Real-Time Tracking of Origami with Physics Simulator Considering Fold Lines

Hiroto Arasaki
Department of Mechanical Engineering
Chiba University
24wm4111@student.gs.chiba-u.jp

Akio Namiki
Department of Mechanical Engineering
Chiba University
namiki@faculty.chiba-u.jp

Abstract: We address real-time tracking of paper during robotic folding by integrating fold-line geometry into a physics-based simulator, enabling localized plastic deformation. In experiments using a standard folding sequence, we succeeded in real-time tracking of a square-base folding sequence by combining point cloud information from multiple RGB-D cameras with estimates from the simulator.

Keywords: Deformable object tracking, Origami, Paper manipulation

1 Introduction

Deformable-object manipulation has gained attention as industries seek to automate manual tasks; however, measuring and controlling deformation remains difficult.

One representative line of work is *origami robotics*. In our previous work, Namiki and Yokosawa [1] realized folding motions using a robot hand. However, to track the paper shape they relied on sheets with printed or attached markers, which limits generality. Therefore, it is desirable to develop a method that can track the shape of ordinary, unmarked paper.

For deformable-shape tracking, prior work has studied cloth and string. Xiang et al. [2] and Chi and Berenson [3] tracked filamentary objects without using physics simulation. Tang et al. [4] and Chi and Berenson [3] performed shape tracking using Coherent Point Drift (CPD) [5] for point cloud registration; notably, Tang et al. [6] combined CPD with physics-based correction. Physics-based simulation of deformables has also progressed rapidly [7, 8, 9]. For robotic applications, Li et al. [8] proposed a differentiable cloth simulator, and Yu et al. [10] used it for motion generation. In this study, we extended the methods proposed by Tang et al. [4] and Chi and Berenson [3] to paper and proposed a paper shape tracking method [11]. This system successfully tracked two-fold and four-fold sequence on a real-time simulator by combining CPD-based registration and interpolation. In this paper, we added a function for detecting fold lines and localizing plastic deformation to fold lines, thereby achieving successful tracking of more complex origami. Specifically, we realized high-precision real-time tracking of a square-base folding sequence.

2 System Configuration

Fig. 1 overviews the paper-shape tracking system [11]. We mount Intel RealSense RGB–D cameras at five viewpoints relative to the table: D435 (overhead, rear) and D405 (left, right, below). RGB–D images are streamed to a vision PC (the *Tracker*), while a physics simulation of the paper (the *Simulator*) runs in parallel and exchanges model point clouds and state updates with the Tracker over UDP.

Fig. 2 shows the processing pipeline. The *Tracker* first segments the paper in RGB and masks the depth image to obtain a 3D point cloud of the paper (the *observed* point cloud). It

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

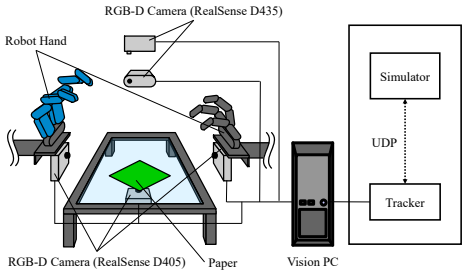


Figure 1: System overview of the paper-shape tracking system. Intel RealSense cameras (D435: overhead, rear; D405: left, right, below) stream RGB–D images to the Tracker; the Simulator exchanges state information with the Tracker over UDP.

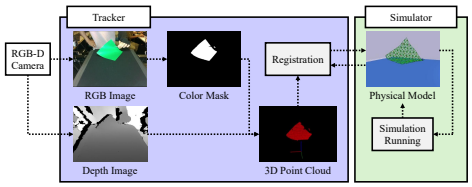


Figure 2: Processing pipeline. The Tracker segments the paper in RGB, masks depth to obtain the observed point cloud, registers the model point cloud from the Simulator using CPD, and returns the update for physics stepping.

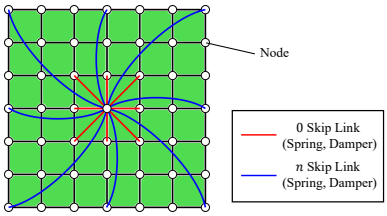


Figure 3: Physical model of the paper: a grid of point masses connected by spring–damper links, including skip links between nodes n steps apart.

also receives the previous model point cloud (the *model* point cloud) from the *Simulator* and performs CPD-based registration to update the model pose. The *Simulator* then advances the physical model using the updated model point cloud.

