Online Workflow Recovery

Michael Tautschnig
Supervisor: Prof. Dr. Rupert Lasser
Instructors: Prof. Dr. Nassir Navab, Dipl. Inf. Tobias Sielhorst

Institut für Informatik, Technische Universität München,
Boltzmannstraße 3, D-85748 Garching bei München, Germany
email: tautschn@in.tum.de

1 Introduction

Workflow recovery deals with the problem of identifying related phases of two recorded processes, given one of them has been annotated as desired. Once the relation has been established, the information can be used for, e. g., documentation purposes or process optimization. On the other hand, training feedback can be given by synchronizing a trainee's 3D hand movements to those of an expert surgeon [7]. In the latter case, dynamic time warping (DTW) has been used, which has also been applied successfully in the context of statistics, speech recognition or error detection in industrial processes [3].

Our motivation stems from the analysis of surgical operations. The offline recovery of the workflow allows for, e. g., automated documentation, but also for general OR workflow optimization. The importance of this issue in the OR of the future has been underlined in the OR2020 workshop [1]. Further work on this subject is based on hidden Markov models to analyze the skills of the surgeon [5], [4]. However, their focus is on extraction and analysis of single tasks, whereas here the analysis of the whole process, i. e., the complete surgery, shall be emphasized. To this end, the work of Ahmad Ahmadi, Tobias Sielhorst, Martin Horn and Ralf Stauder seems to be unique. In the following paragraphs their work and their results are summarized.

Six cholecystectomies, all conducted by Prof. Feussner at Klinikum Innenstadt in Munich, were recorded using a laparoscopic and three external cameras. As the removal of the gall bladder is performed mainly laparoscopically and considered a common case for the surgeon, this type of surgery is particularly suitable. Furthermore, the sequence of steps performed therein is, apart fromm complications, very stable. Thus it was possible to extract 14 effective surgical phases as listed in Table 1, which are expected to occur in this particular order in every surgery of that type.

The recovery has been performed on sets of state vectors extracted manually from the videos. Each second a bit vector representing the tools used has been stored and augmented with a time stamp. Each element of the vector was set to 1, if the tool has been used at the second, else it is set to 0. The 17 tools represented in these vectors can be found in Table 2.

Phase no.	Title
1	CO_2 inflation
2	trocar insertion
3	dissection phase 1
4	clipping/cutting 1
5	dissection phase 2
6	clipping/cutting 2
7	gall bladder detaching
8	liver bed coagulation 1
9	packaging of gall bladder
10	external gall bladder retraction
11	external cleaning
12	liver bed coagulation 2
13	trocar retraction
14	abdominal suturing

Table 1. Phases of a cholecystectomy

Tool index	Name
1	umbilical port
2	trocar 1
3	trocar 2
4	trocar 3
5	trocar 4
6	optics
7	liver rod
8	grasper
9	dissecting PE
10	coagulation rod
11	HF cutting
12	HF coagulation
13	suction and irrigation device
14	laparoscopic scissors
15	clipping device
16	retraction sac
17	drainage tube

Table 2. Tools denoted in the state vectors

Using this data, the aim was to determine the starting points of the phases in any similar surgery, given one has been annotated, i. e., the phases have been extracted and assigned manually. The offline recovery has been performed successfully using dynamic time warping, whereby 92 % of the phase transitions were detected up to a precision of 5 seconds, 83 % were even exact. The quality of the synchronization was further confirmed by playing the videos of the surgeries in parallel. Using the results of DTW, the playback speed has been adapted and the visualization proved the good quality of the synchronization attempt. However, to allow for this success, an appropriate annotated surgery had to be found. It is created from five of the six surgeries by first electing one of them as the reference curve, OP_{Ref}. Using DTW as detailed in Section 2.1, the warp paths h_i, $1 \leq i \leq 5$, are computed. After scaling h_i to some common average timing of length t_{Avg}, a new average warp path is found as

$$h(t) = \frac{1}{5} \sum_{i=1}^{5} h_i(t) \quad 0 \leq t \leq t_{Avg}.$$

Based on this path, an artificial surgery OP_{Avg} is constructed by averaging over corresponding points of the five surgeries. Let $OP_i(t)$ denote the state vector of surgery i at time t. Then OP_{Avg} is defined as

$$OP_{Avg}(t) = \frac{1}{5} \sum_{i=1}^{5} OP_i \left(h_i \left(h^{-1}(t) \right) \right),$$

where $h^{-1}(t)$ is the inverse of $h(t)$, which may need to be averaged or interpolated.

