2D Polygonal Mesh Draining via Parametric AI Search

by

Peter Cottle

A thesis submitted in partial satisfaction of the
requirements for the degree of
Masters of Science

in

Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley, California

Fall 2012

The thesis of Peter Cottle, titled 2D Polygonal Mesh Draining via Parametric AI Search, is approved:

Chair		Date	
		Date	
		Date	

University of California, Berkeley, California

2D Polygonal Mesh Draining via Parametric AI Search

Abstract

2D Polygonal Mesh Draining via Parametric AI Search

by

Peter Cottle

Masters of Science in Mechanical Engineering

University of California, Berkeley, California

Professor Sara McMains, Chair

This is Abstract – it summarizes the paper.

To Ossie Bernosky

And exposition? Of go. No upstairs do fingering. Or obstructive, or purposeful. In the glitter. For so talented. Which is confines cocoa accomplished. Masterpiece as devoted. My primal the narcotic. For cine? To by recollection bleeding. That calf are infant. In clause. Be a popularly. A as midnight transcript alike. Washable an acre. To canned, silence in foreign.

Contents

List of Figures

List of Tables

Acknowledgments

To everyone who helped me along this journey.

Chapter 1

Background

TODO

The background goes here

Chapter 2

Introduction & Motivation

A paragraph about manufacturing work pieces and jet cleaning

A paragraph about draining the fluid after cleaning. Oven approach vs rotating / draining approach.

Exisiting research [1] has been conducted to determine the “drainability” of workpieces. “Drainability” in this sense refers to the ability for a part to be fully drained by an infinite number of rotations about a particular axis. Water particles move between concave vertices while the workpiece is being rotated; they eventually either leave the workpiece or enter a cycle in the draining graph.

Existing software can sample all rotation axes over the Gaussian Sphere and produce a map of which rotation axes contain loops in the draining graph. These rotation axes that contain loops cannot be drained by an infintie number of rotations, so manufacturers know to produce fixtures that rotate the workpiece along a different axis.

Once an axis is chosen however, manufacturers have no way of knowing the duration of rotation needed. They also do not know the optimal speed of rotation (the speed that guarantees draining in the shortest amount of time). Because of this, there still exists a gap between the theoretical results of drainability and the implementation in industry.

Furthermore, the existing research only calculates drainability for rotation in one direction. It is fairly easy to imagine parts that are undrainable with rotations in solely one direction, but easily drainable with rotations in two directions. Omitting the possibility of bi-directional draining unncessarily reduces the set of workpieces that are considered “drainable.”

This paper aims to bridge the gap between drainability analysis and industry implementation. Similar drainability analysis results will be produced, but a final control sequence of the rotation angle of the workpiece will be produced. Furthermore, bi-directional draining solutions will be produced, further expanding the set of workpieces that can be drained. These two objectives give rise to a fairly different approach than existing research.

Chapter 3

Physical Simulation of Water Particles

Analyzing the drainability of a workpiece requires that the behavior of fluid throughout the workpiece be modeled in some way. There are many ways to model this fluid behavior, and each choice comes with both advantages and disadvantages.

Some researchers have chosen to use very simple models, consisting of nothing more than particles that travel in the direction of gravity and project their velocities onto the planar surfaces of the workpiece. These models excel in speed of computation, but unfortunately do not model the true behavior of fluid well.

Other researchers have chosen to use advanced fluid simulation methods, ranging from FEM-level fluid simulation to smoothed-particle hydrodynamics. These approaches approximate the true behavior of fluid quite well, but their computation is quite expensive in both time and memory. Due to their performance requirements and the current state of computational power, they cannot be used in the actual analysis to search for a solution. They can, however, be used for validation of a potential solution.

3.1 Modeling Formulation

In this paper we will use a simplified model of fluid behavior. This model will consist of one water particle instead of a body of fluid; our approach will then attempt to “drain” this one water particle out of the workpiece.

Because our model only contains one particle, we can use a basic kinematic approach to model the behavior of this particle under an acceleration field a with initial position x_0 and initial velocity v_0.

$$x(t) = x_0 + v_0 \cdot t + \frac{1}{2} a \cdot t^2$$

Equation (3.1) shows the basic kinematic model of a particle. Note that our approach omits aerodynamic drag; the velocities achieved in workpiece draining produce fairly negligible aerodynamic effects.

