Kemény, Zs.; Csáji, B. Cs.; Viharos, Zs.: Optimization of Timing Parameters for Vision-Based Monitoring of Automated Production Lines, International Journal of Factory Automation, Robotics and Soft Computing, ISSN 1828-6984, Vol. 4, Nr. 1, 2009, pp. 30-38.

Optimization of Timing Parameters for Vision-Based Monitoring of Automated Production Lines

ZSOLT KEMÉNY, BALÁZS CSANÁD CSÁJI, ZSOLT JÁNOS VIHAROS
Computer and Automation Research Institute
Hungarian Academy of Sciences
H-1111 Budapest, Kende u. 13–17.
HUNGARY
{kemeny,csaji,zsolt.viharos}@sztaki.hu

Abstract: **The key advantage of camera-based monitoring in automated production lines is its extended functionality combined with reduced costs, owing to the possibility of replacing several classical sensors with one vision-based system. While appropriate image processing parameters are, usually, easy to find, the same cannot be said about timing parameters of the logical network which detects events in the raw binary output of visual preprocessing. The paper proposes a possible transformation of the manual tuning problem to an optimization task, allowing to find suitable timing parameters in an automatic or semi-automatic way. Finally, solutions and test results using Greedy Tabu Search are presented.**

Key-Words: **multisensor systems, image sensors, logic design, production systems, process monitoring, optimization methods**

1. Introduction

Automated production lines have to run within defined tolerance limits. To this end, appropriate control must be thus applied to the processes, mostly in *closed-loop control to handle uncertainties and unpredictable disturbances*. The use of feedback necessitates sensors whose introduction, however, is associated with higher costs and higher setup time. This is why the justification of applying a given sensor depends on the ratio of the estimated gain or quality of control vs. the costs and time demand of installation and use. Applying a cheaper sensor while providing the same reliability can either *i)* decrease costs of control solutions which require feedback anyway, or *ii)* allow the placement of sensors (and, thus, possibly improve control or surveillance) where this did not pay before. The use of cameras to replace several sensors, for example, would be in accordance with the aforementioned goal, and its feasibility was successfully demonstrated by the recently completed 6th Framework EU project *MultiSens* (Cameras as Multifunctional Sensors for Automated Processes) [1].

Production systems and processes are usually supervised using process or product level signals, followed by signal processing and decision-making modules [10, 8]. In contrast, the MultiSens vision system *directly replaces sensors feeding PLCs (programmable logic controllers) with binary signals*, aside from providing an additional visual aid for setup and surveillance of the given process. The range of possible approaches was quickly narrowed down by the well-defined structure of industrial environments and the finding that most of the targeted applications reveal process complexity over an entire *sequence* of images and not in a single snapshot. This calls for simple but fast and robust, rather local, image processing (referred to as *VBLS, vision-based logical sensors*), as opposed to sophisticated feature identification [11] or tracking [7], often in calibrated images of a single camera or an entire network [3], generation and verification of image hypotheses [12], or handling of global image properties [2], the latter prevailing in quality check and not direct process control.

Since a one-to-one mapping is not always given for "conventional" sensors and their vision-based counterparts, additional *sequential logical processing* is needed to generate the binary outputs. For the cases examined, the required logical network remained fairly simple, but related event generation problems can also be addressed on a more elaborate level in the *complex event processing (CEP)* paradigm which is, as for manufacturing, focusing on higher levels of production [6]. CEP is considered one of several sub-domains of *event stream processing (ESP)* [5], all of these dealing with extracting and processing events under real-time conditions.

In our case, the processing tasks remain simple and the number of sensor inputs is usually small. Therefore, the logical structure for the processing network can be usually found either immediately in an intuitive way, or

relying on a set of basic design rules. While this means that the first design phase is not too difficult, *initial results highlighted the determination of correct timing parameters as a more demanding and crucial issue in setting up the sequential evaluation apparatus. Since this can be a potential problem for less experienced maintenance staff as well, the timing problem calls for a tool automatically finding the correct parameters or providing guidance for a manual setup.*

2. PROBLEM STATEMENT

2.1 Vision-based feedback in MultiSens

As mentioned before, the goal of *MultiSens* was the replacement of one or more "conventional" sensors with a single camera, providing the same binary signals for the PLC controlling the process as the original sensors. For this purpose, *vision-based logical sensors* (VBLS) are applied, each of them processing the image properties within its own *region of interest* (ROI) and performing an adequate mathematical operation on this ROI to compute a binary output. The VBLSs of a given evaluation network are referred to as the *VBLS level*.