In continuation of their work, the present work has been conducted to test the feasibility of online recovery. Thereby even context aware actions could be performed, such as switching on and off the lights or even context aware information systems could be implemented. Furthermore, knowledge about the expected next steps would allow for automated preparation of the required tools. To this end, possible algorithmic approaches had to be found and evaluated.

Due to the success of workflow recovery by Ahmadi et al., the quality of any online approach is measured by the similarity to the offline results computed using dynamic time warping. Apart from that, the time stamps of phase transitions provide another way to estimate the quality of any approach. Before getting at the details, let us introduce some basic terms.

2 Basic Concepts

A *time series* is a set of data points with associated time stamps, which allow for a total ordering of the set. We denote a *data point* as a pair (t, x), where t is some numerical representation of time. The effective data x is an n-dimensional

vector, $n \geq 1$. If A is a time series, A_m is the subset of the first m points of A, and a_m denotes the m-th data point of A.

Given two time series A and B, the question of workflow recovery is to establish a relation $\sim$ between the data points of A and B. The desired relation must meet the following criteria: It must preserve time ordering, that is for all $(t_1, a_1), (t_2, a_2) \in A$, $(t_3, b_1), (t_4, b_2) \in B$ and $t_1 \leq t_2$

$$(t_1, a_1) \sim (t_3, b_1), (t_2, a_2) \sim (t_4, b_2) \Rightarrow t_3 \leq t_4.$$

This justifies the term *synchronization* of the trajectories. Furthermore, the relation must satisfy certain optimality criteria, e. g., the total distance is to be minimal:

$$\sum_{(t_i, a_i) \sim (t_j, b_j)} \|a_i - b_j\| \to \min$$

On the other hand, the relation might be required to be total, i. e., for all $a \in A$ there must be at least one $b \in B$, such that $a \sim b$, as well as for each $b \in B$ there must be at least one $a \in A$ satisfying $a \sim b$.

2.1 Dynamic Time Warping (DTW)

DTW features non-linear alignment of two time series. It squeezes or stretches corresponding substrings to obtain a relation of minimum distance. A thorough introduction to the algorithm is found in [6], where it has been developed in the context of speech recognition. The use for the alignment of curves is detailed in [9].

Given a distance function $\mathrm{d}(x, y)$ and two time series A and B of length m and n, the $\mathrm{DTW}(A, B)$ is defined as follows:

$$\mathrm{DTW}(A, B) = \mathrm{d}(a_m, b_n) + \min \left\{ \begin{array}{c} \mathrm{DTW}(A_{m-1}, B_{n-1}), \\ \mathrm{DTW}(A_{m-1}, B), \\ \mathrm{DTW}(A, B_{n-1}) \end{array} \right\}$$

By means of dynamic programming the complexity of this recursive algorithm can be reduced to $\mathcal{O}(|A| \cdot |B|)$. This is accomplished by storing the values in a matrix, which is filled incrementally. Let us illustrate this in the following example: Let $A = (1, 3, 3, 4, 2)$ and $B = (1, 2, 3, 3, 3, 2, 2)$ be two trajectories. We omit the time stamps and just fix the order as given. Using the absolute value as the distance measure, one can fill in the values row-wise from left to right or per column from the bottom to the top. The result is shown in Table 3.

