figure 3 – Output visualization of the container detection algorithm on the image shown in Figure 1. The white-on-black lines are edges as detected by the Canny filter. Red and yellow dots represent discarded line intersections, while the estimated container position is marked by a green rectangle.

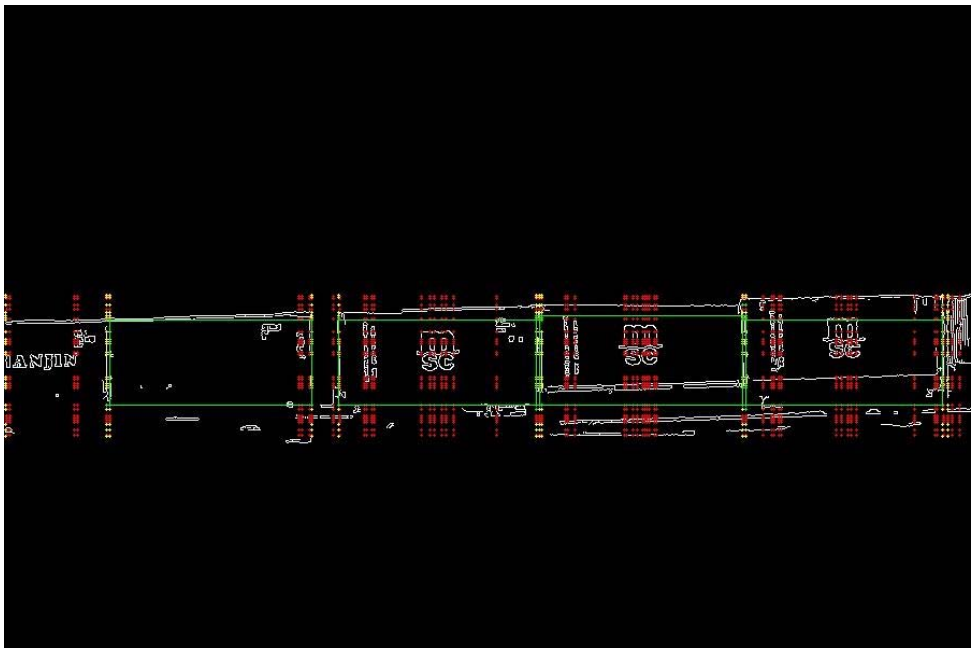

Figure 4 – Visualization of the algorithm's output when applied to the image shown in Figure 2. We can see that the purposefully introduced registration error (some slope of the train with respect to the horizontal axis of the image) impacts on the estimated vertical positions but not on the more important horizontal ones. Moreover, adjacent containers are correctly detected as separate objects.

Red and yellow dots correspond to line intersections (potential vertices) that have been removed in the various filtering stages.

The results of our prototype algorithm look promising, but are subject to the following limitations:

- the background removal phase is only “simulated.” Since the quality of background removal can significatively alter the performance of the algorithm, it is important to replicate the experiments using as input real videos subjected to actual automatic background removal.

- we have been able to acquire images containing only 20 feet containers, and thus could not verify experimentally that our algorithm is reliable in not erroneously localizing two 20 feet containers when a 40 feet container is present (the conditions for a reverse error – two neighbouring 20 feet containers – are instead included in our dataset, and the algorithm succeeds in detecting them as separate entities).

We plan to overcome those limitations in the study with the upcoming data acquisitions, that will allow us to evaluate in an experimental way also the performance of the mosaicing strategy described in D2.2. That same strategy, however, is successfully used in [KAHVNJ07] for substantially the same task (i.e. Reconstruction of the train profile as a sequence of empty or filled wagons via container detection): some evidence in favour of its feasibility is thus present in the literature.

We remark that not much further prior information about the system can be used. For example, according to our SME customers, wagon height is not standardized. So we have to resort to background removal and edge detection to locate the containers.

Automatic load verification

For security reasons, the system will have also to check that containers are placed correctly on wagons, with wagon retaining locks properly inserted in the corner holes of the container.