Next—if needed—the VBLS signals are processed by *compound logical sensors* (CLS) which realize logical functions (NOT, AND, OR etc.) and provide the option of *temporal aggregation* (TA), i. e., thresholded statistical filtering of a given time window. CLSs form directed trees (with data flowing from the "leaves" towards the "root") and comprise the so-called *CLS level*. The final layer declares a so-called *scenario* (SC), where *leading edges* of the root CLSs in the CLS level generate *events* which are checked against a timing table describing when and with what accuracy the given events should occur. The scenario is successfully completed if all prescribed events are detected and are in accordance with nominal occurrence times and tolerances. All three levels working together form a complete *VBLS—CLS—SC graph* (Fig. 1). To succeed in setting up an adequate VBLS—CLS—SC network (i. e., which recognizes a process within tolerance limits as successful and discards every other case as a failure), following steps should be completed:

1. Define the area to be observed, provide a suitable camera, and set proper lighting conditions.
2. Select regions of interest (ROIs) in the picture of the camera from which the same information can be extracted as from the sensors meant to be replaced.
3. Assign each ROI to a VBLS, select the mathematical operation and adequate parameters for each.
4. Set up the logical structure of the CLS tree and the events of the scenario.

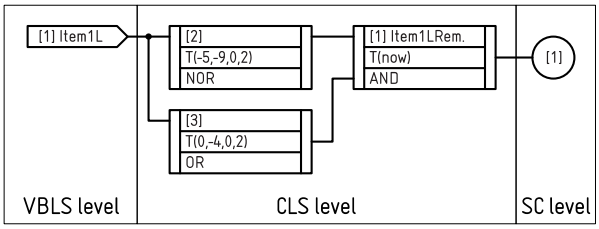

Figure 1: Simple example of a VBLS—CLS—SC network with a single event on the scenario level

5. Find the timing parameters for the TA operators and the SC events, so that process classification of success vs. failure is as required.

The first three tasks can be easily solved relying on knowledge of the automated process, some image processing practice and common sense. The fourth step may already require specific experience but a collection of design cases could still facilitate the setup. Lack of practical skills may also cause problems in completing the last step, as it is not always straightforward to find the right timing parameters. The latter shortcoming led to the formulation of the problem discussed in this paper, namely, of the need of:

- *a benchmarking method expressing the qualities of the given timing configuration numerically, and*
- *an optimization method which can automatically find the timing parameters best fit for the given purpose (and using the aforementioned measure as an objective function).*

2.2 Properties of the timing problem

Naming conventions, basic assumptions. Before examining the timing behavior of the VBLS—CLS—SC network, let us take a look at some fundamental definitions. As cameras deliver sequences of *frames*, these can be taken as basic time units, obtaining a *discrete-time network*. Here, some basic notations are introduced that are used through the paper (see also Fig. 2):

Number of a given frame	t
Current frame	t_{NOW}
Leading and trailing edges	
· of a signal, resp.	t_s, t_e
· of a negated signal, resp.	t'_e, t'_s
· of the ith input's signal, resp.	$t_{s\ in\ i}$, $t_{e\ in\ i}$
· of an output signal, resp.	$t_{s\ out}$, $t_{e\ out}$
Number of last frame	
(constant for a given sequence) . . .	t_{LF}

Also, it is assumed that all evaluation and event triggering activities are finished by the last frame of an image sequence:

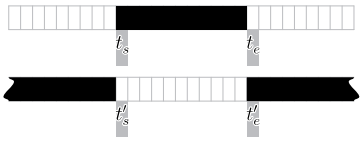

Figure 2: Leading and trailing edge definition of a signal (top), and its negation (bottom)

$$\forall t: \;\; 0 \le t \le t_{LF} \tag{1}$$

Simple logical operators. Using the simplifying assumption of a logical variable behaving bounce-free over time (i. e., only the first leading edge and the last trailing edge are taken into consideration), let us examine the timing-related properties of the logical operators used in a VBLS—CLS—SC network. For logical AND with n inputs, one obtains for the leading edge:

$$t_{s\ out} = \max_n t_{s\ in\ n} \quad n = 1 \ldots N, \tag{2}$$

which can be rewritten to the following set of N inequalities:

$$\bigwedge_n t_{s\ out} \ge t_{s\ in\ n} \quad n = 1 \ldots N, \tag{3}$$

while for the trailing edge, one obtains

$$t_{e\ out} = \min_n t_{e\ in\ n} \quad n = 1 \ldots N \tag{4}$$

$$\bigwedge_n t_{e\ out} \le t_{e\ in\ n} \quad n = 1 \ldots N. \tag{5}$$

A nonzero output is generated if all inputs overlap in at least one frame (see also Fig. 3):

$$t_{s\ out} < t_{e\ out} \tag{6}$$

For logical OR, XOR, and negation, one would obtain similar sets of equations and inequalities.

Temporal aggregation. Resembling a statistical low-pass filter with a time delay, temporal aggregation (TA) was introduced to allow *i)* time delay as required for the processing network, *ii)* filling of short signal bounces, and *iii)* issuing trigger signals for comparison with scenario event specifications.