Note, that the smallest possible distance for the complete sequences is immediately found in the upper right corner. To compute the effective relation, a backtracking is started from there to the lower left corner. The path obtained by selecting one of the neighbors of locally minimal value is found in time $o(|A| + |B|)$. In our case we find the relation to be

$$(1, 1), (1, 2), (3, 3), (3, 3), (4, 3), (2, 2), (2, 2).$$

2	8	4	3	3	3	**2**	**2**
4	7	4	2	2	**2**	3	4
3	4	2	1	**1**	1	2	3
3	2	1	**1**	1	1	2	3
1	**0**	**1**	3	5	7	8	9
	1	2	3	3	3	2	2

Table 3. Matrix for DTW(A, B)

In the context of DTW the resulting relation is also called the *warp path*. It is guaranteed to contain all elements of A and B, i. e., it is total, but some of them might be used several times. Thereby stretching and squeezing of similar substrings is accomplished. As seen in the introduction, it may be convenient to write the warp path as a function

$$h : \{1, 2, \ldots, m\} \to 2^{\{1,2,\ldots,n\}}.$$

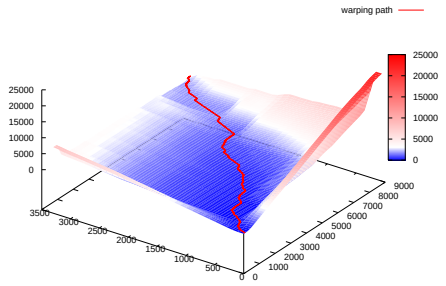

Figure 1. Visualization of a DTW matrix and the warp path

Figure 1 shows a visualization of a DTW matrix obtained for one of the surgeries. The values of the entries of the matrix are drawn on the z axis. Additionally, the warp path is shown as a red line.

3 Online Synchronization

To obtain results close to those of offline recovery, presumably we need to base our approach on DTW. However, we somehow need to guess the subset of the reference trajectory the current status of the surgery is to be matched to. In our context, the *reference surgery* is the time series the data of the ongoing surgery is registered to.

The dynamic programming approach, as discussed above, already works incrementally. Thus the online implementation can, e. g., add a row each time the routine is called. However, performing the backtracking as in case of the offline version, is not appropriate unless the the ongoing surgery has just finished. But then we could do offline recovery anyway. Thus, the task of all approaches described below is solely finding an appropriate end point, i. e., the data point of the reference model that matches the current last element of the ongoing process. Once such a point has been found, backtracking from this point can be performed as usual.

3	4	2	1	**1**	1	2	3
3	2	1	**1**	1	1	2	3
1	**0**	**1**	3	5	7	8	9
	1	2	3	3	3	2	2

Table 4. Matrix for DTW(A, B_3)

As an example, reconsider the time series used in Table 3. Under the assumption that only $(1, 3, 3)$ is known of B Table 4 is obtained. If B continues as before, the bold numbers mark the correct warping path. However, it is not obvious why the 3 in the middle should be chosen. Before discussing possible solutions, parts of this problem shall be analyzed.

3.1 Subproblems

Constant substrings. As each action usually takes more than one second, the state vectors of the reference curve stay constant for some time. Even if state vectors matching some part of the reference model are found in the running surgery, it is a priori undecidable how many elements of the reference are to be associated with the current data. A possible approach is to synchronize each second of the running surgery with one second of the reference, as long as the end of this action is not known, which would mean that the action had finished. An example of this problem is found in Figure 2, where the red marks denote the online guesses, whereas the a posteriori correct warp path is drawn as a green line. Such situations are also called flat areas as the value of DTW is all the same in these parts of the matrix. It should be added that these situations do not impose problems for pure phase synchronization, however, it is even there desirable to gain some knowledge about how long a phase will continue. Subsequently, video synchronization will require correct alignment in flat areas as well.