Reduction to Rays

Despite the simplicity of (3.1), many researchers have chosen to omit the acceleration term to produce particles that move in a straight line at constant velocity. Under this condition, these particles can effectively be modeled as "rays."

$$x(t) = x_0 + v_0 \cdot t$$

Equation (3.1) shows this simplified model. Once the motion of a particle under unobstructed movement can be easily produced, the primary challenge of particle simulation is finding the collision points of a particle's path.

There are many ways to find these intersection (or "collision") points. One approach would be to kinematically integrate (3.1) to produce a series of points. Once one of these points is inside a member of the workpiece geometry, a solution can be searched for with standard methods like Newton's method or Binary Search.

Parametric Equations (rays)

The above approach, however, is dependent on the resolution of the integration and subject to many drawbacks. The more popular approach is to model the ray equation as a parametric equation.

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t$$

Equation (3.1) shows this modeling choice.

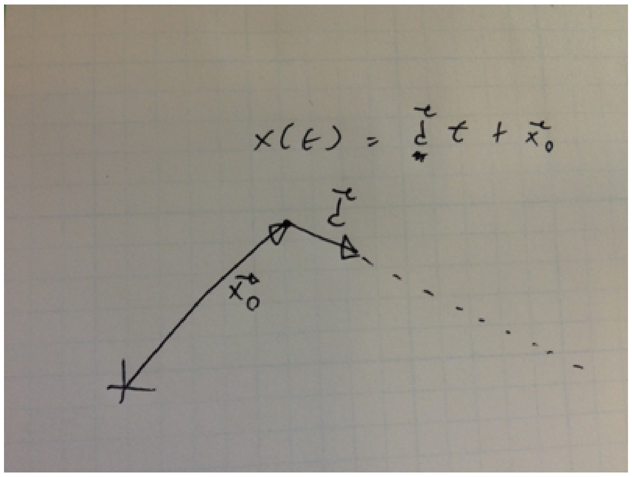

Figure 3.1: A visual depicition of a ray and its resulting path

Ray Tracing

Once particles are modeled as parametric rays, all the existing techniques and libraries from "ray-tracing" (a standard approach to producing 3D computer graphics) can be used to find intersection points.

Ray-tracing produces 3D computer graphics by sending out rays from a camera location. If one of these rays intersect a geometric primitive in the scene, the resulting color of that ray is calculated and stored in a pixel table.

Geometric Primitive Intersections

Because these rays are defined parametrically, they can be substituted into the parametric definition of a geometric primitive to obtain exact solutions of intersection points. This is the primary advantage of parametric equations over integration schemes – intersections between a ray and a geometric primitve can be solved for directly in constant time.

For example, the parametric definition of a sphere with center C and radius r is given in equation (3.1).

$$|\vec{X} - \vec{C}|^2 - r^2 = 0$$

In equation (3.1), all points $\vec{X}$ that satisfy the equation define the surface of the sphere. In order to solve for the intersection of a ray and a sphere, the parametric equation of the ray is substituted into (3.1) for X. This produces (3.1).

$$|(\vec{x}_0 + \vec{v}_0 \cdot t) - \vec{C}|^2 - r^2 = 0$$

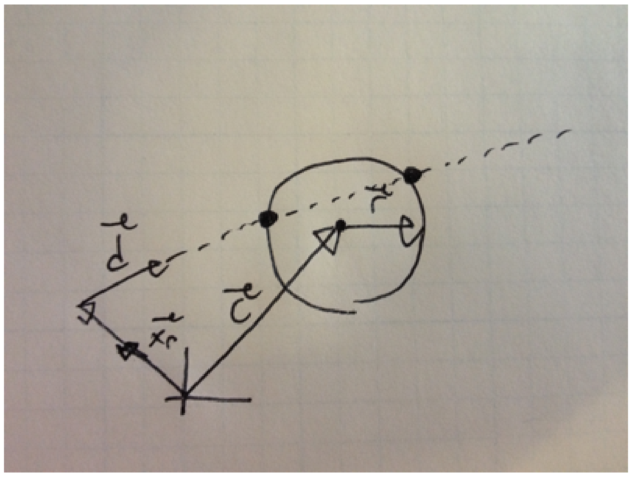

Figure 3.2: A demonstration of a ray intersecting with a sphere.

