20th RoboCup International Symposium, Leipzig, Germany, July 2016.

Robust Multi-Modal Detection of Industrial Signal Light Towers

Victor Mataré, Tim Niemueller, and Gerhard Lakemeyer

Knowledge-Based Systems Group, RWTH Aachen University, Germany
{matare, niemueller, gerhard}@kbsg.rwth-aachen.de

Abstract. Introducing robots to provide flexible logistics in a smart factory and cohabitation of robot workers and human operators will require robots to recognize and interpret the same cues in the environment as humans do. In this paper, we describe a novel method to detect machine light signal towers as one such cue that are frequently seen on production machines. It uses color information to determine basic regions of interest and applies a number of spatial constraints to make it robust against many common disturbances. As an option, the algorithm can use laser data for machine-specific reduction of the search space for a speed up by an order of magnitude providing fast, accurate, and robust detection. It recognizes the respective activation states and even blinking lights.

1 Introduction

Industrial manufacturing is expected to change considerably in the near future – a paradigm shift often called Industry 4.0 [1]. Part of this vision are *smart factories*, context-aware facilities that can take into account information like object positions or machine status [2]. They provide *manufacturing services* that can be combined efficiently in (almost) arbitrary ways. This challenge is modeled by the *RoboCup Logistics League (RCLL)* [3].

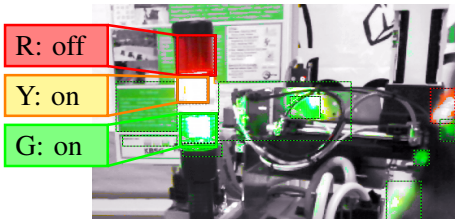


Fig. 1. Illustration of recognition and noise.

While some factories will be designed according to this vision with networked machinery, even more existing facilities will be incrementally upgraded for economic reasons, requiring the robots to adapt to existing machines, and to work safely alongside humans [4,5]. The light signals used in the RCLL are industry-standard parts[1] that are often used to indicate a machine's status, e.g. when it is about to run out of material, or whether it is currently safe for a human to perform certain operations. Being able to visually recognize these is important even in the presence of a network to communicate that very information, for

[1] Similar to http://www.werma.com/en/s_c1006i2580/K37_cable_24VAC/DC_GN/YE/RD/69811075.html

example to prevent misunderstandings between humans and robots in case of a signal or network failure.

In this paper, we describe a novel method that uses a coarse (yet expressive and very efficient) color model to search for relevant regions of interest (ROI) of the light colors red, yellow, and green. These regions are then filtered by a number of spatial constraints to eliminate typical false positives like colored reflections on metal parts of the machine. A machine-specific laser-based detection of the signal tower can be used to reduce the image search space considerably, providing an order of magnitude speed-up while increasing reliability. Eventually, the detected ROIs for the three colors are analyzed for their activation state (cf. Fig. 1) and for temporal relations to detect blinking lights.

In the following Sect. 2 we briefly describe the RCLL and the problem of light signal tower detection. In Sect. 3 we highlight some related work before describing the method in detail in Sect. 4. We provide evaluation results in Sect. 5 before we conclude in Sect. 6.

2 RoboCup Logistics League and Signal Light Towers

RoboCup [6] is an international initiative to foster research in the field of robotics and artificial intelligence. Besides robotic soccer, RoboCup also features application-oriented leagues which serve as common testbeds to compare research results. Among these, the industry-oriented RoboCup Logistics League[2] (RCLL) tackles the problem of production logistics in a smart factory. Groups of three robots have to plan, execute, and optimize the material flow and deliver products according to dynamic orders in a simplified factory. The challenge consists of creating and adjusting a production plan and coordinating the group [3].

A game is split into two major phases. In the *exploration phase*, the robots must determine the positions of machines assigned to their team and recognize and report a combination of marker and light signal state. During the *production phase*, the robots must transport workpieces to create final products according to dynamic order schedules which are announced to the robots only at run-time, while the machines indicate their status with light signals.

