



LAWRENCE  
LIVERMORE  
NATIONAL  
LABORATORY

# Tracking Non-rigid Structures in Computer Simulations

A. Gezahegne, C. Kamath

January 11, 2008

2008 IEEE International Conference on Image Processing  
San Diego, CA, United States  
October 12, 2008 through October 15, 2008

## **Disclaimer**

---

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

# TRACKING NON-RIGID STRUCTURES IN COMPUTER SIMULATIONS

*Abel Gezahegne and Chandrika Kamath*

Lawrence Livermore National Laboratory  
Livermore, CA 94551

## ABSTRACT

A key challenge in tracking moving objects is the correspondence problem, that is, the correct propagation of object labels from one time step to another. This is especially true when the objects are non-rigid structures, changing shape, and merging and splitting over time. In this work, we describe a general approach to tracking thousands of non-rigid structures in an image sequence. We show how we can minimize memory requirements and generate accurate results while working with only two frames of the sequence at a time. We demonstrate our results using data from computer simulations of a fluid-mix problem.

*Index Terms*— tracking, non-rigid structures, correspondence problem

## 1. INTRODUCTION

Tracking of moving objects is a problem which has long been studied in domains such as surveillance, computer vision, and meteorology. More recently, tracking techniques have also been used in molecular bioimaging [1], combustion experiments [2], and computational fluid dynamics simulations [3], where they are used to track non-rigid structures over time. A key task in tracking is finding the correspondence between structures at consecutive time steps. This can be very challenging if there are a large number of structures which change shape over time, appear and disappear, and merge and split.

In this paper, we describe an approach to tracking coherent structures in simulations of the Rayleigh-Taylor instability (RTI) [4]. We first describe the data set and discuss the challenges in addressing the correspondence problem. We then describe our solution approach, followed by the results using sample images from our data.

## 2. PROBLEM DESCRIPTION

The Rayleigh-Taylor instability occurs when an initially perturbed interface between a heavier fluid on top of a lighter fluid is allowed to grow under the influence of gravity. The

fingers of lighter fluid penetrate the heavier fluid in what are referred to as ‘bubbles’, while ‘spikes’ of heavier fluid move into the lighter fluid. With time, these structures, which are initially distinct, continue to evolve; in the process, they may grow, split, merge with surrounding structures, or shrink in size relative to other structures which grow and overtake them.

In earlier work [5, 6] we used image processing techniques to analyze the data from three dimensional high-fidelity simulations of RTI. Our goal was to count the structures of interest. We first converted the three-dimensional data to two dimensional images, and then obtained the bubble (spike) counts by exploiting characteristics of the  $x$ - and  $y$ - velocities and the height (depth) at the bubble (spike) surfaces. In the process of incorporating a constraint on the  $z$ -velocity to define rising bubbles and falling spikes, we found that our approach could also be used to track the bubble and spikes over time, providing us insights into the dynamics of these structures.

Figure 1 shows a  $450 \times 200$  subset of a  $3072 \times 3072$  image at time steps 52, 54, 56, and 58. The variable shown is the magnitude of the  $x$ - $y$  velocity at the bubble surface - this is small at the bubble center and around the bubble perimeter (as indicated by the darker pixels). The pixels highlighted in red have a  $z$ -velocity smaller than  $-0.15$ , those in green have a  $z$ -velocity greater than  $+0.15$ , and the remaining pixels are in yellow. The pixels in dark blue at the center of many of the bubbles are pixels whose magnitude of the  $x$ - $y$  velocity is less than  $0.15$ . By our definition, green regions, which contain at least one dark blue pixel, form a rising bubble.

In the rest of this paper, we will focus on the rising bubbles; the tracking of the falling spikes is similar.