Manually misplacing containers on wagons in order to replicate the error conditions to be evaluated in laboratory tests is beyond the forces and the power of the RTD performers. Since the preliminary results obtained on simulations are not promising, the SME's agreed to consider this objective as a low priority one. Once the load/unload field tests are completed, more realistic tests will be possibly resumed. We have tried to overcome this lack of examples by using models in laboratory conditions. As a model of the container we used a shoe box of similar aspect ratio. From a camera resting in a fixed position, pictures were taken of the container model both simply laying on the table and tilted in various different ways corresponding to different possible misplacements of the container on the pins. Information about the container orientation is extracted from the image applying in sequence edge and line detection in order to evaluate the slopes of the container sides.

As can be seen in the sample screenshot, lighting effects such as shadows or reverb near the container's edges can easily mask a misplacement, even in a visibility situation that is very favorable to the eye.

We recommend using the 3D capabilities of the close-range vision system performing the hole localization task in WP3 to check the presence of the pin locally, as the result will be more robust to similar illumination problems.

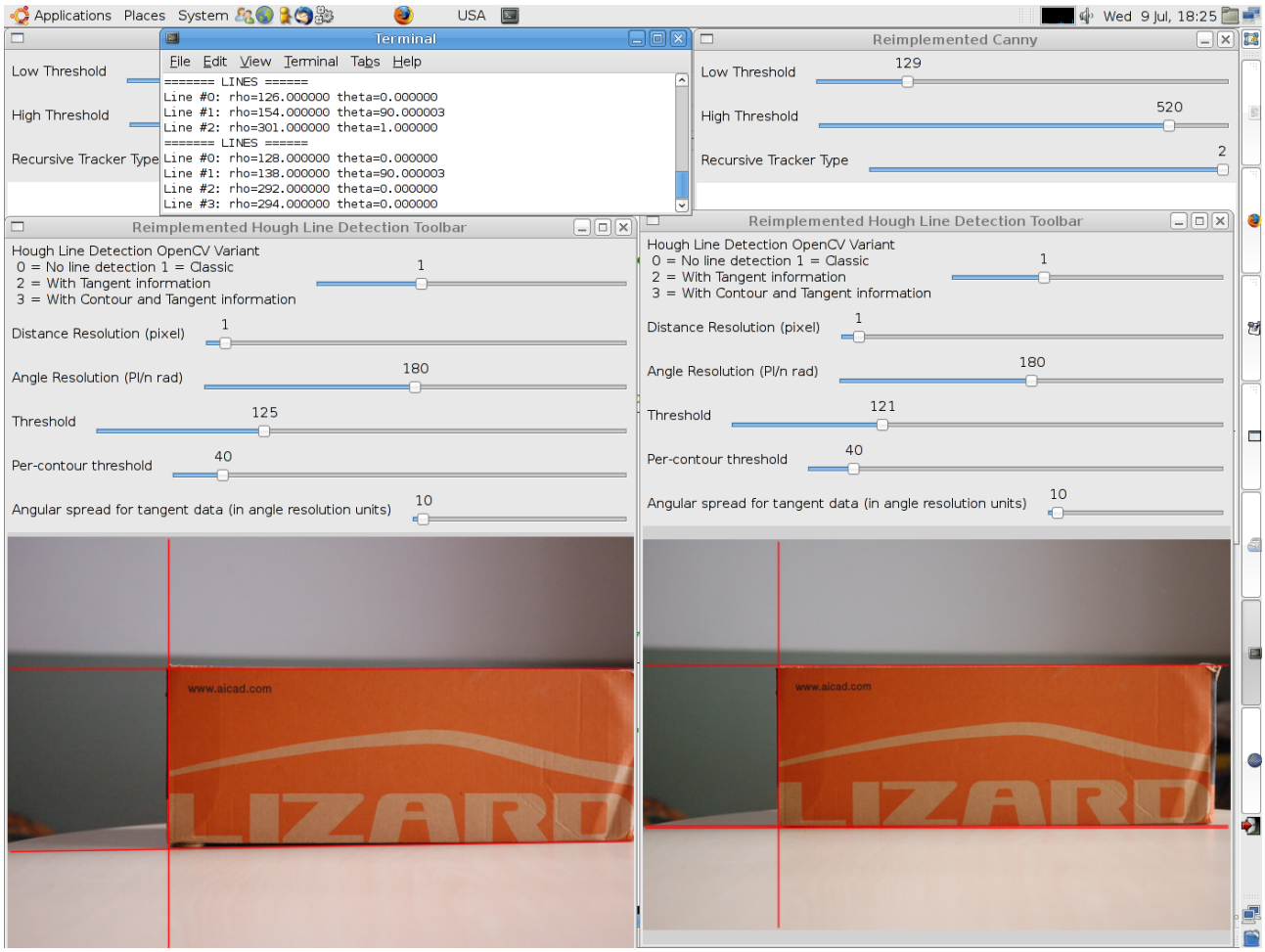

Figure 5 – Screenshot of a test application made for the purpose of checking the robustness of small tilt detection via Hough transform. Here we can see how shadow, light reverb, and other small noise on the object edges can easily mask the detection of such small slopes

Conclusions

By user requirement:

- **Reconstruct the train profile (as a sequence of empty or filled wagons).**
  - Positive results of the experiments for the use of rectangle detection in order to localize containers in images which have undergone effective background removal. More data needed for a more precise evaluation both problems (40' containers, real videos, etc.)
- **Verifying that each container or swap body is correctly loaded, i.e. that all retaining blocks fit correctly the relevant corner fitting and that the container has been fully lowered on the block.**
  - This verification is meant to be redundant with respect to the primary load verification required in WP3. Discouraging results regarding the robustness of hough-transform based methods to illumination noise, even in lighting conditions that would otherwise be considered favourable. We recommend to rely only on the local check for the pin via the equipment designed by WP3.
- **Generate a recording of the sequence of containers on the trains, including the information on their size (20', 30', 40', 45')**
  - The container size should be correctly detected by the container localization module. We have already tested one of the possible ambiguous situations: two adjacent 20' containers, that could potentially lead to a false detection of a single 40' one.

