Synchronizing 3D movements for quantitative comparison and simultaneous visualization of actions

Tobias Sielhorst Tobias Blum Nassir Navab

Technische Universität München, Boltzmannstr. 3, 85748 Garching b. München, Germany
E-mail: sielhors|blum|navab@cs.tum.edu

Abstract

In our poster presentation at ISMAR'04 [11], we proposed the idea of an AR training solution including capture and 3D replays of subtle movements. The crucial part missing for realizing such a training system was an appropriate way of synchronizing trajectories of similar movements with varying speed in order to simultaneously visualize the motion of experts and trainees, and to study trainees' performances quantitatively.

In this paper we review the research from different communities on synchronization problems of similar complexity. We give a detailed description of the two most applicable algorithms. We then present results using our AR based forceps delivery training system and therefore evaluate both methods for synchronization of experts' and trainees' 3D movements. We also introduce the first concepts of an online synchronization system allowing the trainee to follow movements of an expert and the experts to annotate 3D trajectories for initiation of actions such as display of timely information. A video demonstration provides an overview of the work and a visual idea of what users of the proposed system could observe through their video-see through HMD.

1. Introduction and related works

The presented work can be applied for different tasks of reproduction, synchronization and comparison of movements in AR. The environment of our research is medical education.

1.1 AR Birth Simulator

The "Klinik für Orthopädie und Sportorthopädie r.d. Isar" has developed a birth simulator consisting of a haptic device in a body phantom and software that simulates biomechanical and physiological functions [7]. The position of the 3D model is visualized on a screen and audio output is generated. While still in development process, it already represents a delivery simulator that provides multimodal functionality (see figure 1). The long term goal of the proposed delivery simulator is offering a device for improved training in order to reduce the amount of cesarean sections as well as the number of perinatal deaths.

As a first step we have combined augmented reality vision with the birth simulator. The user can see the virtual images of what should happen inside the simulator. Images of the baby's head and the hip bone can be seen inside the phantom, which used to be on a screen next to the phantom (see figure 2).

The augmented reality system we use is the research system RAMP. It has been developed by Siemens Corporate Research (SCR) for real time augmentation in medical procedures [10]. It is optimized for accurate augmentation regarding relative errors between the real and the virtual scene. The system features a high resolution video see through HMD and infrared inside-out tracking. Its accuracy, its high resolution, high update rate and its little lag is currently state of the art.

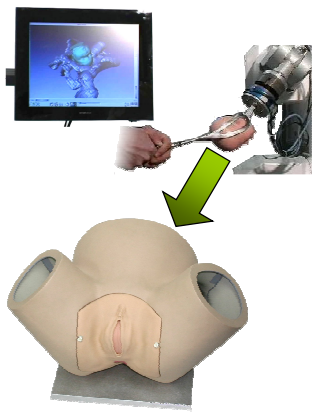


Figure 1. Delivery simulator: Virtual simulator on the screen, phantom head on robot's arm, female body phantom

Digital Object Identifier (DOI): 10.1109/ISMAR.2005.57

1.2 AR + Forceps issue

The first functional addition to the delivery simulator after the combination with the AR system was the visualization of the forceps. The forceps are an instrument which are used for real deliveries in the critical case of a birth stop. The baby's head has to be drawn in order to support the birth to prevent undersupply of oxygen.

In order to visualize instruments inside the phantom and record the users movements, the 6DOF of the forceps must be tracked. There are different options for this tasks. For two reasons we decided not to take the same tracking system as the AR system does. First, there is the line of sight problem. The targets for the single camera tracking of the AR system are likely to be occluded since it needs about eight markers [12] for sufficiently accurate, reliable and robust tracking. These are needed for each of the two forceps parts. Second, the error function of targets of the single camera tracking is unequally distributed in space [12]. The least accuracy is in the orientation of the view. For real-time augmentation this is fine, because less accuracy is needed in viewing direction for a satisfactory overlay that includes a minimal mean error in the 2D image [4]. We intend to record the movements of the instrument and visualize it from potentially any direction. Therefore we have chosen an external tracking system that tracks with a more equally distributed error function and uses a minimal set of markers per target in order to estimate position and orientation. Practically we use a multiple-camera infrared tracking system by A.R.T. which uses retro-reflective markers like the inside out tracking of the AR system. This detail made it simple to register one tracking system to the other. Both systems can therefore be registered using a common reference target. Thus, we can provide the instrument's position in the coordinate system of the head tracking target that is tracked by both systems (see figure 3).

In order to transfer the coordinates of the forceps, which

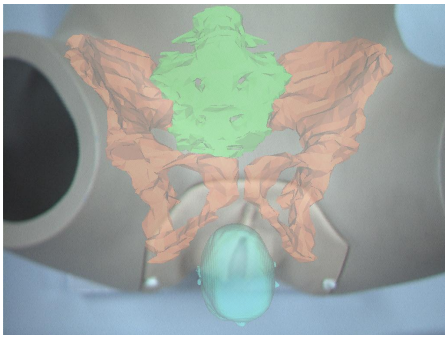


Figure 2. Augmented view of the delivery simulator

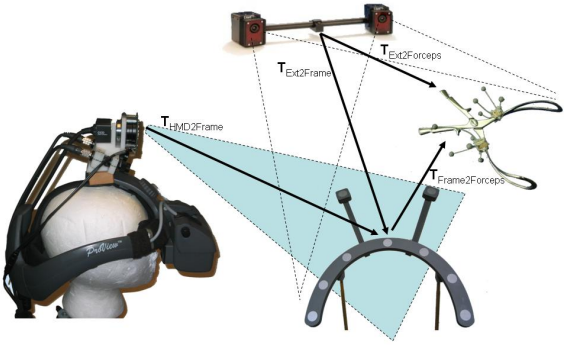


Figure 3. Inside out and outside in tracking

are provided by the outside-in tracking system, into the coordinate system of the marker frame, the simple equation