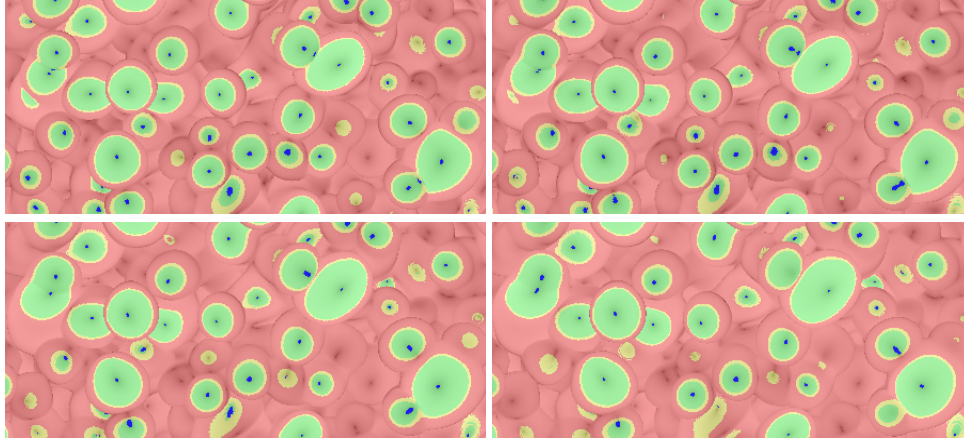
## 3. CHALLENGES TO THE TRACKING

Our goal is to track the rising bubbles over a sequence of 248 frames (corresponding to time steps in the simulation), each of size  $3072 \times 3072$ . There are over 7000 rising bubbles at early time, which finally reduces to less than 100 as the bubbles merge or fade away.

The evolution of the rising bubbles in green in Figure 1 illustrates several challenges in tracking. The bubbles come in different shapes and sizes. Several bubbles in the lower left corner gradually shrink and eventually disappear. Some, like the one in the lower right, stop being a rising bubble as they

---

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.



**Fig. 1.**  $450 \times 200$  pixel subset of the images to be used for tracking the rising bubbles, indicated by green regions containing dark blue pixels. From left to right, top to bottom - time steps 52, 54, 56, and 58.

no longer have blue pixels with a low magnitude of the  $x$ - $y$  velocity. The bubbles may merge completely, such as the two in the upper left corner, or merge for a while and then split like the two near the upper right corner. In the process, one of the rising bubbles may stop being a rising bubble as is the case in the lower right corner. We have also observed one bubble merging with another, which in turn is merging with a third, and, one bubble merging with a second bubble, which is splitting from a third bubble.

A solution to the tracking of non-rigid structures which merge and split is proposed in [2]. Our problem differs in two key ways. First, two bubbles can merge for a short while and then split to become two distinct bubbles again. In the process, they should keep the labels they had originally. This means we need to record how long two bubbles have been merging; if this is greater than a certain predetermined number of time steps,  $t_{merge}$ , we consider them to be merged. Second, the size of our images is very large, with a large number of structures. Therefore, any tracking approach should be computationally efficient and should not require us to process more than two images at a time to solve the correspondence problem. We accomplish this by maintaining a history of bubbles which are merging so that they can be assigned the correct label in case they split before  $t_{merge}$  time steps.

#### 4. SOLUTION APPROACH

To track the rising bubbles, we start by assuming that each green pixel in the image at time  $(t-1)$  is correctly assigned a permanent label. The green regions in the image at time  $t$  are also labeled using a connected component algorithm. However, these labels are temporary and serve only to identify the connected regions in the image. Our task is to correctly assign labels to the pixels in the green regions at time  $t$  so that an object at the two time steps has the same label.

Solving the correspondence problem can be viewed as

laying the image at time  $t$ , with its temporary labels, on top of the image at time  $(t-1)$ , with its permanent labels, and propagating the labels from time  $(t-1)$  to  $t$ . Let  $O_i(t)$  be the object with temporary label  $i$  at time  $t$  and  $label(O_i(t))$  be the permanent label assigned to this object during the tracking.