- **Reconstruction can be done either whilst the train is coming to a stop in the station (in this case in initial speed of 60 km/h with 1 m/sec2 deceleration should be considered) or when it is standing in the station**
    - The considered solutions do not depend on which option is chosen. In the framework under observation, choosing one option over the other would only imply minor changes in the input preprocessing (i.e. mosaicing and background subtraction) techniques. As long as the acquisition+preprocessing module is able to comply with the requirements expressed by assumptions 1-5 above, locating containers along the train profile is to be deemed a feasible task, using the algorithm we have described in this section. Some indications on the feasibility of complying with these assumptions are included in the discussion.

By task goal:

- **T4.2 - Study and design of image analysis methods for extracting the profile of the train:**
    - We have a working prototype of a system capable of localising and segmenting containers from images of a train to which the background has been removed. In the remaining months, further data acquisitions will allow us both to extend the system by including an internal background removal module and to test it against a more extensive data set.
- **T4.2 - Study, design and evaluation of global and local vision-based methods for the verification of the train load.**
    - Experimental analysis of global geometrical methods (hough-transform based) in laboratory conditions on a model setup showed that local methods such as those used for the pin localization in WP3 are to be preferred, due to very limited robustness of the former to illumination noise at the precision required

By measurable objective:

The measurable objective set by the SME's for this task in the user requirements (deliverable D2.1) is as follows:

- **Maximum error percentage - reconstruction of the train profile: 1 position per train of 22 wagons equals 4.5% over the total number of observed wagons in standard conditions, 2 positions equals 9% in extreme conditions. An error is observed either if an empty slot of a wagon is missed, or a container size is wrongly associated.**
    - In performed tests, no false detections of containers (the prerequisite for missing a slot) happened, nor any container size was wrongly associated in standard conditions and with the presence of distant or adjacent (and thus possible to mistake for longer ones) 20' containers. The module has yet to be validated in extreme conditions and for 40' and 45' containers, which are less common than 20' ones.

3. Localization and recognition of the ownership code

ISO standard 6346:1995 is concerned with the coding, identification and marking of freight containers; it requests each container to be marked with a unique identification code. One of the targets of WP4 is to assess the feasibility of automatically reading such codes from a video stream of the incoming train.