3 Origami Simulator

This section describes the origami simulator used in the Simulator process. Fig. 3 illustrates the physical model of the paper: point masses arranged on a grid connected by links. Each link comprises an elastic element (spring) and a damping element (damper). Links are placed between adjacent nodes as well as between nodes that are n steps apart (skip links).

3.1 Update rule of the simulation

We adopt Substep Extended Position-Based Dynamics (XPBD) [7], and in this work we use only *distance constraints* between two points $\mathbf{x}_1$ and $\mathbf{x}_2$, which keep the distance at d. The constraint function is $C(\mathbf{x}_1, \mathbf{x}_2) = |\Delta\mathbf{p}| - d$ where $\Delta\mathbf{p} = \mathbf{x}_2 - \mathbf{x}_1$. Let m_1 and m_2 be the masses of the two points. The update rules are given by

$$\Delta\mathbf{x}_1 = \frac{C - \gamma\Delta\mathbf{p}\cdot(\mathbf{x}_1^{\text{new}} - \mathbf{x}_1^{\text{old}})/|\Delta\mathbf{p}|}{m_1(m_1{}^{-1} + m_2{}^{-1} + \frac{\alpha}{\Delta t_s{}^2})}\frac{\Delta\mathbf{p}}{|\Delta\mathbf{p}|}, \tag{1}$$

$$\Delta\mathbf{x}_2 = -\frac{C + \gamma\Delta\mathbf{p}\cdot(\mathbf{x}_2^{\text{new}} - \mathbf{x}_2^{\text{old}})/|\Delta\mathbf{p}|}{m_2(m_1{}^{-1} + m_2{}^{-1} + \frac{\alpha}{\Delta t_s{}^2})}\frac{\Delta\mathbf{p}}{|\Delta\mathbf{p}|}. \tag{2}$$

Here α denotes spring compliance and $\gamma = \alpha\beta/\Delta t_s$ with damper coefficient β.

3.2 Reflecting tracking results in the physical model

Given the model point cloud from the Tracker, the Simulator must update the positions of the physical model's point masses. We treat the received point cloud as a desired position $\mathbf{X}_d$ and apply virtual zero-rest-length springs between $\mathbf{X}$ and $\mathbf{X}_d$ to provide input to the dynamics.

3.3 Fold line detection and plasticity localization

To reproduce folding phenomena, the simulator introduces plasticity into the spring–mass model [12]. A trade-off arises: if plastic deformation is too easy, springs near folds deform largely and tracking accuracy improves, but any mass point may also be affected strongly by input noise, causing failure. Conversely, if plastic deformation is too difficult, the model becomes robust to noise but folding deformation near creases is suppressed, again leading to tracking failure. The key insight is that reliable tracking requires allowing large deformations only in the vicinity of fold lines, while keeping other regions elastic.

To achieve this, we added a function for detecting fold lines as *cluster boundaries*. First, the simulator estimates to which cluster each mass point belongs. For every spring in the physical model, if its endpoints belong to *different clusters*, plastic deformation is applied. If the endpoints belong to the *same cluster*, the plastic strain of that spring is reset to zero and no plasticity is applied. In this way, only springs around fold lines undergo plastic deformation, enabling the simulator to combine accurate tracking of paper folds with robustness against observation noise.

For clustering, each mass point is represented as a 5-dimensional vector $\mathbf{x} = (x, y, z, u, v)$, where (x, y, z) denotes the 3D spatial position and (u, v) are the intrinsic coordinates on the undeformed 2D sheet. We adopt the k-means method, which iteratively updates cluster centroids and reassigns points to the nearest centroid until convergence. Unlike the conventional k-means that relies on Euclidean distance, we employ the Mahalanobis distance to account for anisotropy of the paper surface:

$$d_{\mathrm{M}}(\mathbf{x}, \boldsymbol{\mu}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}, \tag{3}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and covariance of the cluster. Decomposition of $\boldsymbol{\Sigma}$ leads to $\boldsymbol{\Sigma} = \mathbf{W}^\top \boldsymbol{\Lambda} \mathbf{W}$ where $\mathbf{W}$ are the principal axes and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5)$ are the principal components, corresponding to the cluster variances. To encourage clusters to approximate 2D distributions, we replace $\lambda_3, \lambda_4, \lambda_5$ with a small constant S (set to 0.001^2 in this study).