We maintain three types of information to identify merging objects correctly. A *MergeList* for an object  $O_i(t)$  is the list of labels of objects from time  $(t-1)$  which are under it, excluding the largest such object, where the largest is based on the number of overlapped pixels. *MergeMap* and *EraseMap* are data structures used to maintain the history of merging bubbles; they are implemented using the map container from the Standard Template Library (<http://www.sgi.com/tech/stl>). A *MergeMap* entry for an object with label  $k$  contains the label of the object it is merging to and the time step when the merge started. A *MergeMap* entry exists only for objects which are merging to other objects. Once an object completes its merge, its entry is removed from *MergeMap* and moved to *EraseMap*. The former exists across all time steps, while the *EraseMap* and the *MergeList* are specific to each time step.

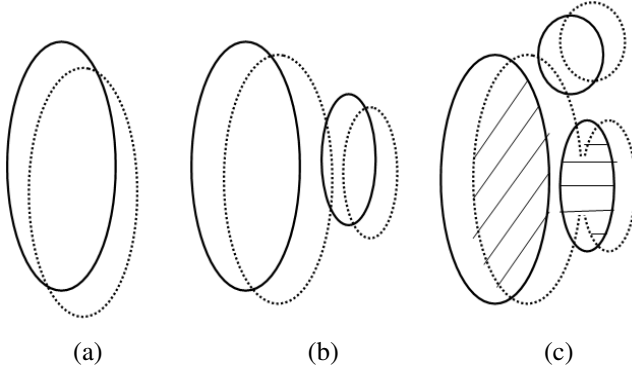
The process of tracking consists of the following steps:

**STEP 1: Identify overlap** - For each object  $O_i(t)$ , determine the labels and the number of overlapped pixels of the objects that lie under it at time  $(t-1)$ .

**STEP 2: Identify object labels** - We have two cases:

**Case 1:** There is only one object, with label  $k$ , under  $O_i(t)$  (Figure 2(a)); set  $label(O_i(t)) = k$ . Pixels of  $O_i(t)$  which overlap this object or the background (i.e., non-object pixels from time  $(t-1)$ ) get assigned its label in Step 4.

**Case 2:** There is more than one object under  $O_i(t)$ . Set  $label(O_i(t))$  to the label of the largest object overlapped by  $O_i(t)$ . The choice of the largest object is a heuristic which worked well in our data; other options are also possible. We also create a *MergeList* for each such object, containing the labels of objects under it, excluding the largest object.



**Fig. 2.** Possible situations in the evolution of bubbles (solid boundary is object at time  $(t-1)$ , dotted boundary is object at time  $t$ ). (a) Single object; (b) an object overlapping two objects due to stray pixels; (c) labeling non-obvious pixels.

**STEP 3: Identify the truly merging objects** - Next, we focus on objects  $O_i(t)$  which have a MergeList associated with them. We want to identify which of the objects in the MergeList is really merging with the current object  $O_i(t)$ . We have several cases:

**Case 1:** An object with label  $k$  in the MergeList exists at time  $t$ . Then, we have the situation in Figure 2(b), where a few stray pixels from object  $k$  at time  $(t-1)$  overlap with  $O_i(t)$ . This is not a true merging situation and we can remove label  $k$  from the MergeList.

**Case 2:** An object with label  $k$  in the MergeList does not exist at time  $t$  and does not have a MergeMap entry corresponding to it. This indicates that the object with label  $k$  has just started merging to the current object. We create a MergeMap entry for this object which includes the label of the current object ( $label(O_i(t))$ ) and the current time step indicating the start of the merge.

**Case 3:** An object with label  $k$  in the MergeList does not exist at time  $t$  but has a MergeMap entry for it, indicating that the two objects have been merging for a while. If this time is longer than  $t_{merge}$ , then remove the merging object entry from MergeMap, add it to the EraseMap, and mark the entry in MergeList to indicate the two objects have merged.