Machines in the RCLL are represented by Festo's Modular Production System (MPS) stations, each equipped with a red/yellow/green *signal light tower*. For example, in Fig. 2 a robot approaches a ring station, where the signal tower is on the front left corner of the station.

Fig. 2. Robot approaching a ring station.

The distinctive feature of this vision problem is the presence of *active light sources* with an extreme

[2] RoboCup Logistics website: `http://www.robocup-logistics.org`

variation in brightness which far exceeds the sensitivity range of our consumer-grade cameras.

To be able to detect blinking states, we have to recognize both lit and unlit signals, but depending on ambient light, unlit signals may be captured as almost all black while lit signals are captured as mostly white (cf. Fig. 3). Another problem is the fact that the individual red/yellow/green segments are not optically separated internally, for example, a lit red segment will always make parts of an unlit yellow segment appear red. In combination with extensive and *unpredictable background clutter* (cf. Fig. 3) coming from colorful reflections on shiny machine parts, colorfully dressed spectators and other objects, false positives become a major problem. Since individual segments are made of a transparent material with a fluted surface, the use of many light emitting sensors like a Kinect is infeasible. The use of stereo cameras is made difficult since the amount of textures is low if the region of a color is mostly a bright spot if the light is turned on, or the remainder of the image too dark if tuned down.

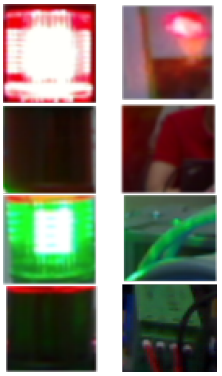


Fig. 3. Actual light signals vs. environment clutter.

3 Related Work

Automatic detection of roadside traffic lights is a related field in particular for autonomous driving. Ziegler et al. describe the challenges posed by a long real-world overland journey [7] under urban and rural conditions at daytime. While it is in principle possible to work around the whole issue by broadcasting traffic signal states over radio, this would require major infrastructure investments [8].

A common practice is to build a database containing features of known intersections to assist locating a traffic signal within a camera image [7,8]. The required data are gathered on a special mapping run of the routes. Fairfield and Urmson generate a detailed prior map that contains a global 3D pose estimate of every traffic signal [8]. Ziegler et al. create a manually labeled 2D visual feature database [7]. During autonomous driving, these hints are then used to limit the search space for the classifier that detects the red, yellow and green lights.

Such approaches do not cover some of the typical problems outlined in Sect. 2 and do not use a second sensor that allows to reduce the problem space.

Another approach in the RCLL has been to reduce camera exposure and contrast until only lit signals would create a saturated output [9]. A drawback of this approach is that this makes the camera unusable for other tasks.

Color detection has been a long-standing issue in RoboCup. In other leagues like the Standard Platform League, lookup tables were sufficient while constant lighting was provided [10]. These methods generally cannot capture the dynamic range with active light sources. Edge and color segmentation have been used to detect vertically stacked color-coded landmarks [11]. While somewhat similar in shape, they did not change during the game and had no temporal dependencies.

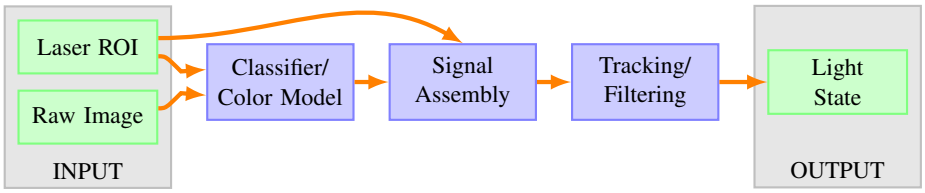


Fig. 4. A model of the processing pipeline.