$$T_{Frame2Forceps} = T_{Ext2Forceps}.T^{-1}_{Ext2Frame}$$

can be applied where T is the matrix notation of the translation and rotation in homogeneous coordinates. This is possible because both tracking systems use the same model data for the marker frame, and both systems generate the coordinate system in the same way from the model data. With this combination, the system is highly dynamic and robust. With this setup we can move the cameras of the external tracking system while the system is running, without any need for re-calibration.

1.3 Omnidirectional viewing

For optimal learning of complex spatial movements and tasks, the desirable procedure is an expert demonstrating actions and giving feedback to the practicing students immediately. In general, the schedule of experienced experts is tight. This makes it difficult to demonstrate the task to each individual student and provide him/her feedback. In addition, it would be desirable to allow the majority to learn from the best international experts in their field. We propose an AR system that learns from the expert by tracking his movements while he/she uses a simulator or performs a real (often complicated) task. This information is reproduced for demonstration to students in an enhanced simulator. By comparison of the experts and students performance, direct feedback is provided.

Dosis et al. [1] record kinematic and visual data of a surgery for assessment of surgical skills. They use electro magnetic tracking with 3DOF for data acquisition for tracking the instrument tip. For displaying the position of the instrument they use video capturing. We intend to visualize the experts movements into the real world. We cannot use a video stream, because we want to allow the student to be able to look from any direction.

Reproduction is, in AR systems, a particularly challenging task if the viewpoint is not fixed.

Cheok et al. [8] capture 3D real time content for insertion of dynamic content into mixed reality. They use 14 cameras to capture dynamic content. Views from other viewpoints than the cameras are generated pixelwise. For our solution we need additional quantitative data of the movement of the expert or his/her instruments in order to compare it with the movements of the student to show and measure differences (see next section).

In order to assess comparable data and visualize the movements in an AR system we track the object and later visualize its 3D model.

The movements of the expert can be visualized while the students work with the simulator. This allows them to try to imitate the expert and it provides a permanent feedback whether the action is correct or not.

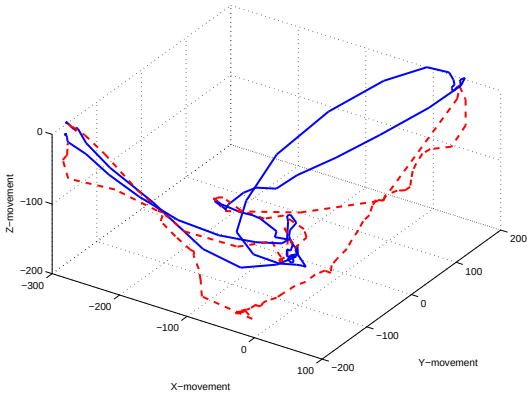
Since the system assesses quantitative data of the performances of expert and student it would be interesting to find out how to compare the performances electronically to have an objective measure.

1.4 Comparison of trajectories

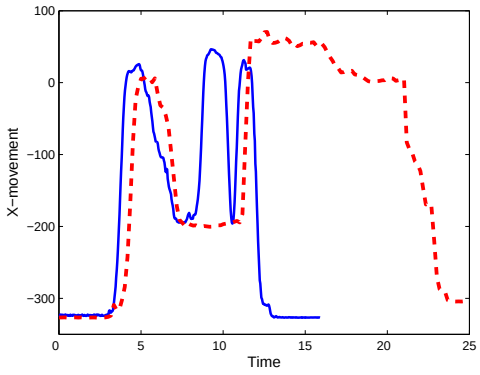
In order to compare two performances of the same action we want to

- be able to visually compare two trajectories either done by professionals or by a student and a professional. For this purposes we want to replay two previously performed and recorded motions synchronously, using AR to have an omnidirectional viewing on both which helps to identify and study subtle differences.
- get a similarity measure between two previously recorded trajectories to quantitatively measure and automatically judge the performance of a student who tried to reproduce the movement of a professional

Figure 4(a) shows the movement of a tracked instrument when trying to perform the same motion twice. A straight forward approach to get a similarity measure between both would be to use the euclidean distance at chosen points in time, which turns out to be inadequate. This can be seen in figure 4(b), which shows the x-movement over time. The motions were performed at a different speed and so the similarity measure would state them as very dissimilar. Even scaling them to the same length as done in figure 5 would not lead to a satisfying result, because the speed at which both tasks are performed will most likely change during execution. The same applies to our second objective. When simply starting a replay of both movements at the same time, we would not be able to see subtle differences since they are shown at a different speed. Just as well scaling will



(a) trajectories A and B



(b) trajectories A and B over time

Figure 4. (a) shows the same motion performed twice, (b) shows the x-movement of the same trajectories over time

not give us a replay where both motions are shown synchronous.