Container identification codes, as per standard 6346, consist of a total of eleven alphanumeric characters, structured as follows:

- Three latin alphabet uppercase letters, identifying the container's owner as registered by the BIC (Bureau International des Containers).
- One latin alphabet uppercase letter, identifying the equipment type. For the containerization equipment mobilized by METROCARGO, this letter is always 'U'.
- Six arabic digits, constituting the container's serial number as assigned by the owner
- One arabic digit, which is a checksum – calculated following a standard algorithm – of the whole code.

The layout of the code can be both horizontal and vertical, and variables such as font, font size, spacing, colour, the presence of whitespace between different segments of the code, are not fixed. We have counted tens of different layouts just in the quite limited dataset in our possession. Such variability in sizes, spacing, layout and whitespace makes the template-based approach used sometimes for licence plate detection and reading involuted and, what is more, hazardous: the system could easily fail in presence of a code that doesn't follow any of the expected templates. An unconstrained, adaptive approach seems more appropriate to the problem.

Ownership code localization

Figure 6 – Test of the text detector on a sample image. Connected components detected as text are marked in green. The aig carrier logo is filtered out for its size.

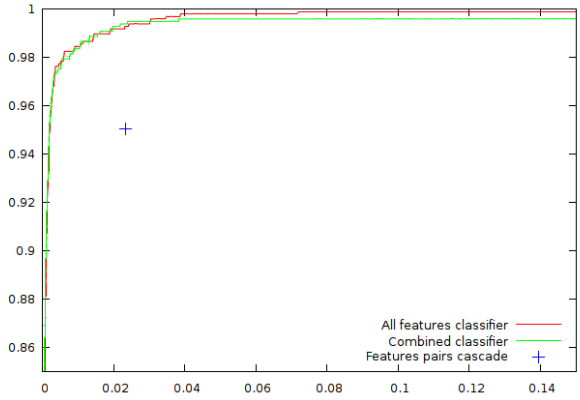

Figure 7 – ROC curves (probability of correctly detecting a character against probability of detecting a nonexistent one) for different set-ups of the classifier, as in article [ZDO09].

Unconstrained text localization and reading is an open problem for the computer vision research community; its more advanced developments were compared at the contests held at the ICDAR conference in 2005 [RRRes05]. The system we have used is described in detail in [ZDO09] and implements a cascade of classifiers on the connected components in the image. The cascade-of-classifiers architecture is a popular way to speed up classification algorithms. It works by breaking up computationally complex classifiers into simpler ones, each of which discards a large part of the “uninteresting” samples it is presented with while keeping almost all the ones that should be detected. This means that only the positives and a small part of the negatives go through all the clas-

sification process, leading to a considerable speed-up in settings where the classifier has to examine a large number of examples, the large majority of which are negative ones. Detection problems very often exhibit this pattern, and thus the cascade architecture, or more in general focus-of-attention methods, are important to build systems with high data throughput.

The pipeline of the text localization module is as follows. First, a black and white version of the image is segmented using Niblack's algorithm [Niblack86]. Niblack's algorithm assign pixels to different classes by comparing them to thresholds calculated as linear functions of the mean and variance of the grey values in a neighbourhood of fixed, user selected size. This local approach offers a better time performance in comparison to global methods such as c-Means or mean-shift. Moreover, results are quite independent from the parameter choice, and sometimes even better than those obtained by more computationally intensive algorithms.

After segmentation, the obtained connected components are first filtered according to their size, which allows for a very quick elimination of some blatant negatives and of some text which is not interesting for our application (see Figure 3.1), then described using a set of twenty relevant features which will be used for classification. For a detailed listing and description of the selected features we refer to [ZDO09] and the references therein; as a broad overview, we mention that they include geometrical features such as aspect ratio, direction variations across the contour, hole number, or stroke width, along with various moment-like descriptors. The descriptors are not computed all at once, but on demand as they are needed for classification. This is what makes the cascade approach in classification computationally favourable. Not all descriptors analyzed for a general-purpose application in [ZDO09] are used in the system described here. We found that some of them became redundant in the specialized container situation after filtering on the connected component's size, meaning that their calculation burdened computationally the module without giving back a significant improvement in speed. Such removed descriptors are: the four contour roughness and grey level homogeneity descriptors, edge symmetry, and the third among the grey level correlation measurements.