4 Multi-modal Light Signal Detection

Image processing is performed as a sequence of operations forming a processing pipeline that is depicted in Fig. 4. A *classifier* takes an input image and determines *regions of interest (ROI)* by detecting colors along a grid with pixels of relevant colors according to *similarity color models*. An assembly stage combines ROIs of different colors according to some spatial constraints. Additionally, based on the detection of the flat side panel of the MPS (cf. Fig. 2) by means of a 2D laser scanner, the ROIs can be further constrained by an estimate of the expected position within the image. This combination of different sensors makes this a multi-modal approach which significantly reduces the search space and the chance of false positives. Distance-based tracking ensures that consecutive frames are accepted for small movements. A brightness classifier detects lit/unlit signal segments in the determined ROIs and temporal aggregation is performed to detect blinking signals.

In the following we will detail the major components of the pipeline which has been implemented using the computer vision framework in Fawkes [12].

4.1 Color model

The color model is responsible for deciding whether an input color matches a certain reference color. The used color model has been ported from the VLC video player.[3] It works directly with the YUV colorspace that is produced natively by most webcams, thus eliminating colorspace conversion. In the YUV colorspace, the *luminance* (roughly conforms to the concept of brightness) information is encoded entirely in the Y dimension, while the color value (*chrominance*) is a 2D vector in the UV plane. The saturation of a color then corresponds to the length of the UV vector.

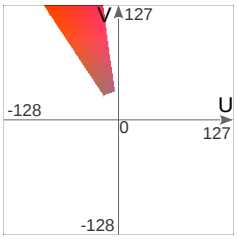


Fig. 5. Sector of the UV plane recognized by the color model.

Normalizing the two color vectors by their saturation and computing the length of the difference vector then gives a reasonable similarity measure: $\delta_{UV} = | \, |\mathbf{r}| \cdot \mathbf{c} - |\mathbf{c}| \cdot \mathbf{r} \, |$, where $\mathbf{r} = (u_r, v_r)^T$ is the reference color, $\mathbf{c} = (u_c, v_c)^T$ is the

[3] Based on VLC's (`http://www.videolan.org`) color threshold filter (colorthres.c).

input color, and δ_{UV} is the scalar color difference. Specifying a threshold on δ_{UV} then allows us to decide whether some pixel from the camera image matches a given color within a certain tolerance. Along with a threshold on $|\mathbf{c}|$ and on δ_Y, such a color model describes a subset of the UV space (similar to Fig. 5) that extends through a portion of the Y dimension. Multiple such color models can be combined into a multi-color model that contains all shades we expect to see e.g. in the red light in a signal tower.

4.2 Classifier

A classifier takes an input image and outputs regions of interest. The color classifier used in this work takes a color model that ascribes a principal color to a pixel color and a scanline grid. The classifier then analyzes each crossing of the grid. If the pixel is found to belong to a known color class, it considers the direct 5×5 neighborhood. Only if a sufficient number of neighboring pixels are assigned to the same color class, the pixel is considered as a positive match. Areas with a sufficient number of similarly colored points result in a ROI. A post-processing step merges overlapping or adjacent ROIs of the same color.

Algorithm 1 Detect a signal tower based on ROIs returned by the classifiers.

Input: R_1, R_0: sets of red on/off ROIs, G_1, G_0: sets of green on/off ROIs, S: the set of previously detected signals
Output: $S \cup T$ where T are detections in current image

$l \leftarrow$ GET_LASER_ROI(); $T \leftarrow \varnothing$
for all $(R, G) \in \{R_0, R_1\} \times \{G_0, G_1\}$ **do**
  $T \leftarrow T \cup$ CREATE_LASER_SIGNAL(R, G, l)
end for
if $T \neq \varnothing$ **then**
  $T \leftarrow \{\arg\max_{\mathbf{t} \in T}$ MATCH_QUALITY$(\mathbf{t}, l)\}$
else
  for all $(r, g) \in R_0 \cup R_1 \times G_0 \cup G_1$ **do**
    $T \leftarrow T \cup$ RED_GREEN_MATCH(r, g)
  end for
end if
if $S = \varnothing$ **then return** T **end if**
for all $\mathbf{t} \in T$ **do**
  $\mathbf{s}, dist \leftarrow$ CLOSEST_MATCH_BY_DISTANCE(S, t)
  if $dist \leq cfg_{max\_jitter}$ **then**
    UPDATE_STATE$(\mathbf{s}, \mathbf{t})$
  else
    $S \leftarrow S \cup t$
  end if
end for
return S