Another simple approach is to take every point from the first trajectory A, find the geometrical closest point in the second trajectory B and base the similarity measure on this distances. Again this would not deliver a satisfying result as the temporal order of the trajectories would be disregarded. To point out the problem with this method one should think about measuring the similarity between the same motion performed once forwards and once backwards. This simple similarity measure would see these two movements as equal. In general this similarity measure provides strong response as soon as as there exist similar parts within the trajectories.

Besides, this does not help us in our task to show synchronized replays where the temporal order must be preserved.

So we need a method that on one hand can handle speed variation without downgrading the similarity measure and on the other hand accounts for the temporal order. Furthermore we need a synchronization of both trajectories so that

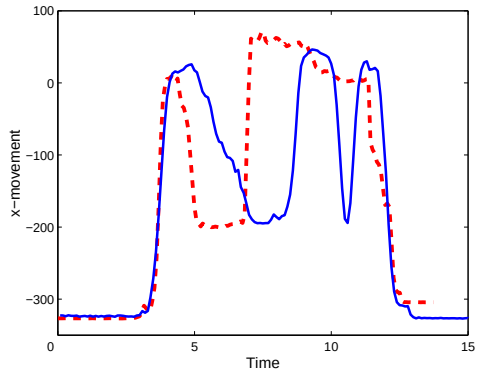


Figure 5. performance time of trajectory B has been scaled down to fit A

speed variation between them are removed and similar parts are shown at the same time, whereas of course the temporal order is preserved.

1.5 Comparison of values

This can be posed as a problem, where we have two trajectories $A = (a_1, \ldots, a_m)$ and $B = (b_1, \ldots, b_n)$, a_i being the data we got at a certain time, in our case containing a time stamp, location and rotation of the tracked object. We need a time-invariant similarity measure S(A, B) and a monotone mapping between the points of both, $w = ((i(1), j(1)), \ldots, (i(K), j(K)))$ such that one trajectory is synchronized to the other one. Functions i and j define the mapping between the elements of the two series. This mapping w can also be seen as a warping function or warping-path, that is applied to the time-axis of one trajectory and synchronizes this to the other one.

At this point we refrain from giving a mathematical definition of a time invariant similarity since different approaches also use different definitions and we do not want to tie ourselves down to one at this point.

1.6 How to match points

In order to quantitatively compare trajectories we have to match points from one trajectory to another. This problem is similar to registration tasks, where we only want to register one dimension. This registration problem cannot be solved using an approach analog to rigid or affine registration, as such transformations could not deal appropriately with the synchronization. Attempts to solve this kind of problems using landmark based registration [2] suffered from the problem to determine the landmarks which is a time consuming and error-prone task. Some applications only need a time-invariant similarity measure and no mapping. Li Zhai Zeng et al. [5] suggest an algorithm that is based on a similarity measure using SVD to do gesture recognition, however it does not return a mapping between both trajectories. More promising to fulfill our requirements are non-rigid registration techniques, which have the drawback of being very slow. Other applications that require registration in only one dimension and can exploit constraints use methods that are similar to non-rigid registration but take less time to compute. There are Dynamic Time Warping (DTW), which is well known in speech analysis [9] and has been used in statistics [14] and signature verification [6], and the Longest Common Subsequence (LCSS) that has been used for similarity measures between mobile object trajectories[3]. The last two are the appropriate ones for our application. The next two sections provide detailed description of these methods.

1.6.1 Longest Common Subsequence (LCSS)

A subsequence S of the set A is a sequence of the form (a_{n_r}), $r \in \mathbb{N}$, where n_r, $r \in \mathbb{N}$ is strictly increasing. More intuitively spoken, you can get the subsequence S by dropping some points of A. The LCSS is better known for obtaining a similarity measure between two strings by computing the longest common substring. The version for two trajectories shares the same idea and defines similarity as a high number of points that are common to both trajectories and have to be in the same temporal order. When computing common subsequences for strings we simply look for characters that are elements of both strings. With 3d points it is very unlikely that we find points that are included in both movements. For this reason we see two points a_n and b_m as equivalent if their distance d(a_n, b_m) is below some chosen threshold ϵ.
Let $A_i = (a_1, \ldots, a_i)$ and $B_j = (b_1, \ldots, b_j)$.

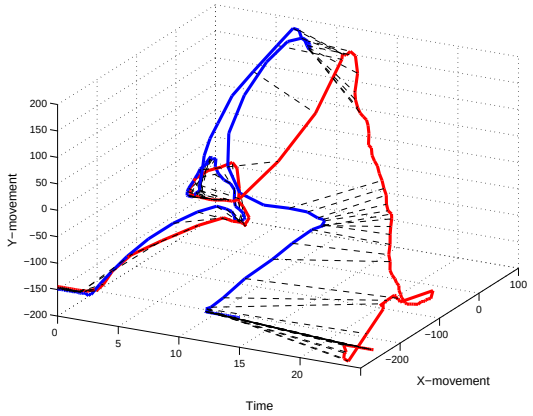
DEFINITION 1. *Given a distance-function d(x,y) an integer δ and a real number ϵ, the $LCSS(A, B)_{\delta,\epsilon}$ is defined as:*