The classifier we have implemented combines the speed of a cascade-based architecture with the precision of a monolithic classifier which simultaneously keeps into account all features. This is obtained by a simple hybrid design, which resorts to a monolithic classifier after having discarded the majority of negatives with a cascade-based one. Our specific implementation makes use of a two-layer cascade based on the aspect ratio and edge strength features, which discards between 20% and 30% of the candidate connected components, leading to an overall system which is approximately four times faster than one based on just the monolithic classifier, with a negligible loss in classification performance as can be seen in Figure 3.2. A purely cascade-based classifier, constructed by using at most two features for each stage, would have led to another halving of the computation time, but at some expense of the classification performance, as shown again in the same figure.

The data set used for estimating the performance includes some pictures taken in adverse weather conditions (heavy rain, strong illumination casting shadows, fog), on which the system does not show any significant performance decrease.

A large part of the false negatives (which total about 2% of the characters in test images) are lost in the segmentation phase, due mostly either to paint erasure or blur; thus any effort directed at improving the performance of the character detection module should be focused on the segmenter first.

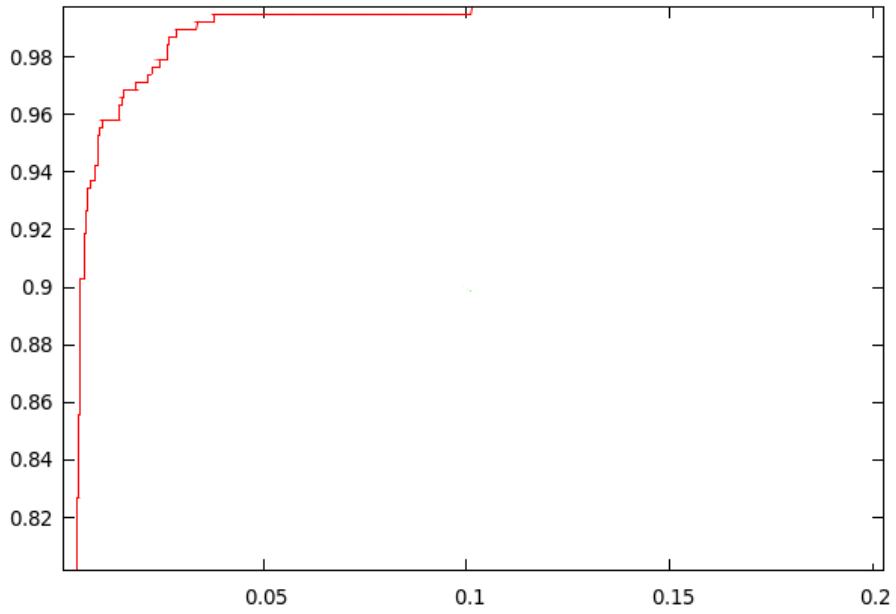

Figure 8 – ROC curve of the part of the cascade of classifiers following the size filters (equivalently, calculated only on connected components of acceptable size). The working point is at the equal error rate of 2%.

Reading functionalities

For optical character recognition, we have used Gaussian kernel SVM classifiers in a one-vs-all scheme. In more detail, for each alphanumeric character a gaussian kernel SVM was used to construct a binary classifier which discriminates between the character under examination and the rest of the alphanumeric alphabet. SVM parameters were chosen via cross-validation individually for each character. In the OVA (one vs all) scheme, those binary classifiers are then combined by applying each of the SVM filters to the classfiand input and choosing as the presumed character the one corresponding to the filter with the highest response.

We remark that, due to limited availability in container images, the number of examples used for training the classifier is not high, and not all characters are equally represented in the dataset (see table 1).

1	2	3	4	5	6	7	8	9	0				
270	398	122	222	72	98	124	150	76	138				
A	B	C	D	E	F	G	H	I	J	K	L		
56	48	112	0	40	4	180	20	14	20	16	66		
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
86	56	4	14	0	30	20	48	198	0	8	14	36	0

Table 1 – Cardinalities of the letter examples in our container dataset.