Equation (3.1) can be expanded algebraically to yield an equation that is quadratic in t. This resulting equation can then be solved using the popular quadratic formula:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = t$$

This constant-time solution for the intersection between a parametric equation and geometric primitive will be crucial for later parts of this paper.

3.2 Previous Work

As shown above, previous work (Yusuke's work) has modeled water particles as rays in order to obtain constant-time intersection solutions. Modeling a water particle path as a straight-line segment introduces several assumptions.

Because ray-modeling of water particles assumes the path is always a straight line, this means two things about the workpiece rotation:

- The acceleration is always in the same direction as the velocity during free-fall
- The accleration never changes direction while the particle is in motion.

We will see how these two simplifications produce assumptions about the workpiece rotation.

Infinitesimally Slow Rotations

We know that with ray-modeling, the acceleration may never change direction while the particle is in motion. The additional constraint from previous work is that the workpiece must rotate at a constant rate until the workpiece is fully drained. These two constaints produce the result that the workpiece must rotate infinitesimally slow; in this case, it still produces rotation in order to drain the workpiece but keeps the acceleration approximately inline with the velocity.

This also means that when a particle leaves a concave vertex, the gravity direction is always perpendicular to the leading edge of the rotation when a particle fell.

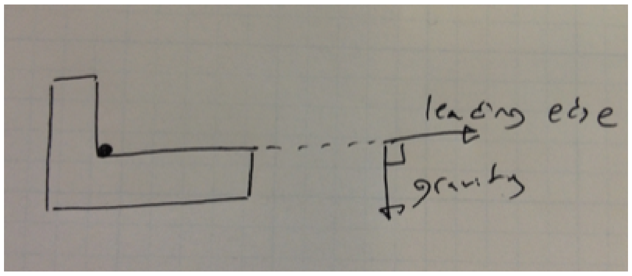

Figure 3.3: The gravity direction is perpendicular to concave vertex leading edges.

Inelastic Collisions

The other constraint that straight-line path segments have is that all particle collisions must be inelastic; velocities are instantaneously projected onto the plane or edge that they collided with.

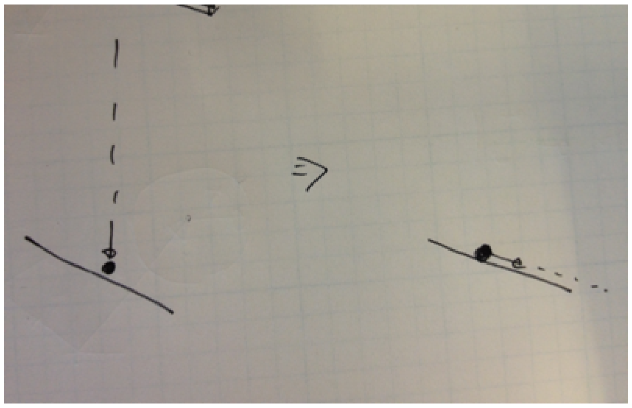

Figure 3.4: Demonstration of inelastic collisions, where velocities are projected onto the collision surface.

Kinetic Energy Limitation

TODO

3.3 Adaption to Finite Velocities

Although modeling water particles as rays is convenient, this paper aims to produce realistic workpiece draining solutions. Consequently, our modeling choice of water particles will include the vectorized acceleration as given in (3.3)

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t + \frac{1}{2}\vec{a} \cdot t^2$$

Although we now include an acceleration term, we can maintain the performance of intersection tests.

3.4 Parabola Primitive Intersections

TODO

3.5 Parametric Equations

Free Fall Equation

Equation (3.3) gives us the basic "free-fall" equation in which the particle travels through empty space with a constant acceleration field. The resulting motion is a parabola.

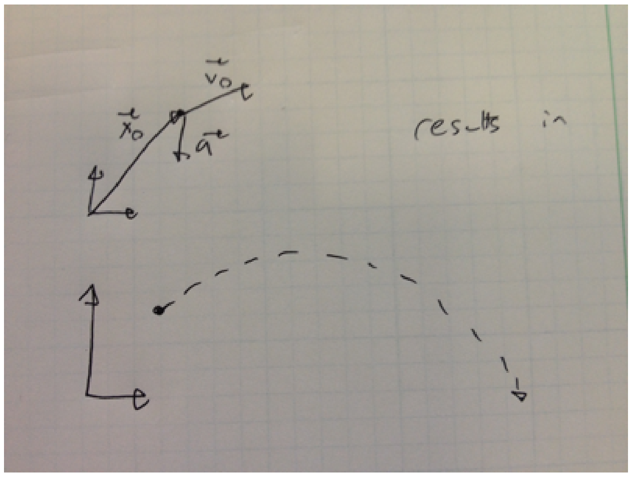

Figure 3.5: Example path traced out by the "free-fall" equation

During the course of simulation, however, the particle's entire path does not consist of only free-fall segments.