As shown in Fig. 4a, the TA operator looks back in time, beginning with a lag of T_1 frames and ending with a lag of T_2 frames, and counts the number of frames within this observation window where the input is active. Encountering A_{min} active frames or more, the current output of the TA operator becomes active (referred to, more generally, as "Level 1"), while going beyond the next threshold A_{max}, "Level 2" is entered which, in the current implementation of TA, means that the output becomes false again. A reasonable choice of A_{min} and A_{max} is suggested by:

$$0 \le A_{min} < T_2 - T_1 + 1 \tag{7}$$

$$0 < A_{max} \le T_2 - T_1 + 1 \tag{8}$$

$$A_{min} \le A_{max}, \tag{9}$$

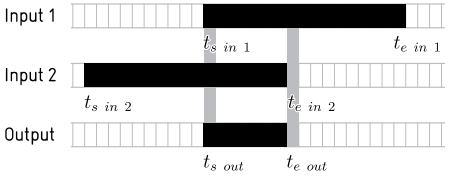

Figure 3: Timing diagram for logical AND with two overlapping inputs

including the cases of the TA always being at least in "Level 1", and the TA never reaching "Level 2".

Passing a signal through a TA element, the earliest occurrence of "Level 1" and "Level 2" states (see also Fig. 4b,c) are encountered at

$$t_{s\ out\ 1} \ge t_{s\ in} + T_1 + A_{min} - 1, \tag{10}$$

$$t_{s\ out\ 2} > t_{s\ in} + T_1 + A_{max} - 1. \tag{11}$$

respectively, while leaving "Level 2" (i. e., re-entering "Level 1") and leaving "Level 1" occur, respectively, in accordance with

$$t_{e\ out\ 2} < t_{e\ in} + T_2 - A_{max}, \tag{12}$$

$$t_{e\ out\ 1} \le t_{e\ in} + T_2 - A_{min}. \tag{13}$$

Cascading several TA operators introduces a series of subsequent delays into the evaluation process whose sum is subject to an upper limit t_{lim} either given by T_{LF}, or some expected event of the scenario by which all corresponding evaluation has to succeed:

$$\sum_{i \in I} T_{2\ i} < t_{lim} \tag{14}$$

All inequalities of the form (14) must simultaneously hold for all evaluation chains containing TA operators, in addition to the following fundamental inequalities:

$$0 \le T_{1\ i} \le t_{LF} \;\; \forall i \tag{15}$$

$$0 \le T_{2\ i} \le t_{LF} \;\; \forall i \tag{16}$$

$$T_{1\ i} \le T_{2\ i} \;\; \forall i \tag{17}$$

$$0 \le A_{min\ i} \le A_{max\ i} < T_{2\ i} - T_{1\ i} \;\; \forall i \tag{18}$$

(TA operators assume that all VBLS outputs before frame 0 are uniformly zero.)

Scenario evaluation. As already mentioned before, the root of each CLS tree triggers an *event*, which is an element of a *scenario*. Triggering occurs when a leading edge (either the first one, or all) occurs on the output of the corresponding root CLS. A given SC event is considered successful if

$$|t_{SI\ i} - T_{SI\ i}| \le \Delta T_{SI\ i}, \tag{19}$$

where $t_{SI\ i}$ is the frame where event i is triggered, $T_{SI\ i}$ is where it should nominally occur, and $\Delta T_{SI\ i}$ is the radius of a tolerance interval around $T_{SI\ i}$. If the triggering of an event within this interval is a sufficient condition, the SC event is called *relaxed*, while in the necessary case, it is a *strict* scenario event. For all prescribed

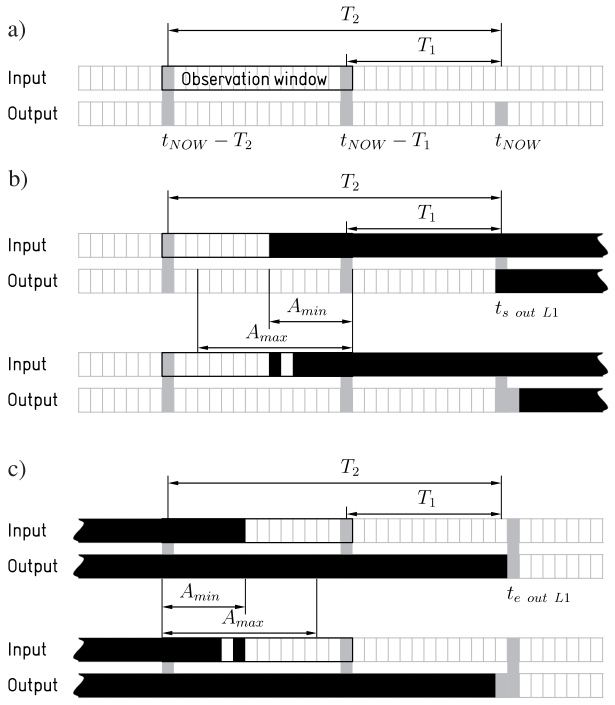

Figure 4: Properties of a TA operator: a) time window parameters, b) leading edge behavior, and c) trailing edge behavior.