The container code structure reflects on the population histogram: digits are more common than characters, and the most common character is the letter “U”, which is mandatory in identifier codes or the type of container under study.

These examples were tagged by looking at the binarization of the connected component extracted by the text localization system, i.e. at the same data that would be available to the classifier. Comparing the taggings with the complete source images, one can see that, without the support of context, even a human classifier has difficulties in distinguishing certain letters and digits (“G” and “6”, for example; when binarized, they look very similar).

For this reasons (small cardinality of the letter sample, close similarity – leading to high confusion rates both for humans and automated systems of some letter with some digit) we have decided to merge very similar letter and digits, leaning on the fact that the introduced ambiguity can be removed by the container code syntax. The identifications made were the following: “0”-”O”, “1”-”I”, “2”-”Z”. “6”-”G”, “7”-”T”, “8”-”B”.

This leaves out some characters which remain poorly represented. Due to this problem, the dataset is unsuitable to build a full confusion matrix, which would allow for a deep analysis of the reading errors: more data would be required for this task. However, we can say that in leave-one-out cross-validation, the error rate of the OCR system on this dataset is 3.4%. It is our opinion that a larger, and most of all more balanced, training set would improve this figure.

Post-processing, grouping and alignment

The module described in the above paragraph returns the segmented connected components recognized as alphanumeric characters and their position. Before comparing them to the expected codes as given in the load plan, we need to place them in the correct relative order. Moreover, character detection, although quite reliable, has an estimated probability of failure of 2% for each character. This means that even if the detected characters are correctly ordered, the resulting string could still have gaps at the boundaries as in the middle, so that we cannot be sure of the absolute position of each character in the container code; we should somehow estimate the positions of the gaps in order to get the correct alignment between the read and the expected string.

Characters are of course ordered left to right and top to bottom, as conventional for the western alphabet. However, which spatial ordering should prevail depends on the layout of the code, which has been observed to be both vertical and horizontal. However, since some long substrings (like the serial number in the container code) have never been observed to be broken in multiple lines, what happens is that, organizing the characters in lines and columns via geometrical grouping, vertically oriented codes display more lines than columns, and horizontally oriented ones the opposite. This is precisely the heuristic we have adopted: we group the characters both according to line grouping and column grouping, and then decide whether to read the string horizontally or vertically by checking which group is more numerous.

Grouping is made by considering the upper-left angle coordinates of the connected component's bounding box. Whenever they are approximately aligned, they are considered to be part of the same row/column. Introducing some tolerance on the alignment is necessary in order to compensate for various sources of error:

- discretization
- segmentation
- prospectical effects (common in images taken in an unregistered setting, or due to non-flat surface of the container – see figure 3.3)
- differences in character width (for column groupings; on the contrary, character height seems fixed with the font)
- Dirt or erasures in the paint.

The first two sources of error are listed for completeness, but are usually negligible with respect to the others. In the current implementation, the tolerance parameter is set to the very liberal value of 2/3 of the character width or height. This has not caused unwanted groupings in our tests.

Figure 9 – Examples of horizontally and vertically aligned container identification codes

We have employed the Needleman-Wunsch [NW70] algorithm in order to align the read string with the expected one. Output can then be easily processed by calculating a weighted variant of the Lehvenstein (or “edit,” since it is originally defined [Lehvenstein66] as the minimum number of single-character editing operations that are necessary to transform one string in the other) distance. Gaps introduced in the expected string are interpreted as the result of false positives or spurious characters not belonging to the container code but erroneously brought in by the grouping algorithm. Such kinds of error are deemed negligible and do not contribute to the final computed distance. Gaps in the read string are interpreted as false negatives in the text detection system, and are assigned a score of one half. Proper mismatches are assigned a contribution to the total distance of a value of one. For string matching, we don't require an exact match but admit a tolerance on this calculated score instead, based on the observations made in the following paragraph.

Code verification

The final objective of the code analysis is not in general to get a correct reading of the container code, but more in particular to check this reading against an expected value stored in the load plan. This allows to introduce some fault-tolerance with respect to reading errors at the verification stage. Why this can be very effective and indeed necessary is easily seen intuitively: on relatively long strings, such as container codes, the probability of misclassifying at least one character is significantly higher than the misclassification probability as measured on a single-character basis (indeed, this increase can be shown to amount to a multiplication factor approximately equal to the number of characters in the string, if the single-character misclassification probability is small enough: this leads to high error rates on the strings even when the error rates on the single characters are low).