$$LCSS(A, B)_{\delta,\epsilon} =$$

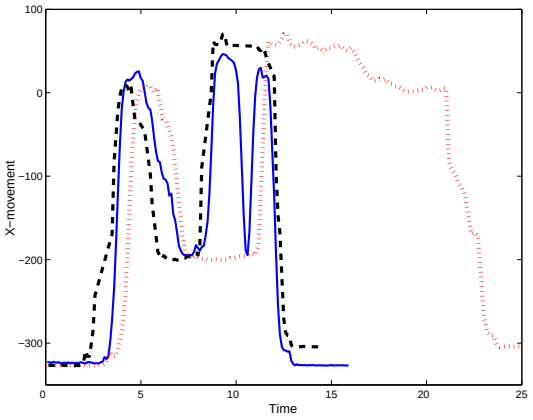
$$\begin{cases} 0, \text{if } A \text{ or } B \text{ is empty} \\ 1 + LCSS(A_{m-1}, B_{n-1}), \text{if } d(a_n, b_m) < \epsilon \text{ and } n - m \leq \delta \\ Max(LCSS(A_{m-1}, B), LCSS(A, B_{n-1})), \text{otherwise} \end{cases}$$

where δ defines a matching window that limits how far in time we search for matching points. The output is the length of the longest common subsequence, i.e. the number of matchings that are possible.

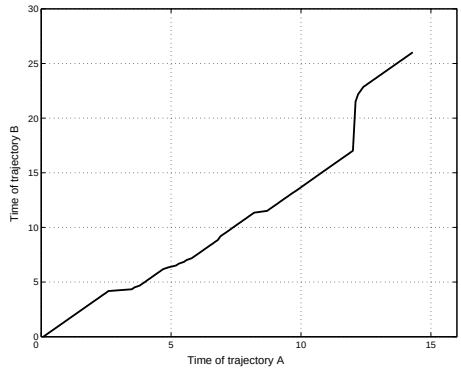
The recursion can by explained as follows. The first conditional value is the termination criteria. The second conditional value can be interpreted as: If the last points of both trajectories, a_m and b_n, are closer than the matching threshold ϵ, we correlate these points, memorize the matching by increasing the result by one and continue by computing the



(a) mapping between A and B



(b) dotted trajectory is synchronized to solid trajectory, synchronized one is dashed



(c) warping path

Figure 6. (a) shows two trajectories and the mapping between them, we got from LCSS, (b) shows the same trajectories and one synchronized to the other, (c) shows a warping path that synchronizes B to A

	T	R	A	I	N	I	N	G
R	0	**1**	1	1	1	1	1	1
I	0	1	1	**2**	2	**2**	2	2
N	0	1	1	2	**3**	3	**3**	3
G	0	1	1	2	3	3	3	**4**
I	0	1	1	**2**	3	**4**	4	4
N	0	1	1	2	**3**	4	**5**	5
G	0	1	1	2	3	4	5	**6**

Figure 7. matrix filled up when computing LCSS of two strings

longest common subsequence of the rest of both trajectories. Otherwise we have to leave either the point a_m or b_n unmatched, depending on what gives us the higher result.

This recursive definition is top-down, and starts the computation on both complete trajectories, lessen them in each step. This problem can also be solved using a bottom-up dynamic programming approach that has a computational complexity of O(δ(n+m)). The corresponding algorithm fills up a n by m matrix row-wise or column-wise, where in each step(i,j) the LCSS(A_i, B_j) is computed, based on the results of LCSS(A_{i-1}, B_{j-1}), LCSS(A_i, B_{j-1}) and LCSS(A_{i-1}, B_j). For simplification the example in figure 7 shows the matrix that is used to compute the LCSS of two strings. The bold numbers show characters that are common to both strings. Each field(i,j) of the matrix contains LCSS(A_i,B_j). So the LCSS('TRAINING','RING') would be 4 and LCSS('TRAIN','RING') is 3. If only a distance measure and no warping path is required only values from the last and the actual step must be kept in memory.

Since the value LCSS computes depends on the length of both trajectories, we need to normalize the output. We define the similarity function derived from LCSS as follows:

DEFINITION 2. *The similarity function $S1_{\delta,\epsilon}$ based on the LCSS between two trajectories A and B is defined as follows:*

$$S1_{\delta,\epsilon} = \frac{LCSS(A,B)_{\delta,\epsilon}}{min(m,n)}$$

After computing the LCSS, a mapping can be obtained that connects all points that are included in the longest common subsequence both trajectories share. To get this mapping the matrix LCSS created during computation must be back tracked by starting at the lower-right field. If the upper or the left neighbor contains the same value as the current field you enter this one. Otherwise you store this point as common and enter the upper-left neighbor. This is done until you reach the upper-left field of the matrix. In figure 7 this would give you the common substring 'RINING',

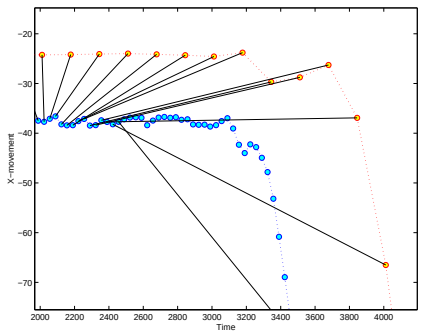


Figure 8. Problems of LCSS when two trajetories have different update rates

the correspondent algorithm for trajectories would give you mapping between points.

Figure 6(a) shows two trajectories that have been recorded by an optical tracking system when trying to perform the same movement twice. The lines connecting both trajectories represent the mapping we got from LCSS. Figure 6(b) shows the same trajectories and one synchronized to the other. Although only movement in x-direction is shown in this figure, it has been computed regarding the 3-dimensional distance, but not the rotation of the object. The warping path we got from the point correspondences is shown in figure 6(c).

