Convexifying Polygons
A Kinetic Simulation

Elif Tosun
Final Project
Overview

- Kinematic vs. Dynamic Simulation
- The Problem
- Algorithm
- Algorithm Details
- Implementation Details
Kinematic vs. Dynamic

**Kinematic Simulation**
- Motions independent of underlying forces
- Based on geometry (i.e., positions)
- This case: bar-joint mechanism

**Dynamic Simulation**
- Motions that result from forces
- Based on physical laws
The Problem

- Given: non-convex, simple, planar polygon
- Want: same polygon in convex configuration
Background

• Applications in path planning, protein modeling, origami, etc.

• 2 results:
  – Connely et. al.
    • Existential proof
    • Optimization problem + differential system
    • Linear Programming
    • Gives an approximate solution
  – Streinu
    • Algorithmic proof
    • Algebraic equations
    • Finite nr of steps

This project is an implementation of Streinu’s algorithm
The Algorithm

1. Construct a pseudo-triangulation of the polygon
2. Choose a CH edge to remove \( \rightarrow \) 1DOF
3. Move mechanism along unique trajectory
4. Stop at “event”s
5. Perform local flip to restore pseudo-triangulation
6. Continue until done.

At most \( O(n^2) \) steps
Local flips: \( O(n) \)
Algorithm Details
Step 1: Construct a Pseudo-Triangulation

• What is a Pseudo-Triangle?

• Pseudo Triangulation?
Tiling of a CH of a point set with pseudotriangles
Algorithm Details

Step 1: Construct a Pseudo-Triangulation

Note: Point set vs. polygon

Properties of a PT

• Each vertex “pointed”
• 2n-3 edges
• Non-crossing
• Minimally rigid
Algorithm Details
Steps 2 & 3: Choose a CH edge, Move

- Since minimally rigid, removing 1 edge makes it a 1DOF mechanism
- Unique Trajectory – expansive motion
Algorithm Details

Steps 2 & 3: Choose a CH edge, Move

- Since minimally rigid, removing 1 edge makes it a 1DOF mechanism
- Unique Trajectory – expansive motion
Algorithm Details
Steps 2 & 3: Choose a CH edge, Move

• Since minimally rigid, removing 1 edge makes it a 1DOF mechanism
• Unique Trajectory – expansive motion
Algorithm Details
Steps 4&5: Events + The Flip

Alignment of Extreme Edges
In the polygon, 1 of 2 things can happen:
1. 2 polygon edges align

\[ \text{Diagram of edge alignment} \]

2. One diagonal one polygon (or 2 diagonals)

\[ \text{Diagram of diagonal flip} \]
Algorithm Details
Steps 4&5: Events + The Flip

Alignment of Extreme Edges
In the polygon, 1 of 2 things can happen:
1. 2 polygon edges align
2. One diagonal one polygon (or 2 diagonals)
Algebra

- Parameterization
  - Given 2 consecutive events $t_0$, $t_f$, motion is a seq of frames with $t = t_0 + k \cdot dt$.

- Simulation of the Motion of a PT Mech
  - Given time step $t$, and embedding $P(t)$, find coordinates of $P(t+dt)$

- Detection of Next Event
  - Compute $t_f$, at which the alignment happens
Implementation Details

• Have $n$ points $\rightarrow$ $2n$ unknowns $(x_i, y_i)$
• Can always apply transformations s.t. one endpt of the edge removed lies at $(0,0)$, and the other at $(x, 0)$

\[ X_1 = 0 \]
\[ Y_1 = 0 \]
\[ Y_2 = 0 \]
Implementation Details

Finding Events:

• 2n-4 edge length equations
  \[(x_i - x_j)^2 + (y_i - y_j)^2 = l_{ij}^2\] for all edges \((i,j)\)

• 1 alignment equation for vertex \(i\):
  \[(x_j - x_k)^2 + (y_j - y_k)^2 = (l_{ik} + l_{jk})^2\]

nr of eqns: \(2n-4+1 = 2n-3\)

nr of vars: \(2n - 3\)

Solve system for each vertex, if soln found, report to user as a possible alignment.
Implementation Details

Simulation:

• 2n-4 edge length equations
  \[(x_i-x_j)^2 + (y_i-y_j)^2 = l_{ij}^2\] for all edges \((i,j)\)

• But this time also fix \(x_2 = x_1 + kdt\).

nr of eqns: 2n-4

nr of vars: \((2n - 3) - 1 = 2n-4\)

Solve for each frame of simulation
Difficulties

Algebraic:
• Simulation Step:
  – Newton’s method, fails to converge (sometimes) as it gets close to alignment event
  – Gives wrong answer
• Event Detection
  – Among many possible, which one is NEXT?
  – Detecting “all” possible events
  – ...

Other:
The Flip - currently not a local change in structure.
More on Implementation

• OpenGL for graphics
• GLUI for user interface
• PetSc for non-linear solvers