In order to assess more quantitatively the usefulness of that technique, we have performed statistical experiments by simulating the verification process on a large number (one million) of codes, both expected and seen by the system. The codes were generated randomly and syntactically valid. The reading process was simulated by using an estimated confusion matrix for the OCR system; in lack of sufficient data for an accurate statistical estimate, we have used the confusion matrix of an OCR system created for car license plates and rescaled it as to give a similar (3.4%) global performance while keeping constant the relative weights of the possible errors. The idea is

that every OCR system working on the same alphabet should show more or less the same error syndromes on a standard font.

In table 2 we show the results of this statistical estimate. Notice that by not allowing any mismatches we are going to mistakenly reject about 30% of correct codes. This figure drops immediately to 4% as we allow for one mismatch, with no appreciable increase in the false positive rate.

False positives are estimated by feeding the system randomly generated, syntactically valid container codes different from the ones it expects.

Tolerance	False negatives	False positives
0	2.91E-001	0.00E+000
1	4.17E-002	0.00E+000
2	3.59E-003	0.00E+000
3	2.15E-004	0.00E+000
4	5.00E-006	3.90E-005
5	0.00E+000	9.12E-002

Table 2 – Results of one million of simulated readings of the correct code and of an incorrect code at random. The proportion of false alarms (false negatives) and unseen errors are shown as functions of the tolerance on mismatches.

Experiments

Figure 10 – The single image in the test set on which the verification fails. Remark: in an adjacent frame the same code is present, and correctly detected.

Figure 11 – Another image from the same set, in which the extended code is correctly recognized.

We have tested the full pipeline (from localization to verification) on a set of photographs of a passing train supplied by the SME (illumination and point of view problems render this dataset unsuited for mosaicing, background removal and container localization; nevertheless, they remain very close to real data suitable for processing by both modules, as far as container codes are concerned). This image set has not been used in the training and set-up of the system, and thus the performance evaluation is unbiased.

We chose to set the mismatch threshold at three, according to table 2.

In order to maximise the possibility of a false negative we required the system to verify every (with a container code sized font) string of text present in the images, were it only a sub string of the container code or even a superset (other indications can be present which are not strictly part of the container unique identifier, such as the type indication “22G1” in figure 3.4). The system successfully detected the text in every single frame where it was present. On these 162 images with associated text, the verification scheme failed just once. This indicates a good performance of the prototype system.

Conclusions

The user requirements addressed in this section were the following:

- **Read and check against the foreseen load plan the ownership codes (alphanumeric strings) painted on the side of each container. The painting may be faded and damaged, per sample pictures supplied by SMEs**
- **Checking the ID and position of load units versus the anticipated load plan**
- **Verifying the correct loading of the containers in the foreseen positions, checking their ID**
- **Verifying the correct loading of the containers in the foreseen positions of outgoing trains, checking their ID by 2D train scanning. Due to the high level of accuracy of control on the container location within the Metrocargo plant (practically zero error), this verification is actually redundant and it is acceptable that it be done while train is leaving the station. Any mistake thus detected will be dealt with at the following stop of the train.**

- All four actually amount to being able to perform container code verification, which in turn is a quite complex task composed of many steps, which we have analyzed both in the state of the art document and in section 3 of the present one. Implementing a prototype such a system, we have actually carried out the stated purpose of **task T4.3 - Ownership code identification** as per Annex 1, that is “implementation of automatic localisation and character recognition methods.”
- Our evaluation on the feasibility of a system adhering to the requirements above is positive: our pre-prototype, although leaving room for improvement, performs satisfyingly well on both the test images given by the SMEs and additional images taken in unfavorable weather conditions. Indeed, we recall that the quantitative objectives set by the SME's are:

o **Maximum error percentage on incoming train – automatic code reading: 1 container out of 50 equal 2% in standard conditions, 2 containers equal 4% in extreme conditions. A code is erroneously read if it will not match with the anticipated load plan, leading to a remote human intervention, and the load plan is proven correct. A direct human interventions is admitted if the visible ownership code is damaged to be unreadable (it is not measurable).**

o **Maximum error percentage on outgoing train --- automatic code reading: 1 container out of 50 equal 2% in standard conditions, 2 containers equals 4%in extreme conditions. A code is erroneously read if it will not match with the load plan of the Metrocargo plant, leading to notification to the general control system, to human verification and subsequent action at the following station.**