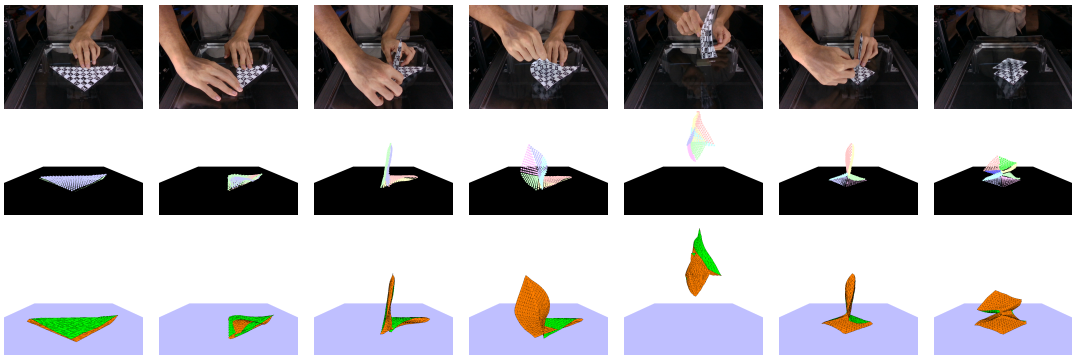
Cluster splitting is triggered when

$$R_{\mathrm{div}} < \sqrt{\frac{\lambda_3 + \lambda_4 + \lambda_5}{\lambda_1 + \lambda_2}}, \tag{4}$$

where R_{div} is a constant (set to 0.13 in this study) and we use $\lambda_3, \lambda_4, \lambda_5$ as before the replacement. Once a cluster has split, further splitting is prohibited until the cluster is regarded as fixed. A cluster is considered fixed when the overall distribution returns close to a 2D sheet: specifically, we perform PCA using only the spatial coordinates $\mathbf{x}_a = (x, y, z)$ and declare the cluster fixed when the third principal component $\lambda_3^{(a)}$ falls below R_{fix}(set to 0.008^2 in this study).

4 Experiments

To validate the proposed method, we conducted real-time tracking experiments with a physical setup. As the target motion, we used a square-base folding sequence, which includes simple half-folds, sheet turnovers, and opening-and-folding operations.



(a) 100 steps (b) 200 steps (c) 225 steps (d) 268 steps (e) 345 steps (f) 460 steps (g) 580 steps

Figure 4: Real-time tracking with ArUco markers. Rows show the RGB frame, the observed point cloud, and the Simulator state.