SC events, the following fundamental inequalities must hold:

$$0 \leq T_{SI\,i} \leq t_{LF} \ \ \forall i \tag{20}$$

$$0 \leq T_{SI\,i} + \Delta T_{SI\,i} \leq t_{LF} \ \ \forall i \tag{21}$$

$$0 \leq T_{SI\,i} - \Delta T_{SI\,i} \leq t_{LF} \ \ \forall i \tag{22}$$

2.3 Problem formulation

The original tasks—*i)* providing a benchmarking tool for assistance of manual setup, and *ii)* elaborating a method for finding suitable timing parameters automatically—can now be refined. *It is assumed that the aforementioned range of problems can be solved by finding adequate objective functions and an optimization method*, resulting in the following steps to proceed:

1. Prepare the environment including camera settings, ROIs etc., design and implement evaluation network, set VBLS parameters (optionally SC events as well).
2. Take a number of video sequences of various runs of the observed process, and label them whether they are successful sequences or failures.
3. Find an objective function expressing how well the VBLS—CLS—SC network can tell success from failure with the given timing parameters. *This will directly deliver a benchmarking tool for manual setup.*
4. Find an optimization method which can deliver a set of timing parameter settings (T_1, T_2, A_{min}, A_{max} for all TA operators, and, optionally, T_{SI}, ΔT_{SI} for scenario events) that allow the given VBLS—CLS—SC network to identify successful process runs and failures. *This will deliver a tool for finding the correct timing parameters automatically.*

3. SOLVING PARAMETER OPTIMIZATION

As mentioned before, the fundamental working assumption of the paper is that it is possible to set the timing parameters of the VBLS—CLS—SC network by optimization. A closer examination revealed that it is most convenient if the video sequences were pre-recorded and pre-processed by the already installed VBLS layer of the VBLS—CLS—SC network, so that the optimization runs would rely on mere binary signal sequences.

It is largely left to engineering decisions taken in the given application environment whether timing parameters are set sequentially, event by event (as usual in automation practice), or for the whole network at once. Nevertheless, the solution proposed here was elaborated for the latter case, still allowing a—computationally less demanding—sequential treatment as well.

3.1 Optimization cases

VBLS-processed binary signal series obtained from pre-recorded image sequences serve as "learning samples" which can realize three different cases:

1. *One recording of a successful run*—allowing a search for a set of timing parameters which best fits the recognition of the given sequence as successful;
2. *Several recordings of successful runs*—allowing additional statistical examination of the runs (estimation of tolerances), especially for the timing of scenario events;
3. *Several recordings of successful runs as well as failures*—foregoing false classification.

The specification of the optimization tasks is also subdivided into several task groups, depending on which parameters are tuned automatically or preset manually by the installation staff:

1. *Scenario fully specified (FS)*—with all SC parameters preset, leaving only $T_{1\,i}$, $T_{2\,i}$ $\forall i$, and $A_{min\,i}$, $A_{max\,i}$ $\forall i$ of the TA operators (with index i) to be sought by optimization;
2. *Scenario partially specified (PS)*—with only the nominal occurrence frames of SC events being preset, adding $\Delta T_{SI\,j}$ $\forall j$ (j being the SC event index) to the list of unknowns defined by FS;
3. *Scenario least specified (LS)*—with the full set of $T_{1\,i}$, $T_{2\,i}$ $\forall i$, $A_{max\,i}$ $\forall i$, $T_{SI\,j}$ $\forall j$, and $\Delta T_{SI\,j}$ $\forall j$ parameters being subject to optimization.

While the latter case implies a large number of possible $T_{SI\ j}$ with corresponding $T_{1\ i}$ and $T_{2\ i}$ in the appropriate CLS trees bringing essentially the same results, one can still rely on engineering "common sense" suggesting that, mostly, at least *the order of events is fixed*, representing "stages" of a production process (resulting in adequate inequalities to be enforced as constraints). Assuming this, two redundancy resolution heuristics may be taken: *i)* generate the event as soon as possible to facilitate timely intervention if needed, or *ii)* generate the event as late as feasible, allowing to gather as much information as possible.