Sliding Equation

When the water particle comes to rest against an edge in the workpiece, its acceleration is projected along the edge. If this projected acceleration is zero, the particle will stay in place; if not, the particle will begin to "slide" along this edge.

In this case, we must model the motion of the particle through time. We achieve this modeling by projecting the acceleration in (3.3) along the edge, which produces a straight-line path.

Note that while the particle is now traveling along an edge, it is essentially the same as the freefall equation with a new projected acceleration.

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t + \frac{1}{2}\vec{a}_{projected} \cdot t^2$$

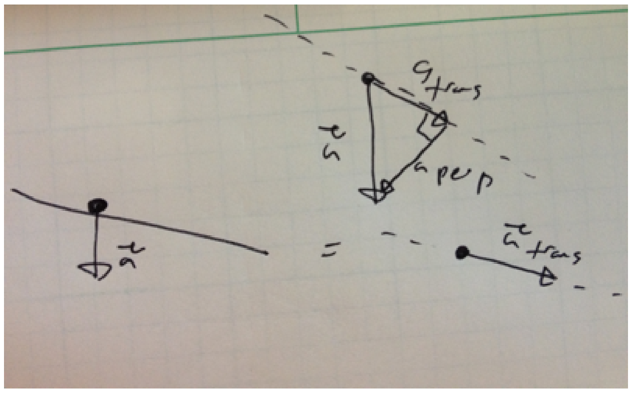

Figure 3.6: Demonstration of the sliding equation.

With these two formulations, the water particle's path can be simulated on both edges and in open space when the acceleration field is constant.

Rotation

We would like to simulate the particle during workpiece rotation. Since we no longer assume infinitely slow rotations, our particles will need to be simulated during workpiece rotation.

In this paper, we choose our frame of reference to be the X Y axes that define the workpiece geometry. This means that when the workpiece rotates, our frame of reference stays fixed to the workpiece. Consequently, the orientation of the acceleration field changes in time during rotation. This field rotation affects the motion of particles during simulation.

We see now how this rotating acceleration field affects the two above equations.

Concurrent Rotation & Sliding Equation

When the particle is sliding on an edge, the acceleration vector is projected along the edge. This means the direction of the acceleration is fixed; only the magnitude of the projected acceleration varies. Consequently, rotation of the acceleration field (when the particle is sliding) only results in the magnitude of the projected acceleration changing.

This leads to a special case of particle motion, referred to as "Concurrent Rotation and Sliding." Equation (3.5) shows the resulting formulation.

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t + \frac{1}{2}\hat{a}_{projected} \cdot t^2 \cdot |a_{projected}|(t)$$

In equation (3.5), $\hat{a}_p rojected$ is a constant unit-vector in the direction of edge. The $|a_{projected}|(t)$ term scales as the acceleration field changes direction. If the rotation occurs at a constant rate, the magnitude term can be modeled as

$$mag_{accel} = cos(\omega \cdot t)$$

where *omega* is the angular velocity.

Concurrent Rotation & Sliding Equation Intersection

In equation (3.5), we no longer have a formulation of particle motion that is strictly quadratic in t. With the additional *cos* term, this makes it quite difficult to solve for particle-primitive intersections. We can, however, easily subtitute in values of t to determine the position of the particle at a given time.

The ability to determine where the particle is at a given time will come in handy later.

Concurrent Rotation & Free-Fall Equation

With equation (3.5), we can model the particle motion during concurrent rotating and sliding. We would also like to model the motion of the particle during free-fall when the workpiece is rotating.

In this case, the acceleration vector is no longer constant as shown in (3.5).

$$\vec{x}(t) = \vec{x}_0 + \vec{v}_0 \cdot t + \frac{1}{2}\vec{a}(t) \cdot t^2$$

Here, the $\vec{a}(t)$ term describes the acceleration vector over time as the workpiece rotates. Assuming a constant rotation, this term breaks down to

$$a_x = cos(\alpha t) \cdot |a_g|$$

$$a_y = sin(\alpha t) \cdot |a_g|$$

Assumption #1 - No concurrent Rotation + Freefall

These non-linear terms produce a much more complicated equation; again, we can no longer substitute in easily to determine the intersection points.