**STEP 4: Label the obvious pixels** - At this point, there are certain pixels in the objects at time  $t$  to which we can assign a label without any ambiguity. These are:

**Case 1:** Pixels which belong to objects which overlap only the background at time  $(t-1)$ : These belong to a new object and a previously unused label is assigned to it, typically by keeping track of the largest label assigned so far.

**Case 2:** Pixels in an object which has no MergeList associated with it: This arises when an object overlaps one object at time  $(t-1)$  and possibly the background or stray pixels from another object which exists at time  $t$ . All these pixels are assigned the label of the object they overlap.

**Case 3:** Pixels in an object which has a MergeList asso-

ciated with it: Let  $k$  be the label of the object at time  $(t-1)$  which has the largest number of pixels overlapped with the current object. Consider the pixels in the current object. If a pixel lies above one labeled  $k$  at the previous time, label it  $k$ . If a pixel lies above one with a label which exists in the current object's MergeList, assign it that label. This correctly propagates the labels of merging objects which have not completed the merge. If a pixel lies above one with an entry in the EraseMap (that is, the object with this label just completed its merge), then identify the object it merged to and assign the current pixel that label. The remaining pixels in the current object lie either above a background pixel or above a stray pixel from an object which exists at time  $t$ . These are labeled in Step 5.

**STEP 5: Label the non-obvious pixels:** When two objects are merging, the pixels of each object must be labeled appropriately so they retain their distinct labels if they do not complete the merge. In the region where the two objects merge, we cannot precisely identify the pixels of each object; as long as the two labels remain distinct, this is not a problem.

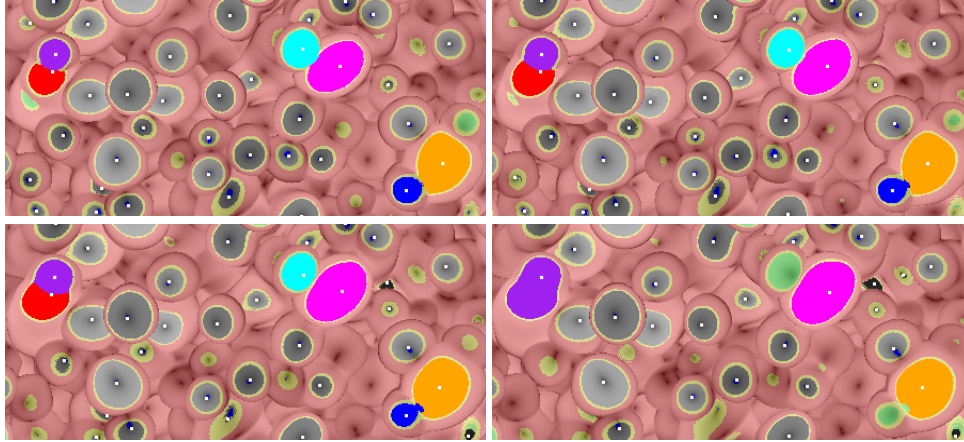
Consider Figure 2(c), where the two oval objects have just started merging. The combined object has pixels which lie over the original objects, the background, and stray pixels from the round object nearby. The pixels which lie over the original objects (indicated by the two shaded regions) are assigned the corresponding labels. Next, we iteratively label the remaining unlabeled pixels inside the two merged dotted ovals. First, we identify all such pixels which are next to a labeled pixel; and then assign them the same label. This grows the boundary of each object by at most one pixel. We repeat the process until all pixels have a label. This ensures that the pixels in the merged object which lie over the round circle, have the label of one of the two ovals, not the circle.

**STEP 6: Labeling structures which have split:** Labeling objects which have split into two is relatively simple as we will have two distinct regions with the same label. In such cases, the higher bubble retains the original label (as we are tracking rising bubbles) and the other bubble gets a new label. Other heuristics are also possible.