Minima outside the warp path. One seemingly obvious approach for defining a proper end point is taking one of the global minima of the row added last to the

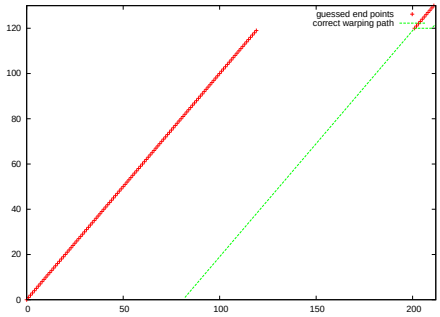

Figure 2. Result of guessed end points (red) vs. offline warp path (green) in flat areas

DTW matrix. As shown in Figure 3 in general this does not provide the correct result. Whereas the minimum would be around 2000 seconds, the offline warp path returns the point marked green, at 3800 seconds. This behavior is due to the

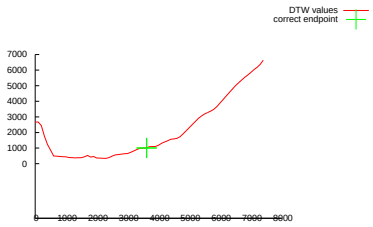

Figure 3. Minimum not matching offline result

fact that the DTW matrix contains the data for all possible alignments. Some of them, however, provide only locally optimal distances, whereas the thereby implied other synchronizations result in higher total distances. When visualized, the valleys of the matrix have a tree-like structure, with its root at the starting point of the sequences. Many branches of the tree yield a steep ascent at some point and thereby stop to be candidates for the correct warp path. These points cannot be known a priori, while doing online recovery. To this end it should be remarked that any online algorithm can be tricked by appropriately constructed data. The DTW based approaches presented in the next Section are particularly vulnerable to such situations.

3.2 DTW Based Approaches

Replacing backtracking by forward search. As backtracking is initially impossible for the online DTW, one possible idea is to replace it by forward search. By starting from the beginning, which is known to be the association of the starting points of both sequences, the warp path is to be found as one of the branches as detailed above. Still, many of the branches might end at some point and thus heuristics need to be employed to select the correct path at each forking.

To clarify the further steps, a few explanations on the warp path are provided. At first, one must be aware of the fact the path only consists of horizontal, diagonal or vertical components, i. e., the slope is either zero, one or infinite. Whereas diagonal parts stand for a 1:1 synchronization of the curves in these parts, horizontal intervals imply that the reference curve contains two or more elements that were matched with a single element of the running process. In other words, the reference curve contains events that could do not exist in the current surgery. Whereas this behavior is desired, vertical parts mean that the reference lacks events, or at least the actions are not known in this context.

Based on the assumption that new events hardly ever occur, forward search can be implemented as shown in Listing 1.1. The threshold variables `hTresh` and `vThresh` direct the search to give preferential treatment to diagonal and, with a very low threshold, horizontal forks of the valleys. Some results obtained using this algorithm are shown in Figure 4. As before, the red marks show the guesses

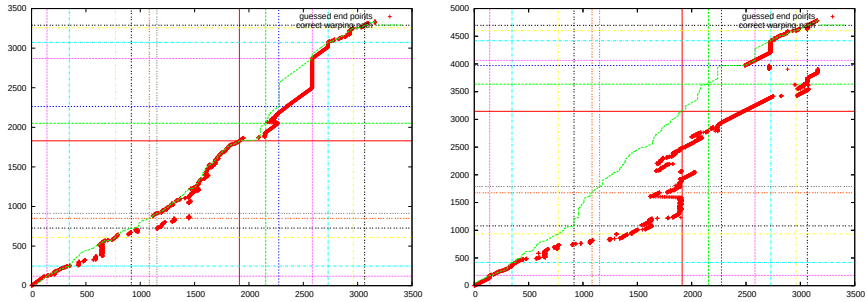

Figure 4. Forward search applied to the second (left) and fourth (right) surgery using OP_{Avg} as the reference.

computed on the fly, whereas the green line is the correct warp path computed offline. While appropriate threshold parameters could be found for the second surgery, all experiments failed for the fourth surgery. Even though a perfect match cannot be expected, at least the transitions of the phases from Table 1 should be detectable. The correct points are marked by the colored horizontal and vertical lines. The crossings of each color denote the point a warp path