Since LCSS is based on points that are similar in both trajectories we only get a mapping that includes similar points. Parts of the trajectories that are too distant are skipped. As we want to see the differences between both movements, we cannot just skip these parts in our replay. This problem can be overcome by interpolating between points that have been mapped.

As long as both trajectories have been recorded with similar update rates LCSS performs well and provides us with a meaningful similarity measure as well as with a quite accurate and very smooth synchronized replay. Due to the normalization that is applied to the similarity measure, different update rates will not affect this measure. But the synchronized trajectory tends to often run ahead.

This happens because for LCSS it does not matter which points are mapped to each other, as long as the maximum possible number of points are matched. An example can be seen in figure 8, where trajectory A has a higher update rate and the earlier points are assigned to every point of trajectory B that is within range ϵ. As the LCSS by definition does not care how these points are matched, it is not possible to find a matching that is more reasonable for our case using only the similarity definition of LCSS.

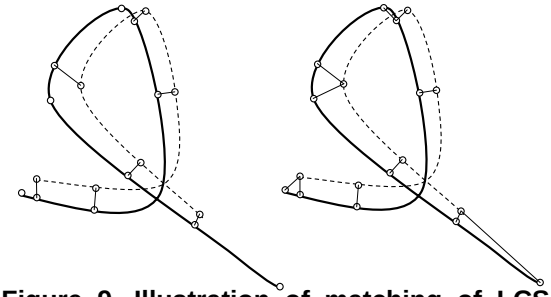


Figure 9. Illustration of matching of LCSS (left) and DTW (right)

1.6.2 Dynamic Time Warping (DTW)

In contrast to LCSS the DTW has to match every point with at least one point of the other trajectory, in particular the first points of both trajectories must be matched to each other just as well as the last points. The differences between both are illustrated in figure 9. All distances between points, that are matched to each other, are summed up. DTW computes the matching that has the lowest summed up distance concerning a given distance function d(x,y). This can be defined recursively as follows.

DEFINITION 3. *Given a distance-function d(x,y), the DTW(A,B) is defined as:*

$$DTW(A,B) = d(a_n, b_m) +$$

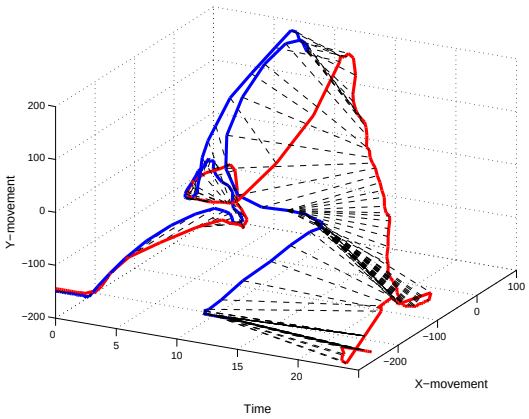
$$min\left(DTW(A_{m-1}, B_{n-1}), DTW(A_{m-1}, B), DTW(A, B_{n-1})\right)$$

In each recursion step the last points of both trajectories have to be matched to each other. Either both points, or only one of them is left out in the next step, depending on what produces the lowest result. DTW can also be computed using dynamic programming in a way very similar to LCSS and takes also time of O(δ(n+m)) when using a matching window of δ, and another step with complexity O(n) afterwards to get the warping path w between A and B. Just as with the LCSS a matrix is filled up with the result of DTW(A_i, B_j) in field(i,j), which is acquired by computing d(a_i,b_j) and adding the minimum of the left, upper and upper-left field. To get the matchings you start at the lower-right field. In each step you store the points a_i, b_j as correspondent and proceed either with field(a_{i-1}, b_j), field(a_i, b_{j-1}) or field(a_{i-1}, b_{j-1}) whichever is smaller.

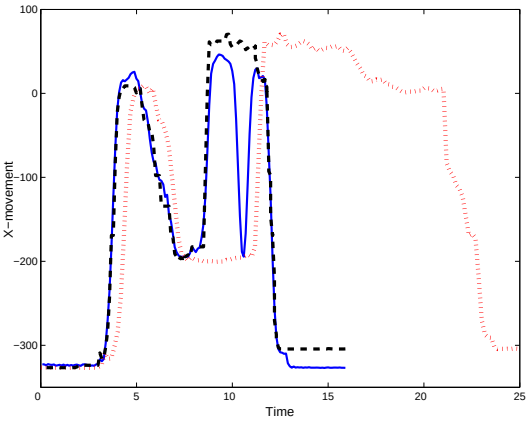
Figure 10 shows the matchings we got, one trajectory synchronized to the other one and the warping function.

Since all points have to be matched, outliers could have a too intense impact on how points are matched. To deal with this, we used a robust measure that restricts the maximum distance between two points.

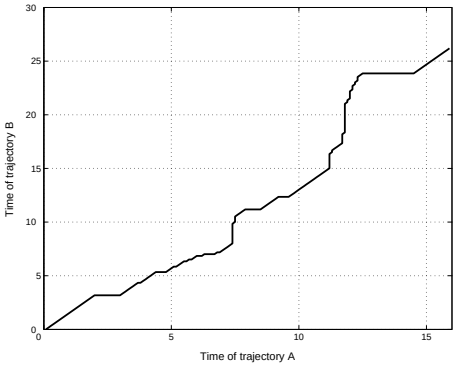
Because the value DTW delivers depends on the number of correspondences, we have to define a normalized similarity function by:



(a) mapping between A and B



(b) B (dotted) synchronized to A (solid), synchronized trajectory is shown dashed



(c) warping path

Figure 10. (a) shows the mapping computed with DTW,(b) shows both trajectories and one synchronized to the other, (c) shows a warping path that synchronizes B to A

DEFINITION 4. *The similarity function S2 based on the DTW between two trajectories A and B is defined as follows:*

$$S2 = \frac{DTW(A,B)}{|w|}$$

where $|w|$ is the number of matchings, which is slightly higher than max(m,n).