#### 4.1. Special cases

There are certain special cases which must be addressed to track the rising bubbles correctly:

- When an object at time  $t$  overlaps more than one object at time  $(t-1)$ , it takes the label of the object with the largest number of overlapped pixels. If at time  $(t+1)$ , a different object satisfies this constraint, we swap object labels so that the tracking is consistent over time.
- If an object A is merging with object B, and B completes its merge with object C, then the MergeMap entry for A is updated to indicate that A is merging with C as object B ceases to exist once it completes the merge.



**Fig. 3.** Results from tracking the bubbles in Figure 1.

- Suppose an object A is merging with an object B which splits into two, labeled B and C. If after the split, A is merging with the part labeled C, then the MergeMap entry for A must be updated appropriately. Similar updates must be done if it is object A which splits into two while merging with B.

## 5. RESULTS

Figure 3 shows the results of applying our tracking algorithm, with  $t_{merge} = 6$ . Regions in green are not counted as rising bubbles as they lack pixels with a low magnitude of the  $x$ - $y$  velocity. Bubbles which do not merge or split are tracked using various shades of grey. Interestingly, two new bubbles (shown in black) start rising at time step 56 (to the right of the purple bubble and in the right corner); another new bubble starts growing near the upper right at time step 58.

The red and magenta bubbles in the upper left corner, which started merging just before time step 52, complete their merge and appear as a single bubble in time step 58. In contrast, the cyan and purple bubbles merge, but split before 6 time steps; the cyan one stops being a bubble at time step 58. Similarly, the dark blue bubble at the bottom right, stops being a rising bubble before it completes its merge to the orange bubble. Note that the boundary between the two merging bubbles is not very clean, and when they split, a few green pixels (indicating a non-bubble) remain in one of them. These pixels are correctly labeled in the next time step.

## 6. SUMMARY AND CONCLUSIONS

In this work, we describe a solution to the correspondence problem in tracking non-rigid structures. Using sample data from a computer simulation, we show how we can handle complicated behavior of the structures such as merging and splitting, as well as splitting before completing a merge. Our technique uses only two frames of the sequence at a time, and

keeps the memory requirements low by maintaining the history of merging structures. The approach is general and is being applied to other problems, including tracking of coherent structures in fusion plasma experiments.

## 7. ACKNOWLEDGMENTS

We thank William Cabot and Andrew Cook for providing access to the data and Paul L. Miller for his support of this work.

## 8. REFERENCES

- [1] E. Meijering, I. Smal, and G. Danuser, “Tracking in molecular bioimaging,” *IEEE Signal Processing Magazine*, vol. 23, no. 3, pp. 46–53, May 2006.
- [2] J. A. Withers and K. A. Robbins, “Tracking cell splits and merges,” in *Proc., IEEE Southwest Symposium on Image Analysis and Interpretation*, 1996, pp. 117–122.
- [3] D. Silver and X. Wang, “Tracking and Visualizing Turbulent 3D Features,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 3, no. 2, pp. 129–141, 1997.
- [4] D. H. Sharp, “An overview of Rayleigh-Taylor instability,” *Physica*, vol. 12D, pp. 3–18, 1984.
- [5] C. Kamath, A. Gezahegne, and P. L. Miller, “Analysis of Rayleigh-Taylor instability, Part I: Bubble and spike count,” Tech. Rep. UCRL-TR-223676, Lawrence Livermore National Laboratory, 2006, <http://www.llnl.gov/casc/sapphire/pubs/TR-223676.pdf>.
- [6] C. Kamath, A. Gezahegne, and P. L. Miller, “Analysis of Rayleigh-Taylor instability: Statistics on rising bubbles and falling spikes,” Tech. Rep. UCRL-TR-236111-Rev-1, Lawrence Livermore National Laboratory, 2007, <http://www.llnl.gov/casc/sapphire/pubs/TR-236111-Rev-1.pdf>.