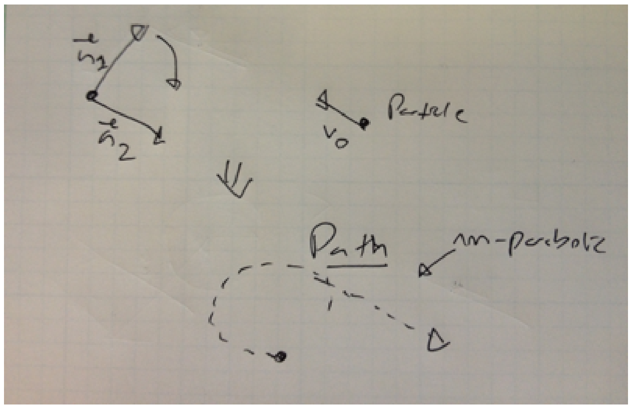

Figure 3.7: Demonstration of the particle path during concurrent rotating and free-fall

Elastic Collisions

Collisions are now elastic, meaning particles maintain a perpendicular velocity when colliding with an edge.

Planar Collision

When colliding simply on an edge, they bounce and transition back into freefall.

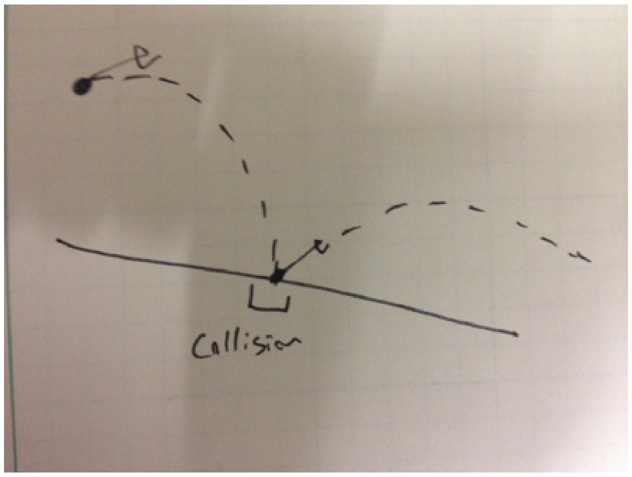

Figure 3.8: 3in

Planar To Sliding Transition

When the particles have some ϵ perpendicular velocity, they transition from freefall to sliding along an edge.

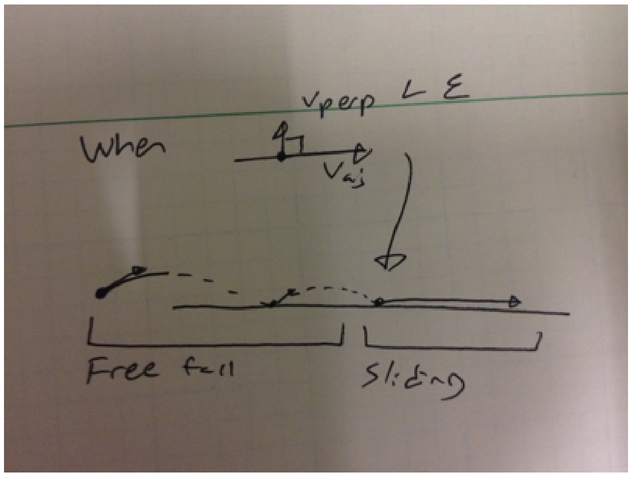

Figure 3.9: 3in

Sliding-Edge Collision

When sliding along an edge, the particle may encounter another edge. If this edge has a dot product greater than zero, it collides with the edge and enters freefall again.