4.3 Signal Assembly

In the signal assembly, we compose a signal of the ROIs denoting enabled or disabled green (G_1 and G_0) and red (R_1 and R_0) signal lights which have been determined by the classifier described above. Algorithm 1 depicts the overall approach: first, it is tried to determine if ROIs can be found that fit into a laser-based ROI (ll. 1–4). If this succeeds, only the best matching ROI combination is kept (ll. 5–6), otherwise a full search on the image is performed (ll. 7–11). If no previous detections exist the algorithm returns the detected signals (l. 12). For the remaining candidates, a distance-based tracking is performed (ll. 13–20). States of previous detections are updated if a new detection is spatially close (ll. 14–17) or just added otherwise (l. 18).

Red/Green Matching A crucial part is the matching of red and green ROIs that are spatially related such that they can represent a light signal. The input ROIs can be of the full image, or constrained to a laser-based ROI (see next section). We limit the search for the signal to red and green ROIs since the yellow light may appear to change color if the lights above or below are lit. Depending on the environment—which might contain arbitrary colorful objects that match the reference colors—the color classifier can return any number of rectangular ROIs, some of which may be part of the signal we are looking for. Algorithm 2 shows the procedure. First, GEOM_OK checks the width and vertical position of the green ROI, and the horizontal alignment of both ROIs:

function GEOM_OK(r, g)
  return whether $\text{width}(g) \leq cfg_{max\_width} \wedge \text{top}(g) > cfg_{green\_horizon}$
    $\wedge\ \text{center}_x(g)$ - $\text{center}_\text{x}(r) \leq cfg_{x\_align\_threshold}$
end function

Any (r, g) pair that does not satisfy this constraint cannot possibly be part of one signal tower, so it is skipped (ll. 2 and 18). A pair that passes is then checked for a special case that can occur due to the extreme brightness of the red and green lights (ll. 3 and 4). The used webcams have an acrylic lens cover that easily gathers a slight haze from dust and wiped-off fingerprints, often causing lit signals to create a colored bloom around the actual light source. The result is a ROI that does contain the signal light, but which is overly large. Whether a ROI ρ_1 is affected by bloom is determined in relation to another ROI ρ_2:

function FIX_BLOOM(ρ_1, ρ_2)
  if [$\text{width-ratio}(\rho_1, \rho_2) > cfg_{max\_width\_ratio}$
    $\wedge\ \text{aspect-ratio}(\rho_1) \leq cfg_{max\_aspect\_ratio}$
    $\wedge\ 0 < \text{vspace}(\rho_1, \rho_2) < 1.5 \cdot \text{height}(\rho_2)$] **then**
    $\text{left}(\rho_1) \leftarrow \text{left}(\rho_2)$
    $\text{width}(\rho_1) \leftarrow \text{width}(\rho_2)$
    $\text{bottom}(\rho_1) := \text{top}(\rho_2) - \text{height}(\rho_2)$
  end if
end function

If bloom is detected, the geometry of the ROI that is likely not or less-affected by bloom is used to improve the geometry. After this, another constraint tests if the vertical space between r and g is sufficient to fit a similarly-sized yellow ROI in between (VSPACE-OK). If this constraint is violated, the (r, g) pair is skipped. Otherwise the two ROIs are aligned well enough horizontally and a similarly-sized gap for a yellow ROI exists in between. If these are still too dissimilar in width (l. 6), the width of both is set to the mean width while preserving the center position (l. 7–9). If a pair of red and green ROIs ran through this process, we assume both must be part of the same signal tower, and generate a yellow ROI y that fits in between (l. 11–14).