Such as with LCSS we get problems, when one trajectory has been recorded with a higher update rate. Since every point has to be matched at least once, we would have to correlate points from the shorter trajectory with multiple points from the longer one. This would lead to a snatchy replay of the synchronized movement but can easily be dealt with. When one point has multiple correspondent points we only take one of them into account and drop the other ones. When both trajectories have about the same size, points that are matched to several points are rare. Therefore this will not cost us too many points. In general we can keep nearly min(m,n) pairs of points.

2 Online synchronization

DTW can be computed as one of the two trajectories is recorded, however it will not give you the similarity measure until the algorithm has ended. The warping path is also is acquired as a following step. In our application it could be very useful to get a mapping that would synchronize a trajectory that is currently recorded to a reference trajectory. This would allow us to show a reference motion while a student tries to imitate it, in which by synchronizing this reference motion to the students motion, we could adapt the speed of the reference motion to the students performance.

Another use would be to automatically execute defined actions at certain points of a workflow. So we could, for example, turn on or turn off augmentations or show certain informations only for a period of time. This could be done by recording a reference workflow and assign an action to a certain point in time on the reference trajectory. By synchronizing the actual performed motion we could estimate when to carry out the action.

Within our video demonstration (see section 3) we show an exemplary illustration of this new concept. The user associates the action to a particular point of the reference 3D trajectory. The action is performed when the trainee's synchronized motion reaches the appropriate position within his/her online trajectory. To achieve such an online synchronization we considered that we could use the intermediate data of DTW to get a preliminary result. Figure 11 shows the matrix that is generated when computing the DTW. On the z-axis the result of DTW(A_x,B_y) is drawn. The path shown in the figure is the warping path that is required to synchronize both trajectories. It is obtained by

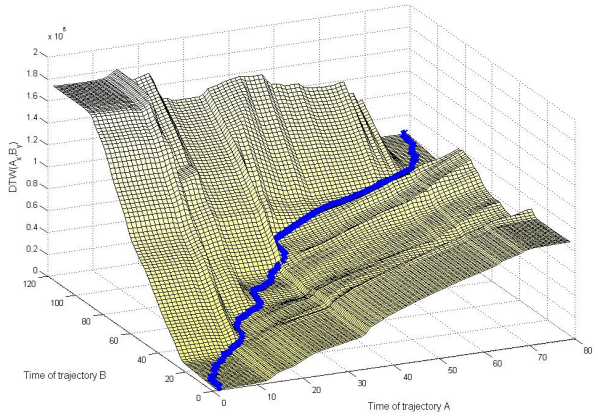


Figure 11. Matrix with intermediate data from DTW

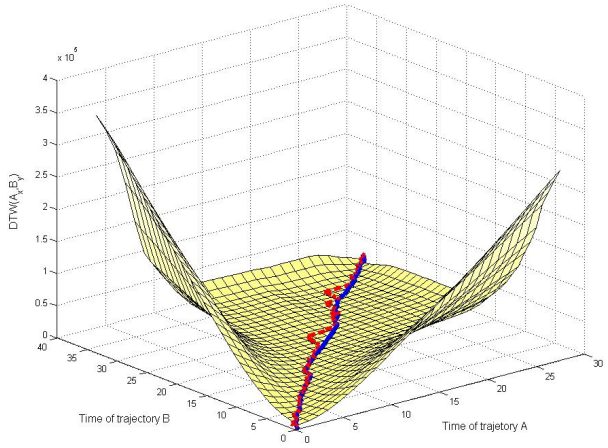
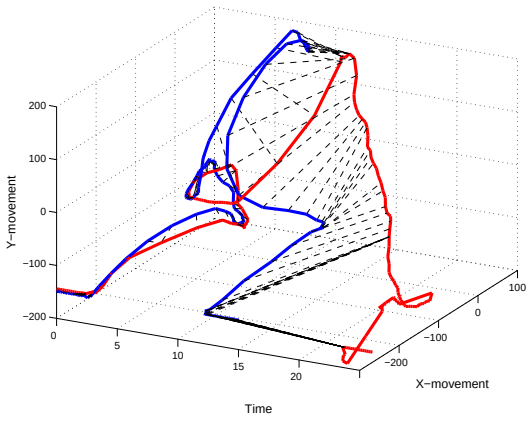