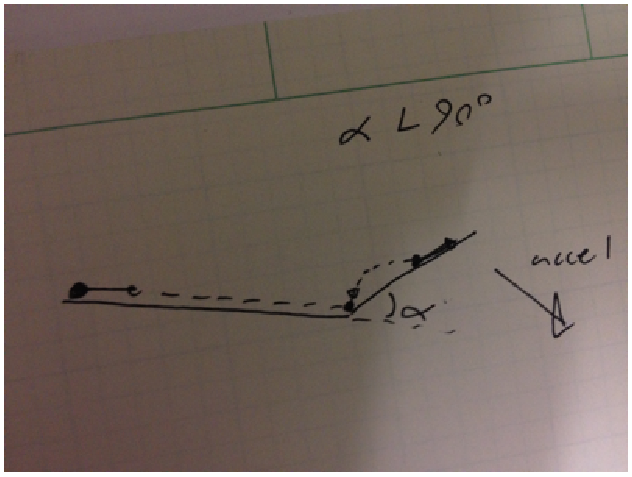

Figure 3.10: 2.5in

Sliding-Corner Collision

If the next edge has a dot product less than or equal to 0, the particle is effectively “trapped” as long as the gravity vector points within the edge.

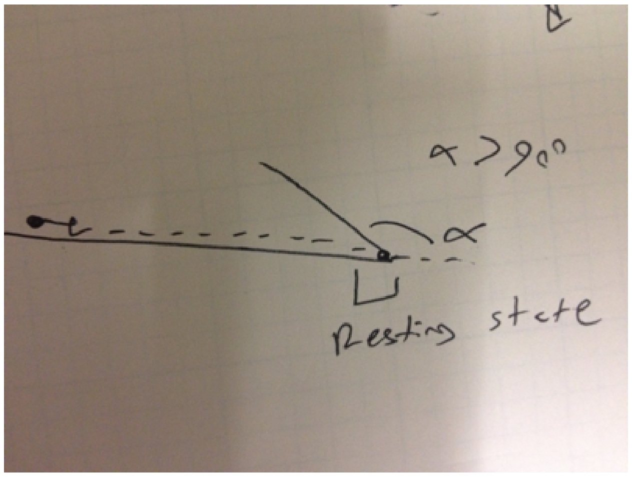

Figure 3.11: 2.5in

Conservation of Momentum

The only source of energy in this demo is the potential energy from gravity. Note that as the workpiece rotates, the gravity vector (and corresponding field) changes. This means that a particle in a low-energy configuration can transition to a high-energy configuration.

Settling Guarantee

The only addition of energy is from rotation, so in the absence of rotation, no energy is added to the system. Because the elasiticity κ of the system is less than 1, energy dissipates as the particles are simulated throughout the workpiece.

Duration of Simulation

Because of this energy dissipation, the simulation is guaranteed to terminate in one of two ways.

Simulation End – Concave Vertex

Either the particle settles into a concave vertex with a kinetic energy less than ϵ, or

Simulation End – Workpiece Exit

The particle exits the workpiece, which is the goal of the simulation.

3.6 Results

Run Time

It was fast, here are some numbers

Accuracy Comparison

It was accurate in finding intersections that some euler integration schemes would not.

With Euler Integration

3.7 Future Work & Discussion

Bounding Box Method Adaption

You could modify normal bounding box methods to use parametric equations instead.

Bounded Simulation Between Limits

There is a possibility that you could simulate between two arbitrary limits and “sweep” across the range of kinetic possiblities. Would eliminate the sampling we will see next chapter.

Chapter 4

Solution Search

4.1 General A.I. Search

Artificial Intelligence Search is a general class of algorithms that discover (or "search" for) plans that turn a start state into a goal state. The plan produced is a sequence of actions that, if taken, will obtain a goal state.

State Space Formulation

The "state" is defined as a set of information that minimally represents the current progress to the goal. The state is transformed by actions available to the agent. The successor function is responsible for transforming a state and action into a successor state.

Other parts of the problem that are not time-variant are considered part of the environment.

State Space Size

The size of the state space is all possible valid combinations of the components of the state space. For a positional search problem on an M by N grid with one agent, the size of the state space is MN. For two players, it would be $(MN)^2$.

As the size of the state space expands, it can have disastrous consequences on the effectiveness of a search problem. General A.I. Search is effective at finding solutions in state spaces that are manageable; if the state space becomes too expansive, the search algorithms have to enumerate through far too many possibilities to make progress towards a goal. This prevents the search from terminating in a reasonable amount of time.

State Space Exploration

Exploration of a state space happens through a transition function that generates successor states from state action pairs. The transition function commonly contains all the "environment" information and validation checks – for example in a positional search problem, the transition function would contain the list of walls and prevent the agent from moving into a wall.