Laser-assisted ROI Pre-processing If the position of the MPS table could be detected with the 2D laser scanner, a bounding box can be estimated in which the colored ROIs are to be expected (cf. pink box in Fig. 6). We call this rectangular region the *laser ROI* or l. Within l, we can expect to find (almost) no clutter, which allows us make additional assumptions, as described in Algorithm 3. For example, we can now handle overexposure (Fig. 6) by simply merging the broken-down red or green ROIs into one (Lines 2 and 3). If the red or green light is switched off, large parts of it may appear in a very dark shade

Algorithm 2 Return ROI tuple (r, y, g) if r and g fit all constraints

```
 1: function RED_GREEN_MATCH(r, g)
 2:     if r ≠ ∅ ∧ g ≠ ∅ ∧ GEOM_OK(r, g) then
 3:         FIX_BLOOM(r, g)
 4:         FIX_BLOOM(g, r)
 5:         if VSPACE-OK(r, g) then
 6:             if ¬(1/cfg_max_width_ratio ≤ width-ratio(r, g) ≤ cfg_max_width_ratio) then
 7:                 δ_w ← width(g) − width(r)
 8:                 left(g) ← left(g) + δ_w/2
 9:                 width(g) := width(g) + δ_w/2
10:             end if
11:             left(y) ← mean(left(r), left(g))
12:             width(y) ← mean(width(r), width(g))
13:             height(y) ← mean(height(r), height(g))
14:             top(y) ← bottom(r) + 1/2(top(g) − bottom(r) − height(y))
15:             return (r, y, g)
16:         end if
17:     end if
18:     return (∅)
19: end function
```

Terminology: left(ρ) denotes the left border of a ROI ρ, so right(ρ) = left(ρ) + width(ρ), top(ρ) denotes the top border of ρ, bottom(ρ) = top(ρ) + height(ρ); vspace(ρ_1, ρ_2) is the amount of vertical space between ρ_1 and ρ_2: top(ρ_2) − bottom(ρ_2), and mean(a, b) is the arithmetic mean value of a and b. The function aspect-ratio(ρ) returns width(ρ)/height(ρ) if width(ρ) > height(ρ), and height(ρ)/width(ρ) otherwise, so that for any ρ, aspect-ratio(ρ) ≥ 1. width-ratio(ρ_1, ρ_2) simply returns width(ρ_1)/width(ρ_2), so it can be used to determine which one is wider than the other.

Algorithm 3 Generate a ROI tuple (r, y, g) from R and G that lies within l.

```
 1: function CREATE_LASER_SIGNAL(R, G, l)
 2:     r_m ← [⋃ R] ∩ l
 3:     g_m ← [⋃ G] ∩ l
 4:     if r_m ≠ ∅ ∧ [b_r ← CLASSIFY_BLACK(above(r_m))] ≠ ∅ then
 5:         δ_y ← top(r_m) − bottom(b_r)
 6:         if −δ_y > height(r_m) then
 7:             δ_y ← −height(r_m)
 8:         end if
 9:         top(r_m) ← bottom(b_r)
10:         height(r_m) ← height(r_m) + δ_y
11:     end if
12:     if g_m ≠ ∅ ∧ [b_g ← CLASSIFY_BLACK(below(g_m))] ≠ ∅ then
13:         δ_y ← top(g_m) − bottom(b_g)
14:         if δ_y > 0 then
15:             height(g_m) ← height(g_m) + δ_y
16:         end if
17:     end if
18:     s ← RED_GREEN_MATCH(r_m, g_m)
19:     if s ≠ ∅ then
20:         return s
21:     else if r_m ≠ ∅ ∧ g_m ≠ ∅ then
22:         e_r ← |1 − height(r_m)/width(l)|
23:         e_g ← |1 − height(g_m)/width(l)|
24:         if height(g_m) > width(l) ∧ e_g > e_r then
25:             FIX_HEIGHT(g_m, r_m)
26:         else if height(r_m) > width(l) ∧ e_r > e_g then
27:             FIX_HEIGHT(r_m, g_m)
28:         end if
29:         return RED_GREEN_MATCH(r_m, g_m)
30:     else if r_m ≠ ∅ then
31:         g, y ← CREATE_ROIS_FROM_RED(r_m)
32:         return (r_m, y, g)
33:     else if g_m ≠ ∅ then
34:         r, y ← CREATE_ROIS_FROM_GREEN(g_m)
35:         return (r, y, g_m)
36:     end if
37: end function
```