Listing 1.1. Forward search

```
/* the threshold for going horizontal vs. diagonal */
const double hThresh = 0.01;
/* the threshold for going vertical vs. diagonal or
   horizontal */
const double vThresh = 1.0;
/* indizes within curve1 (reference) and curve2
  (online) */
int i = 0;
int j = 0;
while( j < curve2.size() - 1 && i < curve1.size() - 1 )
{
  /* assume the next minimum is diagonal ahead */
  double nextMin = dtwMatrix[ i + 1 ][ j + 1 ];
  int next_i = i + 1;
  int next_j = j + 1;
  if( dtwMatrix[ i + 1 ][ j ] < nextMin - hThresh )
  {
    /* horizontal step seems to be more appropriate */
    nextMin = dtwMatrix[ i + 1 ][ j ];
    next_i = i + 1;
    next_j = j;
  }
  if( dtwMatrix[ i ][ j + 1 ] < nextMin - vThresh )
  {
    /* vertical step seems to be more appropriate */
    next_i = i;
    next_j = j + 1;
  }
  /* add next_i, next_j as next step of warp path */
  i = next_i;
  j = next_j;
}
```

Listing 1.2. Path cost analysis

```
/* initialize minimum */
double min_val = dtwMatrix[ 0 ][ curve2.size() - 1 ] +
  pathLength[ 0 ][ curve2.size() - 1 ];
int min_i = 0;
for( int i=0; i < curve1.size(); ++i )
{
  if( dtwMatrix[ i ][ curve2.size() - 1 ] +
    pathLength[ i ][ curve2.size() - 1 ] < min_val )
  {
    min_val = dtwMatrix[ i ][ curve2.size() - 1 ] +
      pathLength[ i ][ curve2.size() - 1 ];
    min_i = i;
  }
}
```

should come over as the vertical lines denote the times of phase transitions in the average surgery, and the horizontal lines those in the second or, respectively, fourth surgery.

Besides the experimental results, a few algorithmic shortcomings are to be noted. At first, this approach highly depends on the value of the threshold parameters, which have been found empirically. Given new data, they might be inappropriate. On the other hand, if the path found by the search ends in a point which is by other means not suitable, probably because of its high DTW value, no way of modifying the path or marking the wrong fork is available. To overcome the latter issue, a slightly different approach is explained next.

Finding the end point via analysis of the warp path The local minima of the last row of the DTW matrix provide a good starting point for backtracking, even though the correct point might still be outside any minimum. The latter is the case for horizontal parts of the warp path, but the incurred error seems acceptable. To mimic the forward search and its emphasis on diagonal and horizontal parts of the warp path, backtracking from all the minima could be started and the best path be chosen, according to the thresholds. In this sense, the thresholds can be interpreted as a distance and thus be summed up to a path length. For an effective implementation of such an approach it should be added that the path length can be computed together with the DTW matrix and stored in a similar table. This reduces the additional cost of computing the path length significantly. In Listing 1.2 the algorithm is sketched. Note, that the decision is based the DTW value and the path length as relying only on the latter showed problems in some cases. The results, as seen in Figure 5, are better than before, but not yet satisfying. The algorithm may be slightly modified by either weighing the DTW and the path value, but also by replacing the less-than

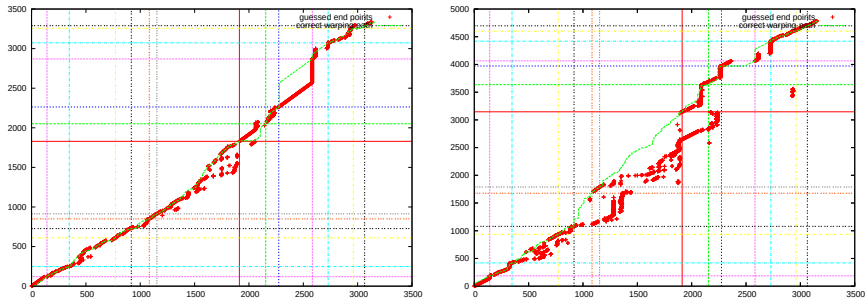

Figure 5. Results of path cost analysis based approach applied to the second (left) and fourth (right) surgery using OP_{Avg} as the reference.

comparison by a less-or-equal operator. However, such experiments did not yield better overall results.