3.2 Building an objective function

As explained before, the VBLS—CLS—SC network performs, in fact, a classification task as it decides whether a given image sequence represents success or failure of the process observed. An objective function expressing the properties of timing settings—thus fulfilling the requirement of a benchmarking/decision support tool as well—should therefore express the classification-related properties *Decision ability (recognition rate)*, i. e., the number of correct decisions vs. all evaluation runs taken, *Decision quality*, a function with its maximum being at the nominal occurrence frame and reaching zero at the decision boundary, and *Decision safety*, expressing how well separated the successful and unsuccessful runs are in the event space. it is recommended to rank these properties according to their estimated importance and compose a *hierarchical* expression where the gradient of a measure does not largely interfere with another one of higher rank.

For examining the objective functions in detail, let us first consider the following definitions:

Number of	
· all successful sequences processed	N_s
· all unsuccessful sequences processed	N_u
· successful runs recognized as such	n_s
· unsuccessful runs recognized as such	n_u
· events within the scenario	E
· events correctly recognized as successful for sequence i	$e_{s\ i}$
· events correctly recognized as unsuccessful for sequence i	$e_{u\ i}$
"Success measure" function for event j in run i which is in the set of the N_s successful sequences	$o(i,j)$
"Success/failure measure" function for event j in run i which is in the set of the N_u unsuccessful sequences, and is computed depending on whether the event itself is labeled successful or unsuccessful.....	$\bar{o}(i,j)$

Keeping these definitions in mind, the components of the objective function are as follows.

Decision ability (recognition rate). On the sequence level, i. e., how many successful runs were recognized as such and how many failures were deemed unsuccessful according to the current timing parameters, we have:

$$Q_1 = \frac{n_s + n_u}{N_s + N_u}, \tag{23}$$

which is a discrete function with a single step being

$$\Delta_1 = \frac{1}{N_s + N_u}\ . \tag{24}$$

For the introduction of a secondary criterion Q_2 which becomes important if Q_1 does not change anymore during search, Q_2 may be scaled so that $\max|Q_2| \leq \Delta_1$. This allows us to add a *secondary measure* expressing decision ability, i. e., how many scenario items were recognized correctly. Successful sequences require a successful predicate for all scenario items as well, making the event recognition rate for successful sequences is equal to

$$Q_{2s} = \frac{\sum_{i=1}^{N_s} e_{s\ i}}{EN_s} \cdot \Delta_1\ . \tag{25}$$

Its counterpart for unsuccessful sequences is more complicated, as here, not all scenario items are required to fail, requiring further distinction:

$$Q_{2u} = \frac{\sum_{i=1}^{N_u} e_{u\ i} + \sum_{i=1}^{N_u} e_{s\ i}}{EN_u} \cdot \Delta_1\ . \tag{26}$$

If correct success/failure labeling is not provided on the event level, one can set $Q_2 = Q_{2s}$. The secondary criterion is discrete as well; with both Q_{2s} and Q_{2u} being used, the related single step size is

$$\Delta_{2s+u} = \frac{1}{E\max\{N_s, N_u\}} \cdot \Delta_1\ , \tag{27}$$

while for using Q_{2s} alone, it is

$$\Delta_{2s} = \frac{1}{EN_s} \cdot \Delta_1\ . \tag{28}$$

Decision quality. Further detailing the objective function is achieved if a *decision quality measure* is introduced, expressing how well a given evaluation instance lies between the decision boundaries. A function $o(i,j)$ for a given $t_{i\ j}$, i. e., an occurrence of event i in sequence j should have following properties:

$$o(i,j) = 1 \quad \text{for} \quad t_{i\ j} = T_{SI\ i} \tag{29}$$

$$o(i,j) > 0 \quad \text{for} \quad |t_{i\ j} - T_{SI\ i}| < \Delta T_{SI\ i} \tag{30}$$

$$o(i,j) = 0 \quad \text{for} \quad |t_{i\ j} - T_{SI\ i}| = \Delta T_{SI\ i} \tag{31}$$

In case of a strict scenario, we have furthermore $o(i,j) < 0$ for $|t_{i\,j} - T_{SI\,i}| > \Delta T_{SI\,i}$, while for a relaxed scenario, we may set $o(i,j) = 0$ for $|t_{i\,j} - T_{SI\,i}| > \Delta T_{SI\,i}$. If we would like to set a limit for $|o(i,j)|$, we may also prescribe $|o(i,j)| \leq 1$. The most simple piecewise linear function complying with the above requirements would thus be

$$o(i,j) = max\left\{1 - \frac{|t_{SI\,i\,j} - T_{SI\,i}|}{\Delta T_{SI\,i}};\ -1\right\} \quad (32)$$

for a strict scenario item and

$$o(i,j) = max\left\{1 - \frac{|t_{SI\,i\,j} - T_{SI\,i}|}{\Delta T_{SI\,i}};\ 0\right\} \quad (33)$$

for a relaxed one. As this linear function is evaluated only for integer values of $t_{SI\,i}$, we may, for a given set of $\{\Delta T_{SI\,i}\}$, determine a smallest step (excluding intervals where $o(i,j)$ is saturated):

$$\Delta_3 = \frac{1}{\max_{i=1}^{E} \Delta T_{SI\,i}} \cdot \Delta_2\ . \quad (34)$$