The union $\rho_1 \cup \rho_2$ is defined as the ROI ρ_m that contains both ρ_1 and ρ_2; the intersection $\rho_1 \cap \rho_2$ is the ROI that is contained in both ρ_1 and ρ_2 (which might be the empty ROI).

that does not have enough saturation to discriminate it from other, unwanted objects. In this case, the merged ROI may still not cover the full area of the signal light, but we also do not suffer from bloom. Since we do not expect black clutter (T-shirts, black machine parts etc.), we can look for the black socket (l. 12) or the black cap on top (l. 4). If the "black" classifier is successful, r_m or g_m may be improved using the respective black ROIs (ll. 5–10 and 13–16). In

the case of green, we only extend g_m (i.e. δ_y must be positive), since an unlit green signal part often turns out so dark as to appear black.

After this pre-processing the red/green matching algorithm is tried once with r_m and g_m (l. 18). If this succeeds, we have successfully obtained a tuple (r_m, y, g_m) that covers the full signal tower and can be passed on for tracking, brightness classification and blinking detection. If the red/green matching fails while both r_m and g_m are defined, one of the two ROIs might be blown up because of bloom, and can be improved if the other one does not suffer from bloom. Since the width of both r_m and g_m is limited to the width of the laser ROI l, we can estimate how badly bloom affects a ROI by its aspect ratio (ll. 22–23). The height of a bloom-affected ROI can then be improved in relation the ROI that is less affected (ll. 24–28).

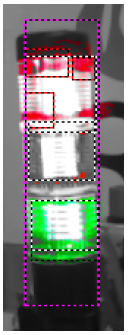


Fig. 6. Laser ROI (pink rectangle), overexposed ROIs

After this, the Red/Green matching is tried once more with improved r_m or g_m. If this fails again, we give up on the current combination of ROI sets.

Apart from the case where we were able to obtain both r_m and g_m, we also handle cases where one of the two is missing (ll. 33–35). If, e.g., there is only a red ROI r_m, matching green and yellow ROIs can be generated. In this case a black ROI b that might have been found can be used to estimate the overall height of the color ROIs. Eventually, three similarly sized ROIs should be found.

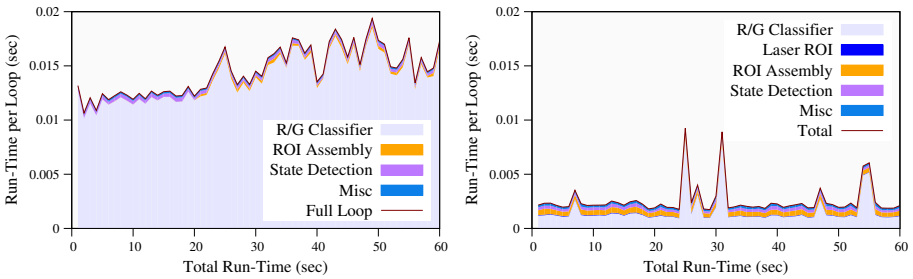
4.4 Tracking, State Detection, and Filtering

After ROIs have been determined, distance-based tracking is performed. A resulting ROI tuple denoting a signal tower is matched against previous detections based on their distance and a maximum threshold (Alg. 1, ll. 14–16).

To determine the activation states, the brightness of the respective ROIs is evaluated. ROIs of high brightness are considered to be active lights. This information is stored in a circular buffer. The buffer length is determined by the number of frames that can be processed per second and the maximum blinking frequency in the RCLL, which is 2 Hz. The light state is considered to be unknown, as long as the buffer is not completely filled. Once filled, the number of on/off transitions is counted. If this is larger than 1, the specific light is blinking.

Additionally, a confidence value is produced based on the visibility of the signal tower. A positive value for this visibility history denotes consecutive positive sightings, negative values how many images the signal tower could not be detected. The value immediately turns negative on failed detections and is not step-wise decremented.