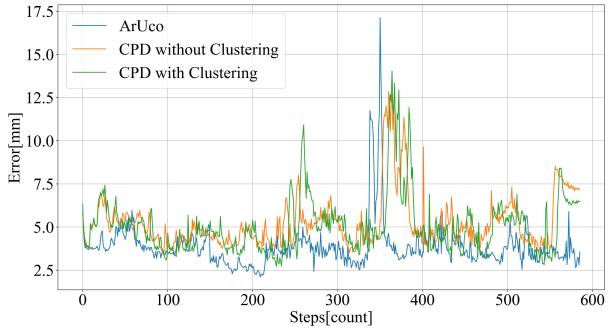


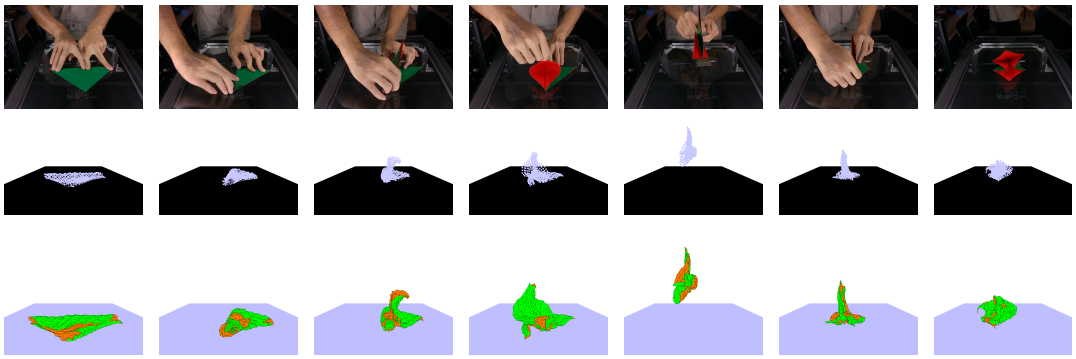
Figure 5: Per-step error E defined in Eq. (5), plotted for three series: ArUco, CPD without clustering, and CPD with clustering. Means across all tracking steps: 3.86 mm (ArUco), 5.26 mm (CPD without clustering), and 5.05 mm (CPD with clustering).

We evaluated two tracking variants in the Tracker: (i) detection of feature points via ArUco markers and (ii) CPD-based point cloud registration. Variant (i) uses a special paper sheet with an ArUco grid and evaluates shape estimation from detected feature points only. Variant (ii) performs CPD-based registration on the observed point cloud (no markers) to evaluate shape estimation for general papers. Computation was performed on a PC with an Intel Core i9-14900KF CPU and an NVIDIA GeForce RTX 4090 GPU. The Simulator ran on the GPU in CUDA C++, and the Tracker ran on the CPU in Python.

4.1 ArUco-based tracking

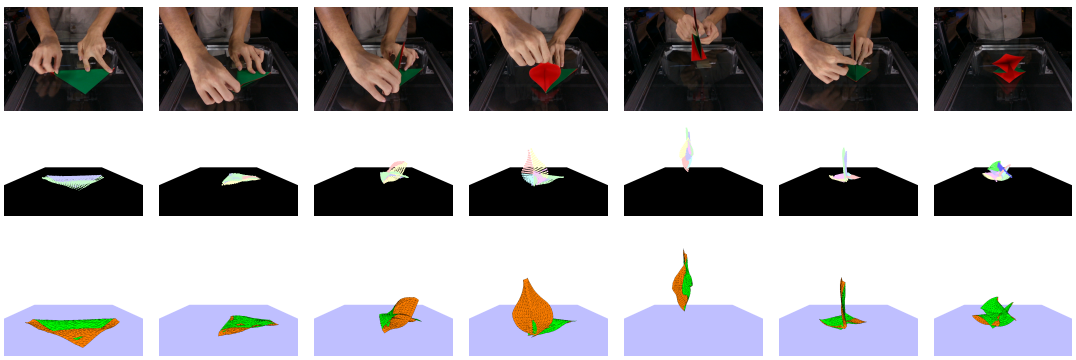
Using paper printed with an ArUco grid, the RGB–D camera detects the positions of feature points (the observed point cloud). We detect 13×13 points and associate them with every other node in the 25×25 physical model grid. Representative tracking results are shown in Fig. 4. Qualitatively, clustering is observed along each fold and plastic deformation remains localized near fold lines. In addition, Fig. 5 plots the per-step one-sided mean nearest-neighbor error E. The error E is computed using Eq. (5), which calculates the error between each point in the observed point cloud Y and its nearest neighbor in the model point cloud X, and then takes the average. The spike near step 350 (ArUco) stems from model tunneling through the sheet. The average processing time was approximately 0.22 s/step.

$$E(Y, X) = \frac{1}{|Y|} \sum_{\mathbf{y} \in Y} \min_{\mathbf{x} \in X} \|\mathbf{y} - \mathbf{x}\|_2 \tag{5}$$



(a) 100 steps (b) 220 steps (c) 255 steps (d) 300 steps (e) 368 steps (f) 470 steps (g) 570 steps

Figure 6: Real-time tracking with Coherent Point Drift (without clustering).



(a) 100 steps (b) 220 steps (c) 260 steps (d) 305 steps (e) 375 steps (f) 470 steps (g) 570 steps

Figure 7: Real-time tracking with Coherent Point Drift (with clustering).