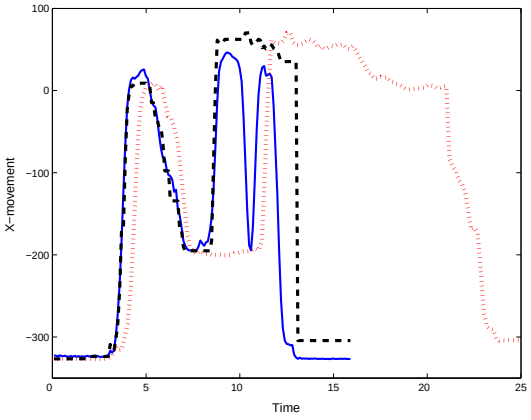
Figure 12. warping path derived from intermediate data DTW delivered (solid was computed online, dashed offline)

backtracking the path from the upper-right corner to the origin. When trying to get a mapping while one trajectory is not finished yet, we cannot use backtracking for the reason that we would not know at which field of the matrix to start. But as we can see in figure 11 the warping path tends to take a course close to the minimum of every row. In figure 12, the warping path for another DTW matrix, obtained by backtracking and the path obtained by selecting the minimum of DTW(A_i,B_1) .. DTW(A_i,B_n) in each step i is shown. This mapping computed while recording one trajectory, will diverge somewhat from the afterward computed one, especially when we have flat areas like in figure 12. But in general it delivers a synchronization that does not vary too much from the offline computed one.

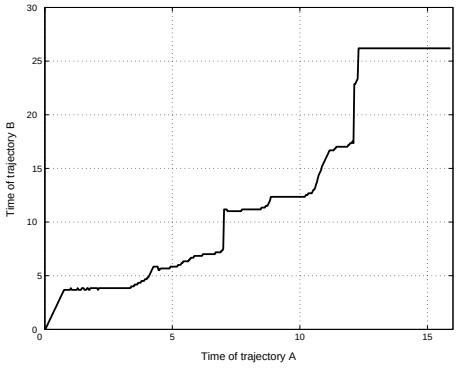
Figure 13 shows an online warping. In order to show comparable results in this figure, we did not synchronize a movement that was recorded during computation. Instead



(a) mapping between A and B



(b) B synchronized to A



(c) warping path

Figure 13. (a) shows the online computed mapping between A and B,(b) shows the online synchronization, (c) shows a warping path

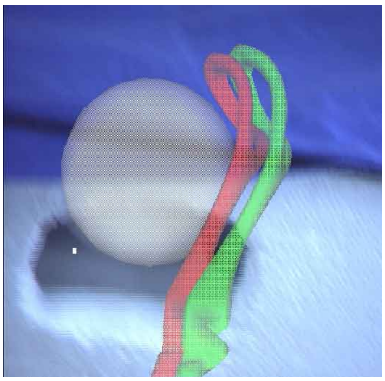


Figure 14. replay of two forceps that have been synchronized

we used the same two trajectories as we used for offline synchronization. Although our first results using an online synchronization are promising, it is not assured to work in every case and will surely be less reliable than offline synchronization. Further testing and supposably some modifications will be necessary.

3 Results

For a better understanding of our results we recommend to have a look at the video on our website. [1] We discovered that finding a time-invariant similarity measure between two 3d-trajectories and a synchronization in time of both can not be solved using simple approaches like the one shown in figure 5. However there are problems which are similar like speech recognition or signature verification, and can be solved using Dynamic Time Warping or Longest Common Subsequence. We implemented both methods and evaluated them for our application. LCSS is able to provide a similarity measure and a mapping that synchronizes one trajectory to the other, as can be seen in figure 6. Problems arise when both movements have been recorded using different update rates. (see figure 8) DTW is also capable of providing a similarity measure and a synchronization as in figure 10. Problems with different update rates can be overcome by a minor change. As we implemented it in the delivery simulator we could test it in an application, where it has shown to provide an appropriate synchronization. Figure 14 shows two forceps that have been synchronized using DTW. While performing the same action twice you can see that user follows a slightly different trajectory.

Real-time dynamic synchronization of user's action with a previously recorded one is more challenging. We did a first step towards solving the problem. For this we computed the warping based on the intermediate data of DTW which is shown in figure 12. The results are not expected to be as good as with offline synchronization (see figure 13 and compare with figure 10) but the errors have been within a low range. These results are promising but further work has to be done on this before it can be applied. The results of offline synchronization and comparison of movements is ready to be further tested in applications.

3.1 Possible development

In our implementation we only took into account the position of the tracked object and disregarded orientation. The effect of including rotation or motion of multiple points on one object should be examined.

Also we used only a basic implementation to test the general ability of DTW to solve our problems. In statistics and speech analysis the DTW is widely known, and several modifications and extensions have been done. So [13] and others proposed different cost functions, that could improve the result, and should be considered. Another common task in statistics is to build average curves based on a large number of sample curves, which can be solved by a method using DTW [14]. It can be thought of building an average trajectory and analyzing which parts of a workflow are common to all samples and where we can find varieties. Further testing of online warping would be necessary before an application can be built on this.

4 Discussion/Conclusion

In this paper, we present systems and methods which allow us to synchronize and compare sequences of captured 3D movements. The method is applied to an AR system designed for training of physicians and midwives. The DTW gives us a similarity measure that could be used to automatically rate the performance of a trainee when trying to replicate a movement. Using only the position of a single point of the forceps is for a real application obviously not enough, but it helps understanding the problem of object motion matching with varying velocities. Since the algorithm uses an arbitrary distance measure we can easily involve other aspects of the action such as orientation of the object, velocity, or even biomechanical data (e.g. measured force) from the delivery simulator. Defining a meaningful distance measure is a non-trivial task. For instance, involving orientation the question arises how to represent the physical state of the object and how to define the distance such that it makes sense within our application. In the case of forceps, we believe that the use of multiple points, minimum of three non collinear ones, makes more sense than using position and orientation. In addition, motion of various parts of the tool have different effects on the overall result of the action. Therefore, we aim at defining the distance as *weighted* sum of distances of corresponding points.