A filtering stage can be used that performs outlier removal, i.e., if the light signal is not visible for a short time the old state is assumed to still be valid. Additionally, the visibility history is used to explicitly state that a signal is unknown if the value is below a given threshold.



(a) Full image analysis (w/o laser ROI) (b) With laser-based ROI

Fig. 7. Run-time data during live detection with and without a laser ROI. The Y axis denotes the time since system start, the X axis shows the run-time of the algorithm in 1-second averages stacked by sub-components.

5 Evaluation

The approach has been evaluated in terms of run-time and detection rates. The experiments were conducted on the actual robot that features an additional laptop (cf. Fig. 2) with a Core i7-3520M CPU and 8 GB of RAM.

Fig. 7 shows the *run-time* per frame as 1-second averages (30 images), without (a) and with (b) laser-based ROI pre-processing. During each run, the situation was modified twice after 20 and after 40 seconds, each time introducing more background clutter. Overall, the classifier requires the largest amount of processing time. After introducing more clutter, this part requires more processing time (to be expected with more pixels classified as red or green), as does the ROI assembly stage, since more ROIs are produced and are tried to be combined to a signal tower. Enabling the laser-assisted ROI pre-processing considerably reduces the overall processing time due to search space reduction for the classifier. The ROI assembly stage takes longer since it now requires additional classifier runs for the black cap and socket. The occasional outliers in (b) are due to the laser-line detection not converging and falling back to full-image classification.

Table 1 shows the *detection rate* from running the image processing pipeline on an actual robot detecting signals on an MPS in three situations posing typical

Op. Mode	**w/o Laser ROI**			**w/ Laser ROI**			**Filtered (w/ LR)**		
Dataset ▷	DS 1	DS 2	DS 3	DS 1	DS 2	DS 3	DS 1	DS 2	DS 3
# T/P	310	350	604	708	764	692	495	463	393
# F/P	49	271	152	137	102	175	25	21	7
# F/N	509	245	112	23	0	1	0	0	0
Rate (%)	**35.71**	**40.42**	**69.59**	**81.57**	**88.22**	**79.72**	**95.19**	**95.66**	**98.25**

Table 1. Results after applying the approach in three situations (cf. Fig. 8), each with seven signal combinations and from four different positions in front of the MPS; we give True (T) and false (F) positives (P) and negatives (N) (T/N omitted in this test), and detection rate.

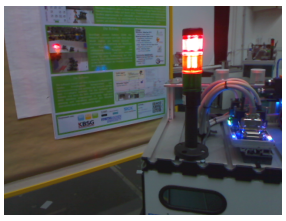
problems. For each situation, the robot moved to four nearby locations facing the MPS and took 30 images. This was done for all valid light signal combinations (no blinking). Fig. 8 shows example images for each dataset. Three different configurations were used. The pipeline was run without and with the laser-based ROI pre-processing. Finally, filtering was enabled. Blind search incurs high runtime and mediocre detection results (first macro column). Using the laser-based ROI vastly reduces the search space, increasing the detection rate considerably (second macro column). This is improved even further using the filtering and outlier removal (last macro column). With conservative settings requiring a high confidence, this results in virtually no false detections in actual games.

6 Conclusion

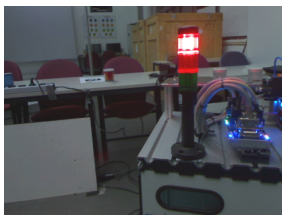
Integrating robots into human working areas will require recognizing cues that were designed for human consumption, such as light signal towers which are mounted to many machines in factories. In this paper, we have presented a novel approach to detect such towers and recognize the respective signal states. The algorithm encodes detailed human knowledge (collected in several RCLL competitions) that deals with typical problems that arise, for instance due to reflections of the lights on metal machine parts, or because colored light shines into adjacent lights when illuminated. To improve efficiency and robustness, a multi-modal approach has been chosen combining detection from a 2D laser scanner and a camera image. To use the algorithm in a new situation, the main modification required is providing a new mapping from such 2D laser scanner data to a region of interest in the image. The evaluation results show that the algorithm performs at a high speed allowing real-time light tower detection with a very good detection rate yielding only a negligible number of false readings.