Unfortunately, this means that Δ_3 could change if any $\Delta T_{SI\,i}$ is subject to search, however, practical limits of ΔT_{SI} values could still give a rough estimate of Δ_3. Though a smallest possible step may not exist for other cases, the hint for Δ_3 in the linear example may still be useful if, for any reason, a quaternary criterion would be introduced. A possible candidate for a nonlinear $o(i,j)$ could be:

$$o(i,j) = 2\exp\left(-\left(\frac{t_{SI\,i\,j} - T_{SI\,i}}{\Delta T_{SI\,i}}\right)^2 \cdot \ln 2\right) - 1 \quad (35)$$

for the strict case, which can be modified as

$$o(i,j) = \max\left\{2\exp\left(-\left(\frac{t_{SI\,i\,j} - T_{SI\,i}}{\Delta T_{SI\,i}}\right)^2 \cdot \ln 2\right) - 1;\ 0\right\} \quad (36)$$

for relaxed SC items. Thus, one could assemble the tertiary criterion as follows:

$$Q_{3s+u} = \left(\frac{\sum_{i=1}^{E}\sum_{j=1}^{N_s} o(i,j)}{EN_s} + \frac{\sum_{i=1}^{E}\sum_{j=1}^{N_u} \bar{o}(i,j)}{EN_u}\right)\Delta_{2s+u} \quad (37)$$

where the function $\bar{o}(i,j)$ is

$$\bar{o}(i,j) = \begin{cases} o(i,j) & \text{if SI}_i \text{ labeled successful in seq. } j \\ -o(i,j) & \text{if SI}_i \text{ labeled unsuccessful in seq. } j \end{cases} \quad (38)$$

If single events cannot be labeled for success or failure, one can rely on the substitute

$$Q_{3s+\hat{u}} = \left(\frac{\sum_{i=1}^{N_s}\sum_{j=1}^{E} o(i,j)}{EN_s} + \frac{\sum_{j=N_u}^{E} \max_{j=1}^{E} \bar{o}(i,j)}{EN_u}\right)\Delta_{2s}\ , \quad (39)$$

using $\bar{o}(i,j) = -o(i,j)$ and the heuristics that in an unsuccessful sequence, the event with the worst evaluation results is most likely to be the most decisive source of scenario failure.

Decision safety. This would express how far away the most unfitting element of the set is from the decision boundary. For this purpose, a linear difference is used again:

$$Q_4 = \left(\sum_{i=1}^{E} \min_j q(i,j)\right) \cdot \lambda Q_3 \quad (40)$$

with

$$q(i,j) = 1 - \frac{|t_{SI\,i\,j} - T_{SI\,i}|}{\Delta T_{SI\,i}}\ , \quad (41)$$

and λ being a scaling component ensuring $|Q_4| < Q_3$. Incorrect decisions result in $\min q(i,j) < 0$, $\min q(i,j) > 0$ implies that all decisions were right.

Now, assembling the criterion function is fairly simple:

$$Q = -\left(Q_1 + Q_{2s+u} + Q_{3s+u}\right)\ , \quad (42)$$

while without event-level labeling, we may use

$$Q = -\left(Q_1 + Q_{2s} + Q_{3s+\hat{u}}\right)\ . \quad (43)$$

3.3 Finding a search algorithm

Optimization space. The discrete optimization space can be narrowed down to tractable size by the inequalities of Section 2, with (1), (15)–(18), and (20)–(22) being most fundamental, and knowledge of the given logical structure refining them further at the cost of a higher computational demand. While choosing the level of restriction detail, it must be noted that cascading several time lags may alter search space boundaries in each iteration (as would T_{SI} of SC events subject to search effect this, too). Also, the valid space may not always be convex or it may consist of disjoint regions.

Behavior of the objective function. Owing to the "decision ability" measures, the objective function contains discrete steps: for linear "decision quality" and "decision safety" measures, it is piecewise linear, while in the general case, it is at best piecewise continuous. Multiple scenario events may result in local extrema, calling—along with the piecewise continuous properties—for a *robust iterative search algorithm*.

Optimization algorithms. The routines suitable for our purposes assume an objective function $f : \mathbb{Z}^n \to \mathbb{R}$

to be *minimized* for a solution in $C \subseteq \mathbb{Z}^n$. Even if linear constraints of the form $Ax \leq b$ (where $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$) are given, f is not granted to be linear, preferring thus more elaborate methods than *linear programming*. Also, sophisticated techniques requiring a differentiable f, such as the Levenberg-Marquardt Method, are excluded due to the discrete steps in our objective function.