4.2 CPD-based tracking

For CPD we similarly use 13×13 points and associate them with every other node of the 25×25 model. We also compare with a baseline that *does not* perform clustering. Representative tracking results without and with clustering are shown in Fig. 6 and Fig. 7, respectively. Clustering along fold lines is also observed under CPD, and the simulated model exhibits fewer wrinkles with clustering than without. A failure occurs at step 570 (CPD with clustering) due to model tunneling through the sheet; this likely stems from CPD not directly observing each model point as in ArUco. Preventing such failure likely requires either collision handling in the Simulator or stronger geometric consistency constraints in the Tracker. CPD per-step errors are also shown in Fig. 5. The spike near step 570 (CPD with clustering) is caused by the same tunneling event. The average processing time was approximately 0.24 s/step.

5 Conclusion

We presented a real-time paper-shape tracking system that augments a physics simulator with geometric information to induce localized plastic deformation along fold lines. The results of the experiment showed that the simulation model improved its fit to the observed data and successfully estimated a continuous square-base folding sequence. Future work includes preventing model tunneling (e.g., via collision handling) and further improving tracking robustness.

6 Limitations

Our system assumes a fixed multi-camera setup; performance can degrade under severe occlusions or highly localized folding operations (i.e., deformations confined to a small region rather than global motions). The simulator currently uses XPBD using only distance constraints and lacks explicit collision and self-collision handling, which can occasionally lead to self-tunneling of the sheet. Fold-line localization relies on k-means in the (x, y, z, u, v) space, using the Mahalanobis distance, and the hyperparameters (S, R_{div}, R_{fix}) are tuned to our setup; applying the method to different environments may require re-tuning. Our experiments focus on a square-base sequence; extending to a broader repertoire of folds and to faster motions remains future work. On our hardware, the average runtime is 0.22–0.24 s/step, limiting the update rate to a few hertz.

References

[1] A. Namiki and S. Yokosawa. Robotic origami folding with dynamic motion primitives. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5623–5628. IEEE, 2015.

[2] J. Xiang, H. Dinkel, H. Zhao, N. Gao, B. Coltin, T. Smith, and T. Bretl. Trackdlo: Tracking deformable linear objects under occlusion with motion coherence. *IEEE Robotics and Automation Letters*, 8(10):6179–6186, 2023. doi:10.1109/LRA.2023.3303710.

[3] C. Chi and D. Berenson. Occlusion-robust deformable object tracking without physics simulation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6443–6450. IEEE, 2019.

[4] T. Tang, C. Wang, and M. Tomizuka. A framework for manipulating deformable linear objects by coherent point drift. *IEEE Robotics and Automation Letters*, 3(4):3426–3433, 2018.

[5] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010.

[6] T. Tang, Y. Fan, H.-C. Lin, and M. Tomizuka. State estimation for deformable objects by point registration and dynamic simulation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2427–2433, 2017. doi:10.1109/IROS.2017.8206058.

[7] M. Macklin, K. Storey, M. Lu, P. Terdiman, N. Chentanez, S. Jeschke, and M. Müller. Small steps in physics simulation. In *Proceedings of the 18th Annual ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–7, 2019.

[8] Y. Li, T. Du, K. Wu, J. Xu, and W. Matusik. Diffcloth: Differentiable cloth simulation with dry frictional contact. *ACM Trans. Graph.*, 42(1), oct 2022. ISSN 0730-0301. doi:10.1145/3527660. URL `https://doi.org/10.1145/3527660`.

[9] J. M. Pizana, A. Rodríguez, G. Cirio, and M. A. Otaduy. A bending model for nodal discretizations of yarn-level cloth. In *Computer Graphics Forum*, volume 39, pages 181–189. Wiley Online Library, 2020.

[10] X. Yu, S. Zhao, S. Luo, G. Yang, and L. Shao. Diffclothai: Differentiable cloth simulation with intersection-free frictional contact and differentiable two-way coupling with articulated rigid bodies. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 400–407, 2023. doi:10.1109/IROS55552.2023.10341573.

[11] S. Takahashi, H. Arasaki, and A. Namiki. Paper shape tracking system using point cloud and physical model. In *2024 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, pages 229–234, 2024. doi:10.1109/CBS61689.2024.10860574.

[12] H. Arasaki, S. Takahashi, and A. Namiki. Realtime paper shape estimation for origami robot system, 4th Workshop on Representing and Manipulating Deformable Objects @ ICRA2024.