[1] http://campar.in.tum.de/pub/Sielhorst2005Synchronizing3D/Sielhorst2005Synchronizing3D.video.avi

For example, parts of the tool which moves inside the patient will be weighted higher than the outside parts. This is one of the subjects of our current research and development. The synchronization process is essential because it not only provides an initial estimate for the performance of the users but also enables us to measure user's performance based on other, often more important, parameters. Force is one of the crucial parameters to take into account, speed, acceleration and angular velocity could also be considered when providing a measure of similarity/performance. Advanced visualization and HCI design could allow us to provide intuitive user interfaces to visualize these parameters and provide detailed measure of success. This is another subject of our current research and development.

In fact even a system with a well designed similarity measure can currently not replace the supervisory comments of a professional. The whole action is very complex. It has parts of different importance. An experienced supervisor knows the crucial parts and can include his knowledge into his judgment of trainee's performance. This additional high level knowledge is not yet included in our system. A possible approach for finding the importance of parts is to acquire many sequences done by different experts and use the matching algorithm proposed in this paper to find statistically significant common parts of the action, which in general are the most crucial parts. Alternatively, the expert could annotate a reference sequence for labeling crucial parts. The automatic matching proposed here would then allow us to transfer this knowledge to trainees.

Nevertheless the warping path that can be obtained when computing the similarity measure is very valuable, as it gives us the possibility to visualize both movements simultaneously and have a look at the differences between two actions. Since we use augmented reality to replay the synchronized movements, we are able to examine them from different viewpoints and analyze them in a much more tangible way than video recordings would offer. Note, that the performance velocity is not lost: The warping path (fig. 10(c)) contains the relative velocity of both motions. It can be used for visualization as mentioned in the previous paragraphs. Also our implementation allows to change the speed at which the two trajectories are shown, stop them or rewind the replay through which a very detailed analysis of the trajectories is possible. In addition, an online synchronization would also be useful as one could provide timely information or execute actions at predefined points of a workflow.

Since this is an Augmented Reality application the best way to appreciate the current achievement is to visualize the results through the HMD. Our video demonstration presents some of the images taken by the HMD camera and provides some impression of what the final users observe.

Acknowledgments: We would like to thank Frank Sauer, Ali Khamene, and Sebastian Vogt from Siemens Corporate Research in Princeton, USA for the courtesy of providing us with the AR system RAMP. We would like to thank A.R.T. GmbH, Herrsching, Germany for the courtesy of their multiple camera tracking system. Furthermore, we thank Rainer Burgkart and Tobias Obst, Klinikum Rechts der Isar, Munich, Germany for their courtesy of their Delivery Simulator.

References

[1] A. Dosis, F. Bello, K. Moorthy, Y. Munz, D. Gillies, and A. Darzi. Real-time synchronization of kinematic and video data for the comprehensive assessment of surgical skills. In *Medicine Meets Virtual Reality (MMVR)*, 2004.

[2] T. Gasser, A. Kneip, A. Binding, A. Prader, and L. Molinari. The dynamics of linear growth in distance, velocity and acceleration. *Annals of Human Biology*, 18(3):187–205, 1991.

[3] D. Gunopoulos. Discovering similar multidimensional trajectories. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, page 673. IEEE Computer Society, 2002.

[4] W. A. Hoff and T. L. Vincent. Analysis of head pose accuracy in augmented reality. *IEEE Trans. Visualization and Computer Graphics*, 6, 2000.

[5] C. Li, P. Zhai, S.-Q. Zheng, and B. Prabhakaran. Segmentation and recognition of multi-attribute motion sequences. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 836–843. ACM Press, 2004.

[6] M. E. Munich and P. Perona. Camera-based id verification by signature tracking. In *ECCV '98: Proceedings of the 5th European Conference on Computer Vision-Volume I*, pages 782–796. Springer-Verlag, 1998.

[7] T. Obst, R. Burgkart, E. Ruckhäberle, and R. Riener. The delivery simulator: A new application of medical vr. In *Proceedings of Medicine Meets Virtual Reality 12*, 2004.

[8] S. Prince, A. D. Cheok, F. Farbiy, T. Williamson, N. Johnson, M. Billinghurst, and H. Kato. 3d live: Real time captured content for mixed reality. In *Proc. IEEE and ACM International on Mixed and Augmented Reality (ISMAR)*, 2002.

[9] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust. Speech Signal Process.*, 26(1):43–49, 1978.

[10] F. Sauer, A. Khamene, and S. Vogt. An augmented reality navigation system with a single-camera tracker: System design and needle biopsy phantom trial. In *Proc. Int'l Conf. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2002.

[11] T. Sielhorst, J. Traub, and N. Navab. The ar apprenticeship: Replication and omnidirectional viewing of subtle movements. In *Proc. IEEE and ACM International on Mixed and Augmented Reality (ISMAR)*, 2004.

[12] S. Vogt, A. Khamene, F. Sauer, and H. Niemann. Single camera tracking of marker clusters: Multiparameter cluster optimization and experimental verification. In *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 127–136, 2002.

[13] K. Wang and T. Gasser. Alignment of curves by dynamic time warping. *Annals of Statistics*, 25(3):1251–1276, 1997.

[14] K. Wang and T. Gasser. Synchronizing sample curves nonparametrically. *Annals of Statistics*, 27(2):439–460, 1999.