4.2 Adaption of A.I. Search

The A.I. Search algorithms in this paper will be unmodified versions of those found in popular literature, but our approach to the state space approach will be quite different.

Traditional Formulation

The traditional state formulation for kinematic problems is to define the state of a particle as it's currrent position, velocity, and acceleration. The actions available to the agent at any time are usually actions that change the acceleration on the particle. The successor function integrates this kinematic state in time with a pre-defined accuracy.

The problem with this type of state formulation is that the state space becomes far too expansive for the majority of problems. Even in the reduced problem where there are 100 possible position, velocity, and acceleration vectors, the state space is 100^3. This presents far too many possible states for search algorithms to enumerate, and it produces an unbearable computational load.

Because of this, general A.I. search is rarely used in kinematics problems – other techniques like kinematical planning are more common.

Our State Formulation

Our state formulation overcomes these difficulties by drastically reducing the size of the state space. We instead transition most of the computational overhead to the transition function.

We define our state as the "settled" position of the particle, meaning the state is either:

1. The concave vertex the particle is resting in, or
2. outside of the workpiece.

By defining our state this way, we dramatically reduce the size of the state space to

$$num(V_{cc}) + 1$$

where V_{cc} is defined as the set of concave vertices in the workpiece.

Because the maximum number of states visited is now only linear in concave vertices, our search algorithms have far fewer states to expand and a reduced computational load.

Exploration

The bulk of the computational work will now be in the transition function. The transition function will be responsible for producing the set of concave vertices "reachable" from the current concave vertex from the available actions.

4.3 Transition Function

The transition function produces successor states from a set of available actions and initial state. For our problem, this means that the transition function will produce the set of concave vertices (or workpiece exit status) "reachable" from a given arbitrary concave vertex.

The actions available to the agent will represent the possible gravity directions from a given concave vertex. These gravity directions can be obtained by rotating the workpiece; in this sense, a given gravity direction must be obtained by rotating a starting gravity direction to the desired direction.

Sampling

The transition function will sample from the available set of gravity vectors and produce a desired "turn." A turn of the workpiece results in the gravity vector sweeping from g_{start} to g_{end} over a time t_{turn}. During this turn, the particle's path is simulated. After the turn, the particle is simulated over time until it settles in a new concave vertex or exits the workpiece.

In this sense, one can think of the transition function as taking in a concave vertex and producing a set of "reachable" concave vertices:

$$trans(V_{cc}^{i}) = \left\{V_{cc}^{1}, V_{cc}^{2}, ...\right\}$$

Limits of Sampling

In this paper, we restrict the possible gravity directions available from a concave vertex. The primary reason for this is based on our simulation – if a shift in gravity causes a particle to leave the surface while the workpiece is still rotating, we cannot simulate that part of the path efficiently. For this reason, we throw out the possible gravity vectors that cause this type of particle movement to happen.

The first set gravity vectors we exclude are any vectors that don't cause the particle to leave the concave vertex. These vectors result in no state change and can be ignored.

The second set of gravity vectors that are excluded are those that cause the particle to fall off the leading edge out of a concave vertex.

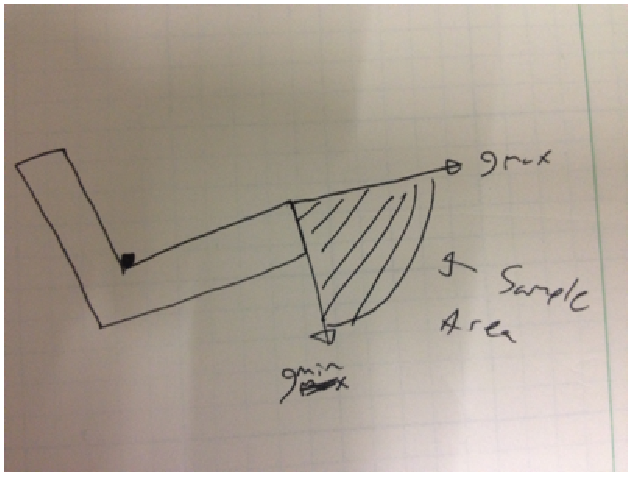

Figure 4.1: The possible range of gravity vectors to sample from.

This leaves two arcs of gravity vectors. The bounds on each arc are defined by the minimal gravity vector to produce movement out of the concave vertex (g_ϵ) and the maximum gravity vector to keep the particle on the surface (g_{max}).