While initial tests also included the Nelder-Mead Simplex Algorithm [9], the volume of the paper only provides for presenting a more efficient alternative, greedy tabu search, in detail. Greedy search (i. e., discrete gradient-descent) examines a set of points within its surrounding and selects the one with the smallest objective function [4]. Avoiding local extrema is possible by maintaining a *tabu list* of points to be avoided in subsequent optimization steps. For the latter purpose, let us consider the L1 norm

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \,. \tag{44}$$

Using this, we can define the distance of two points as $d(x, y) = \|x - y\|_1$, and a sphere with range ϵ around point x as $K_\epsilon(x) = \{\, y \in \mathbb{Z}^n : d(x, y) \leq \epsilon \,\}$ where $\epsilon > 0$ ($\epsilon = 1$ in our specific case). The algorithm updates the tabu list T_i in each iteration i (starting with $T_0 = \emptyset$), restricting the sphere in iteration i to $R_i(x) = (K_1(x) \cap C) \setminus (T_i \cup \{x\})$. Within this set, the tabu search proceeds by:

$$x_{i+1} \in \operatorname*{arg\,min}_{x \in R_i(x_i)} (f(x) + \delta_i), \tag{45}$$

with the starting point x_0 being given in advance or selected randomly from C. Avoiding local minima can optionally be helped with the random variable δ_i with zero mean and $\delta_i \to 0$ as $i \to \infty$ with probability one. If $\delta_i \neq 0$, a variant of the Metropolis (simulated annealing) algorithm can be recognized. A random restart is initiated if $R_i(x_i) = \emptyset$, selecting a new x from $C \setminus T_i$. The tabu list is updated in each iteration by $T_{i+1} = T_i \cup \{x_i\}$, and further maintenance may keep the number of tabu elements small, e. g., in a FIFO manner. Convergence properties can be further improved by step-size adaptation; in the experiments carried out, it doubled the average speed of tabu search.

4. Experimental results

For the verification of the theoretical findings, video sequences of several runs of the same process were recorded and pre-processed to binary sequences using the *MultiSens* prototype application for building VBLS—CLS—SC networks [1]. With these sequences, two test groups were carried out:

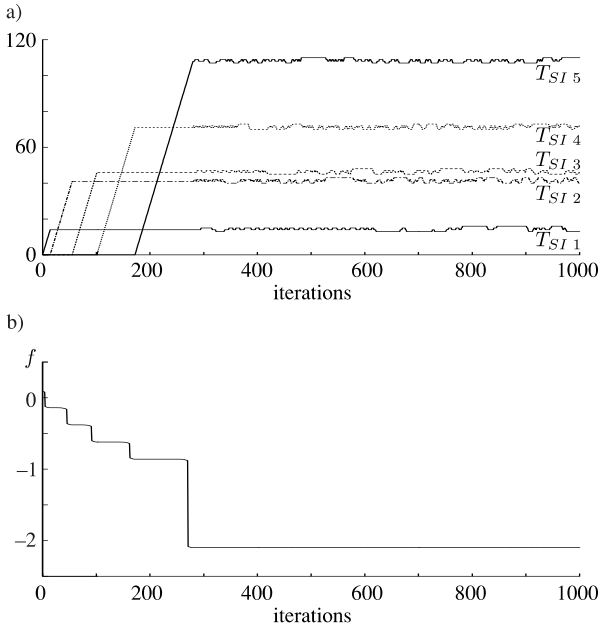

Figure 5: Search of $T_{SI\,i}$ parameters with tabu search: a) $T_{SI\,i}$ values, b) objective function $f = -(Q_1 + Q_{2S} + Q_{3s+\hat{u}})$ during optimization.

1. In the first test series the T_{SI} parameters of all scenario items were subject to search, keeping the TA values fixed. As it can be seen in Fig. 5a, all parameters locked in on the area around the appropriate values which can also be noticed in Fig. 5b that shows the behavior of the target function.
2. During the second test series all parameters of both temporal aggregation operators in the VBLS—CLS—SC network were subject to the search, but the scenario event parameters were fixed. In Fig. 6b, it can be observed that the objective function quickly drops to a near-minimal value and remains there for most of the remaining iteration steps. This hints at large "plateaus" in the terrain of the target function, implying that the recognition quality of the network is robust against timing parameter changes. These plateaus are, however, expected to shrink if more and more video sequences are introduced for training.