An implementation of the algorithm is available as part of the Fawkes software stack release[4] for the RCLL [12]. The datasets and evaluation scripts are available on the project website.[5]

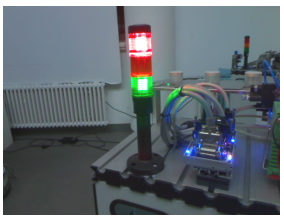
[4] `https://www.fawkesrobotics.org/p/rcll2015-release/`
[5] `https://www.fawkesrobotics.org/p/rcll-signal-vision`



(a) DS 1: green background clutter and reflection.



(b) DS 2: some clutter, red shines into yellow light.



(c) DS 3: good conditions, no clutter.

Fig. 8. Example images from the datasets used in the detection rate evaluation.

Acknowledgments T. Niemueller was supported by the German National Science Foundation (DFG) research unit *FOR 1513* on Hybrid Reasoning for Intelligent Systems (`https://www.hybrid-reasoning.org`).

References

1. Kagermann, H., Wahlster, W., Helbig, J.: Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Final Report, Platform Industrie 4.0 (2013)
2. Lucke, D., Constantinescu, C., Westkämper, E.: Smart Factory – A Step towards the Next Generation of Manufacturing. In: Manufacturing Systems and Technologies for the New Frontier, 41st CIRP Conf. on Manufacturing Systems. (2008)
3. Niemueller, T., Ewert, D., Reuter, S., Ferrein, A., Jeschke, S., Lakemeyer, G.: RoboCup Logistics League Sponsored by Festo: A Competitive Factory Automation Testbed. In: RoboCup Symposium 2013. (2013)
4. Andersen, R.H., Solund, T., Hallam, J.: Definition and Initial Case-Based Evaluation of Hardware-Independent Robot Skills for Industrial Robotic Co-Workers. In: 41st International Symposium on Robotics. (2014)
5. Angerer, S., Strassmair, C., Staehr, M., Roettenbacher, M., Robertson, N.M.: Give me a hand – The potential of mobile assistive robots in automotive logistics and assembly applications. In: IEEE International Conference on Technologies for Practical Robot Applications (TePRA). (2012)
6. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: Robocup: The Robot World Cup Initiative. In: 1st Int. Conference on Autonomous Agents. (1997)
7. Ziegler, J., Bender, P., Schreiber, M., Lategahn, H., Strauss, T., Stiller, C., Dang, T., Franke, U., Appenrodt, N., Keller, C.G., Kaus, E., Herrtwich, R.G., Rabe, C., Pfeiffer, D., Lindner, F., Stein, F., Erbs, F., Enzweiler, M., Knöppel, C., Hipp, J., Haueis, M., Trepte, M., Brenk, C., Tamke, A., Ghanaat, M., Braun, M., Joos, A., Fritz, H., Mock, H., Hein, M., Zeeb, E.: Making Bertha Drive – An Autonomous Journey on a Historic Route. IEEE Intelligent Transportation Systems Magazine **6**(2) (2014)
8. Fairfield, N., Urmson, C.: Traffic light mapping and detection. In: Int. Conf. on Robotics and Automation (ICRA). (2011)
9. Jentzsch, S., Riedel, S., Denz, S., Brunner, S.: TUMsBendingUnits from TU Munich: RoboCup 2012 Logistics League Champion. In: RoboCup Symposium. (2012)
10. Barrett, S., Genter, K., He, Y., Hester, T., Khandelwal, P., Menashe, J., Stone, P.: UT Austin Villa 2012: Standard platform league world champions. In: RoboCup Symposium. (2012)
11. Murch, C.L., Chalup, S.K.: Combining Edge Detection and Colour Segmentation in the Four-Legged League. In: Australasian Conf. on Robotics and Autom. (2004)
12. Niemueller, T., Reuter, S., Ferrein, A.: Fawkes for the RoboCup Logistics League. In: RoboCup Symposium 2015 – Development Track. (2015)