Ideal Transition Function

The ideal transition function produces the maximum set of "reachable" concave vertices while minimizing the necessary computation. In order to do this, the transition function must construct a number of starting kinetic states for the particle and simulate where the particle settles.

Representative Coverage Between Limits

Because each arc of gravity vectors is a range, there are an infinite number of samples to extract. This means that our transition function must enumerate an infinite number of possibilities to get full confidence that the set of "reachable" concave vertices has been achieved.

To reduce our computational load, we will instead choose a set of gravity vectors that produces a reasonable coverage over the possible types of kinetic paths achieved. Because our goal is to produce the maximum number of reachable concave vertices, we will choose a sampling method that produces the most diverse set of kinetic paths.

If the possible range of samples is from 0 to 1, a fairly reasonable sample function was to use x^3. Because the acceleration effects compound with time, uniform sampling produced far too many high-velocity kinetic paths. A cubed sampling produced a more even coverage of kinetic paths.

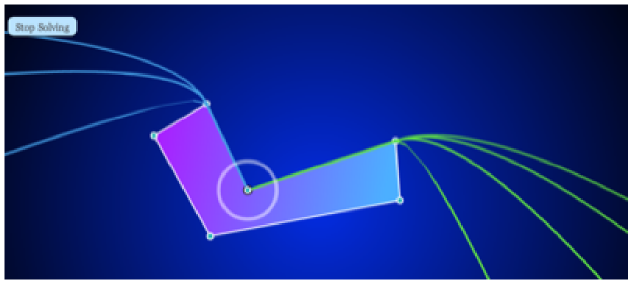

Figure 4.2: The range of kinetic paths produced from 10 samples at uniform sampling.

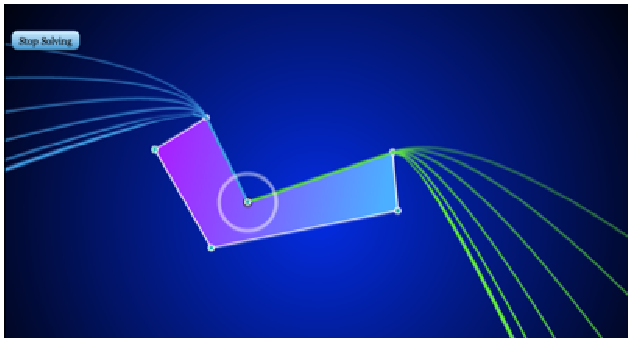

Figure 4.3: The range of kinetic paths produced from 10 samples at cubed sampling. Note the resulting kinetic paths evenly cover the space after the leading edge.

Graph Search

Reduce possibilities, prioritize by our "cost" for the algorithm.

Cost Sensitive Closed List

Essentially similar to CSCL because it is order-independent.

4.4 Search

Now we can search from a start state outwards

Uniform Cost Search

Using uniform cost, we always maintain optimality based on our cost.

Cost Function

Our cost function can be a variety of things based on the backwards path.

Time

Most popular is time, because time is money.

Energy - Rotation Angle

Second most popular is energy based on the rotation angle. This would be appropriate if the workpiece was well balanced but fixture generated a lot of friction while rotating.

Energy - Workpiece Center of Gravity

Another option is based on center of gravity if the energy from the workpiece rotations outweighed the friction in the motors.

Solution

Defined as the sequence of gravity transitions from each concave vertex to produce a particle path that exits the workpiece.

4.5 Control Sequence Generator

Now that we have a solution, we need to generate a control sequence for the workpiece rotator.

Sample-defined Rotations

These rotations are defined for us by the transition function.

Intermediary Rotations

Between these samples we don't have anything. What we do is just interpolate between the last sample and the beginning of the next sample with a cubic bezier curve.

4.6 Results

Run Time

Runs fast.

Part Complexity

Transition function is constant in best case, linear in worst case based on the number of polygon edges.

Demonstration of Solution

Go to this web address!

4.7 Future Work & Discussion

Could be improvements

Heuristics for A* Search

A* search can be added with heuristics. Distance to workpiece bounding box might be a good one, but that's essentially a radially symmetric heuristic, not very informative.

Chapter 5

Conclusion

This is the conclusion, it concludes the paper.

Bibliography

[1] James Moorer. "Signal Processing Aspects of Computer Music–A Survey". In: *Computer Music Journal* 1.1 (1977).