5. Conclusion

While replacement of several "conventional" sensors with a vision system has a great potential in the control of automated processes, lack of initial experience in setting the correct timing parameters for logical evaluation networks may impair its practical applicability. This problem can be overcome by a benchmarking and parameter search tool supporting the installation staff. The paper proposed *a multilevel set of possible measures for the evaluation of recognition properties* in such logical

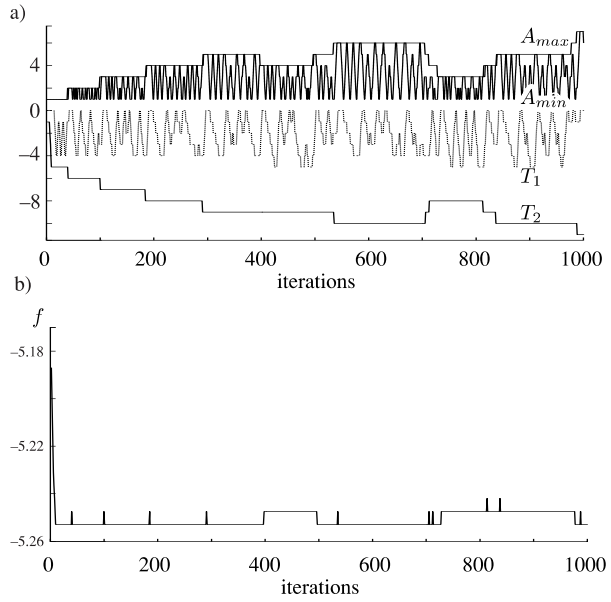

Figure 6: Search of TA parameters with tabu search: a) T_1, T_2, A_{min}, A_{max} values of a selected TA operator, b) objective function $f = -(Q_1 + Q_{2S} + Q_{3s+\hat{u}})$ during optimization.

networks, and argued that *a search for correct timing parameters can be formulated as a discrete multidimensional optimization problem*. During experiments carried out on pre-processed video sequences of a sample physical process, *greedy tabu search was found suitable for timing parameter optimization. Step-size adaptation was found to improve convergence properties.* Altogether, the paper demonstrated that *timing parameter search for networks of vision-based sensors can be formally modeled and effectively dealt with using multidimensional optimization algorithms. Although not addressed directly in the paper, the results can be applied to timing problems related to "conventional" sensors as well.*

Further research may improve the presented findings in several directions. In our experiments, the output of the vision-based sensors was pre-processed to binary sequences, however, a continuous output signal could convey additional information to enhance process monitoring. In that case, *fuzzy logic operators* could be applied resulting in fuzzy CLSs. Also, practical application may increase the number of learning sequences to a computationally untractable degree. For such large datasets, further pre-processing (e. g., *data mining* techniques, as *support vector classification* or *adaptive sampling*) could provide a feasible solution, so that the optimization procedure is carried out on a smaller set of learning samples already compacted by one of the aforementioned methods.

References

[1] A. Argyros, G. Bártfai, C. Eitzinger, Zs. Kemény, B. Cs. Csáji, L. Kék, M. Lourakis, W. Reisner, W. Sandrisser, T. Sarmis, G. Umgeher, and Zs. J. Viharos. *Smart Sensor Based Vision System for Automated Processes*, pages 24–29. International Society for Advanced Research, June 2007.

[2] N. Bauer, editor. *Guideline Industrial Image Processing*. Fraunhofer-Allianz Vision, 2003.

[3] R. B. Fisher. Self-organization of randomly placed sensors. In *Proc. of the Eur. Conf. on Computer Vision, Copenhagen*, pages 146–160, May 2002.

[4] F. Glover. Tabu search—part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[5] M. Li, M. Liu, L. Ding, E. A. Rundensteiner, and M. Mani. Event stream processing with out-of-order data arrival. In *Proc. of the 27th International Conference on Distributed Computing Systems Workshops, 2007, ICDCSW'07.*, volume 983, pages 67–67, September 2007.

[6] D. Luckham, editor. *The Power of Events—An Introduction to Complex Event Processing in Distributed Enterprise Suystems*. Addison-Wesley, 2002.

[7] É. Marchand and F. Chaumette. Feature tracking for visual servoing purposes. *Robotics and Autonomous Systems*, 52(1):53–70, June 2005.

[8] G. Meltzer and N. P. Dien. Fault diagnosis in gears operating under non-stationary rotational speed using polar wavelet amplitude maps. *Mechanical Systems and Signal Processing*, 18(5):985–992, September 2004.

[9] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–311, 1965.

[10] P. Omenzetter, J. M. W. Brownjohn, and P. Moyo. Identification of unusual events in multi-channel bridge monitoring data. *Mechanical Systems and Signal Processing*, 18(2):409–430, March 2004.

[11] C. Setchell and E. L. Dagless. Vision-based road-traffic monitoring sensor. *IEEE Proc. Vision, Image, and Signal Processing*, 148:78–84, February 2001.

[12] G. von Wichert. A probabilistic approach to simultaneous segmentation, object recognition, 3d localization, and tracking using stereo. *Lecture Notes In Computer Science; Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, 2191:100–107, 2001.