One of the major remaining issues is that the reference curve lacks many features, especially surgery number four includes many unrepresented events. To improve the quality of the attempts, a new reference model has been developed.

3.3 The Exhaustive Model

A proper match can only be expected if any order of all possible events is part of the reference model. We call such a model an *exhaustive model*. Of course, especially in the setting of surgeries, not all possible actions can be anticipated. On the other hand, the likelihood of all events at a specific point in time should be represented in the model. These goals are achieved via the following steps: The six surgeries are grouped in pairs, one of each of the pairs is elected as the reference and the warp paths are computed accordingly. If vertical parts are detected in the paths, the corresponding intervals are inserted into the reference and their state vectors are multiplied by 0.5 to denote the smaller likelihood. The same modification of values is applied to all horizontal parts. The artificial surgeries obtained thereby are then combined to pairs again until a single surgery results. It is guaranteed to contain all possible events with an appropriate weighting.

As the exhaustive model becomes artificially long due to the number of events contained therein, the meaning of "point in time" diverts to "in this context". Furthermore, many actions may never occur in any real surgery. To find the correct end point despite these, the algorithm for computing the longest common subsequence deemed appropriate. The according theory is explained in the next section.

3.4 Longest Common Subsequence (LCSS)

As a special case of edit strings (cf. [2]), the longest common subsequence of A and B can be computed as follows [8]:

$$\mathrm{LCSS}(A,B) = \begin{cases} 0, & \text{if } A \text{ or } B \text{ is empty} \\ 1 + \mathrm{LCSS}(A_{m-1}, B_{n-1}), & \text{if } a_n = b_m \\ \max(\mathrm{LCSS}(A_{m-1}, B), \mathrm{LCSS}(A, B_{n-1})), & \text{otherwise} \end{cases}$$

Before discussing the algorithm in detail, let us explain what *subsequence* means. In contrast to a *substring*, a subsequence of a word is obtained by dropping any number of characters at arbitrary points. Thus any substring is also a subsequence, but not vice versa. As an example consider the word *abbaca*. A possible subsequence is *aaa*, but this is not a substring, as *bba* would be.

The recursive definition of LCSS calls for an implementation using dynamic programming. This works the same way as it does for DTW, based on a matrix. The result shall be illustrated using the same example input as for DTW, as seen in Table 5.

2	1	2	2	3	3	**4**	4
4	1	1	2	3	3	3	3
3	1	1	2	**3**	3	3	3
3	1	1	**2**	2	2	2	2
1	**1**	1	1	1	1	1	1
	1	2	3	3	3	2	2

Table 5. Matrix for $\mathrm{LCSS}(A, B)$

The corresponding relation is obtained by selecting entries for matching characters, but at most one entry per row and per column. The marked entries imply the following relation:

$$(1,1), (3,3), (3,3), (2,2).$$

In general, the relation will contain neither all elements of A nor those of B. Furthermore, in the context of process synchronization it is often desirable to stretch or squeeze parts of the trajectories. For the sequences $A = (1,3,1)$ and $B = (1,3,3,1)$ it might be useful to associate the 3 of A with both of the 3's of B. The result of LCSS, however, would simply have one of the 3's dropped.

Despite these possible disadvantages we will discuss the usefulness of LCSS in the context of online synchronization shortly.

Using LCSS to find the end point. Note, that LCSS does not demand too many modifications to be used within the context of online problems, because it will simply skip parts of the sequence that can't be matched, as is the case with the remaining portions of the process in comparison to the reference data. However, backtracking using LCSS would not report matches as good as those of DTW, because LCSS does not stretch or squeeze matching intervals appropriately, but would rather drop some points of the longer sequence. Furthermore,

Listing 1.3. Using LCSS to find the end point

```
/* initialize minimum */
double max_lcss = lcssRow[ 0 ];
int end_i = 0;
for( int i=0; i < curve1.size(); ++i )
{
  if( lcssRow[ i ] > max_lcss )
  {
    end_i = i;
  }
}
/* backtrack using dtwMatrix, starting from end_i */
```

it is not obvious that LCSS would provide results similar to those of the offline DTW, which we aim for.