- The module, as well as the requirements, is symmetric – it makes no difference between incoming and outgoing trains. The single localization and recognition modules have been trained and validated of a set of 237 container images acquired by researchers in various venues, under variable illumination, weather conditions, and with different acquisition devices (photo-cameras / video-cameras of different resolutions). The final test of the full validation procedure was performed on a set of 162 images supplied by the SME's for an unbiased evaluation. Such images did not participate in any way to the learning process, In this final test, we obtained an error rate of 0.06%, well below the requirements set by the SME's.

4. Possible improvements on the present state of the prototypes

- The two modules described above should ideally interact for optimum performance. Container localization can significantly speed up text localization by passing only the relevant areas to the code verification system, largely decreasing the number of connected components to be processed and therefore computation time.
- The container codes constitute also a double check on the detected container size, allowing for a correction of potential detection errors (two 20' containers detected as a single 40' one or vice-versa).
- The aligning algorithms could be used to detect probable false negatives of the text localization (Needleman-Wunsch), and to figure out their location. A prototype of this subsystem can be seen at work in the sample output shown in figure 3.4. Yellow boxes are estimated letter bounding boxes based on the position of the others and string alignment. Since however the OCR system needs a binarized image to work, and most false negatives are due to segmentation problems, how to effectively binarize the recovered regions remains an open problem.
- The container check digit is presently unused in the verification scheme. This because of the relatively high probability of not detecting some character, and thus being unable to even check the code. But making use of this information at least whenever possible should probably increase the performance.
- Many algorithms are data-driven. Retraining them on datasets which are larger and/or more tuned to the data the system will actually process leads to an almost guaranteed improvement.
- The use of redundant information (the same container code will be present in more than one frame) from a video capture of the incoming train could furtherly reduce the error rate.

5. Bibliography

[Canny86] Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

[Hough59] P.V.C. Hough, Machine Analysis of Bubble Chamber Pictures, Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

[DH72] Duda, R. O. and P. E. Hart, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, Vol. 15, pp. 11–15 (January, 1972)

[KAHVNJ07] A. Kumar, N. Ahuja, J.M. Hart, U.K. Visesh, P.J. Narayana, C.V. Jawahar, A Vision System for Monitoring Intermodal Freight Trains, in Proceedings of the IEEE Workshop on Applications of Computer Vision, 2007

[Lehvenstein66] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10 (1966):707–710.

[Niblack86] W. Niblack, An Introduction to Digital Image Processing, pp. 115-116, Prentice Hall, 1986.

[NW70] Needleman SB, Wunsch CD. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". J Mol Biol 48 (3): 443–53. doi:10.1016/0022-2836(70)90057-4. PMID 5420325. http://linkinghub.elsevier.com/retrieve/pii/0022-2836(70)90057-4.

[OpenCV] http://opencvlibrary.sourceforge.net/

[RRRes05] S.M. Lucas, ICDAR 2005 Text Locating Competition Results, Proceedings of the 2005 Eight International Conference on Document Analysis and Recognition, 2005

[Sobel] Sobel, I., Feldman,G., "A 3x3 Isotropic Gradient Operator for Image Processing", presented at a talk at the Stanford Artificial Project in 1968, unpublished but often cited, orig. in Pattern Classification and Scene Analysis, Duda,R. and Hart,P., John Wiley and Sons,'73, pp271-2

[TruVer98] E. Trucco and A. Verri. Introductory techniques to 3D Computer Vision. Prentice Hall, 1998.

[ZDO09] L. Zini, A. Destrero and F. Odone. A classification architecture based on connected components for the detection of text in unconstrained environments. To appear in Proceedings of the 6th IEEE International Conference on Advanced Video and Signal Based Surveillance.