The experiments using the algorithm in Listing 1.3 showed immediate success, as seen in Figure 6. Very similar results are obtained for all other surgeries

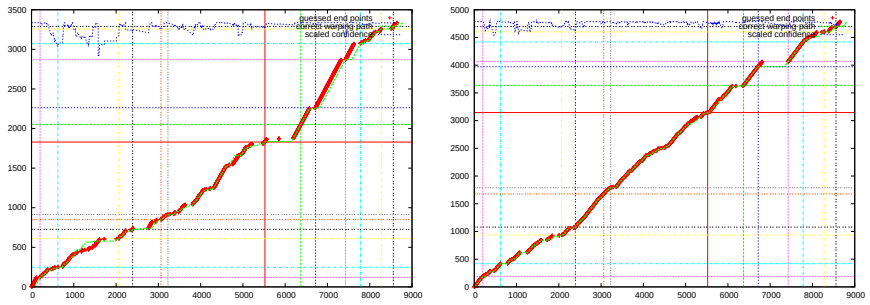

Figure 6. LCSS based approach applied to the second (left) and fourth (right) surgery versus the exhaustive model. The blue lines describe the scaled confidence level.

as well. The details of the implementation shall now be given. As explained above, DTW and LCSS can be adopted for online use by computing a row each time. However, the LCSS matrix need not be kept completely as this would unnecessarily extend the memory requirements, but rather only the last row needs to be preserved. Another issue is the distance and equality function, which must be appropriate for the input data. Note, that probabilities are implemented by setting one or more of the tools to a value between 0 and 1. This, however, may invalidate common ways of measuring the distance between two data points, even similarity is not obvious anymore. Despite this, we still rely on euclidean

distance for the computation of DTW as changes in the metric did not provide any different results for our data. Testing for equality, however, is performed as follows: Two vectors are not equal, if any of the corresponding entries differ by exactly 1, i. e., one of them is 0 and one is 1. Else they are assumed to be equal.

Once the end point has been found, backtracking can be performed from the end point as usual. This is not only necessary output of the procedure, but may also be used to reason about the past, i. e., previous guesses. Thereby an idea of the quality can be gained. Still, it is impossible to use this measure to improve the quality of the next guess. Our way of obtaining the statistics is based on comparing the current warping path to the previous ones. The confidence increases whenever parts of the previous paths are met again. The initial confidence of the new guess is set to the average of the previous path. The pictures in Figure 6 also include scaled graphs of the confidence level. While the details of these statistics remain to be refined, they already provide a reliable idea to the algorithm and the user on whether to trust the guesses or not.

4 Results

Using the LCSS based approach, only one phase transition in the fifth surgery could not be detected correctly. In total, the average overall time error ranged from 20 seconds to 163 seconds in the reference time line, which is two to three times longer than any real surgery. The according simulations to compute the paths took approximately 1 minute per surgery on a 1.5 GHz system, but up to 400 MB RAM were needed, which is due to the size of the DTW matrix. Consequently, this algorithm allows for real time operation on current standard desktop systems.

Despite these promising results, new surgeries are expected to contain new features. As such experiments using modified exhaustive models were conducted. These models were obtained by combining only four surgeries to a new reference, which was then tested against one of the missing surgeries. The expected errors are shown in Figure 7. This result was found for the fifth surgery using the exhaustive model of the first four surgeries as the reference. To alleviate such problems, two workarounds are proposed. On the one hand, no guess may skip workflow phases. Under the hypothesis that these phases are fixed and must be contained in any surgery in the order as listed in Table 1, this is an invariant and should not decrease the quality of any approach. The result is seen in the left picture of Figure 8. Further improvement is easily found by adding a threshold parameter that guards the DTW value of the guess reported by LCSS in comparison to the global minimum of the last DTW row. Although the results, as shown in the right picture of Figure 8, are even better and the implementation is straight forward, it adds a parameter to the algorithm which can only be set empirically. Still, in contrast to the DTW based approaches, it is only a single parameter which can possibly be refined on the fly, using techniques from machine learning.

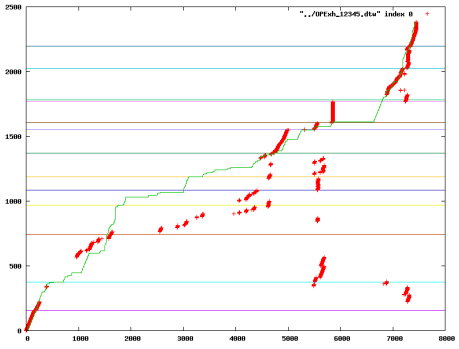

Figure 7. Bad behavior of LCSS based approach in case of missing features

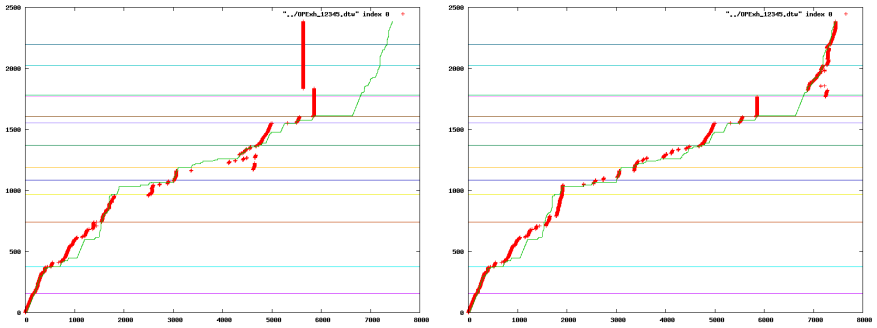

Figure 8. LCSS based approach augmented with phase and threshold guards

5 Conclusion

The feasibility of online recovery has been proven, assumed an appropriate reference curve is provided. This condition should be emphasized and further research on improvements of the exhaustive model may even allow for better results. However, if such a model is not available, a single threshold parameter helps to ensure proper quality of the guesses. This parameter has been found empirically, but in the future means of machine learning should provide an automated approach.

References

1. K. Cleary, H. Y. Chung, and S. K. Mun. OR 2020: The Operating Room of the Future. *Laparoendoscopic and Advanced Surgical Techniques*, 15(5):495–500, 2005.
2. D. Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge University Press, 1997.
3. A. Kassidas, J. F. MacGregor, and P. A. Taylor. Synchronization of batch trajectories using dynamic time warping. *AIChE Journal*, 44(4):863–875, 1998.
4. H. C. Lin, I. Shafran, T. E. Murphy, A. M. Okamura, D. D. Yuh, and G. D. Hager. Automatic Detection and Segmentation of Robot-Assisted Surgical Motions. In J. S. Duncan and G. Gerig, editors, *MICCAI*, volume 3749 of *Lecture Notes in Computer Science*, pages 802–810. Springer, 2005.
5. J. Rosen, M. Solazzo, B. Hannaford, and M. Sinanan. Task decomposition of laparoscopic surgery for objective evaluation of surgical residents' learning curve using hidden Markov model. *Computer Aided Surgery*, 7(1):49–61, 2002.
6. H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.
7. T. Sielhorst, T. Blum, and N. Navab. Synchronizing 3D Movements for Quantitative Comparison and Simultaneous Visualization of Actions. In *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05)*, pages 38–47, 2005.
8. R. A. Wagner and M. J. Fischer. The String-to-String Correction Problem. *Journal of the ACM*, 21(1):168–173, 1974.
9. K. Wang and T. Gasser. Alignment of curves by dynamic time warping. *Annals of Statistics*, 25(3):1251–1276, 1997.