Spectral Rigid Body Dynamics

Mikola Lysenko

May 5, 2010

Overview

Rigid Body Dynamics

Lagrangian Mechanics

Standard Collisions

Constraint Based Collisions

Fourier Methods

Numerical Issues

Rigid Body Dynamics

An approximate model of low energy physics for stiff objects

Rigid Body Dynamics

An approximate model of low energy physics for stiff objects

Pros:

- + Pretty accurate at human energy scales
- + Good for stiff materials (ie metals, plastics etc.)
- + Easy kinematic constraints (useful for mechanisms)
- + Standard animation tool (videogames!)

Rigid Body Dynamics

An approximate model of low energy physics for stiff objects

Pros:

- $+$ Pretty accurate at human energy scales
- $+$ Good for stiff materials (ie metals, plastics etc.)
- $+$ Easy kinematic constraints (useful for mechanisms)
- $+$ Standard animation tool (videogames!)

Cons:

- - Inaccurate at extremely large energies
- - Bad for materials with low elastic modulus
- - Not always solvable! (See: Painleve's paradox)

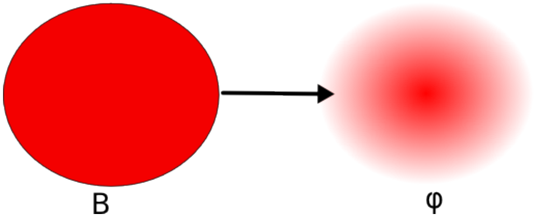
What is a Rigid Body?

An idealized solid object with elastic modulus $= \infty$

What is a Rigid Body?

An idealized solid object with elastic modulus $= \infty$

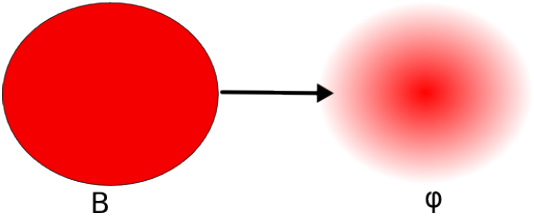
We identify a body B with a scalar field, $\varphi : \mathbb{R}^d \to \mathbb{R}^+$



What is a Rigid Body?

An idealized solid object with elastic modulus $= \infty$

We identify a body B with a scalar field, $\varphi : \mathbb{R}^d \to \mathbb{R}^+$

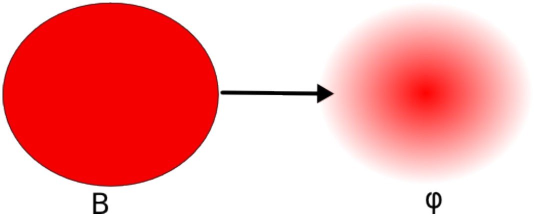


φ represents the mass distribution of B

What is a Rigid Body?

An idealized solid object with elastic modulus $= \infty$

We identify a body B with a scalar field, $\varphi : \mathbb{R}^d \to \mathbb{R}^+$



φ represents the mass distribution of B

$\varphi(x) = 0$ indicates B does not occupy the space at x

Configuration Space of a Rigid Body

Transformations of rigid mass fields must preserve distance and handedness

Configuration Space of a Rigid Body

Transformations of rigid mass fields must preserve distance and handedness

In other words, must be a direct Euclidean isometry

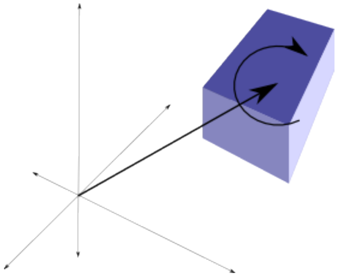
Isomorphic to finite dimensional Lie group, $SE(d) \cong SO(d) \ltimes \mathbb{R}^d$

Configuration Space of a Rigid Body

Transformations of rigid mass fields must preserve distance and handedness

In other words, must be a direct Euclidean isometry

Isomorphic to finite dimensional Lie group, $SE(d) \cong SO(d) \ltimes \mathbb{R}^d$



Can be parameterized by a translation t and a rotation R

Matrix: $\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$

$\binom{d+1}{2}$ degrees of freedom

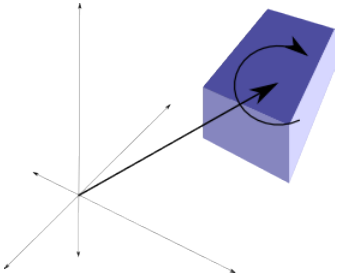
Tangent space: $\mathfrak{so}(d+1)$

Configuration Space of a Rigid Body

Transformations of rigid mass fields must preserve distance and handedness

In other words, must be a direct Euclidean isometry

Isomorphic to finite dimensional Lie group, $SE(d) \cong SO(d) \ltimes \mathbb{R}^d$



Can be parameterized by a translation t and a rotation R

Matrix: $\begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix}$

$\binom{d+1}{2}$ degrees of freedom

Tangent space: $\mathfrak{so}(d+1)$

Motions of rigid objects $\cong$ **curves** $q(t) \subset SE(d)$

Lagrangian Mechanics

Turns physics into an optimization problem.

Lagrangian Mechanics

Turns physics into an optimization problem.

Given a configuration curve q at time t, define the *Lagrangian*

$$\mathcal{L}(q, \dot{q}, t) = T(\dot{q}) - U(q, t)$$

Where $T(\dot{q}) = \frac{1}{2}\dot{q}^T M \dot{q}$ is the kinetic energy and U is a potential

Lagrangian Mechanics

Turns physics into an optimization problem.

Given a configuration curve q at time t, define the *Lagrangian*

$$\mathcal{L}(q, \dot{q}, t) = T(\dot{q}) - U(q, t)$$

Where $T(\dot{q}) = \frac{1}{2}\dot{q}^T M \dot{q}$ is the kinetic energy and U is a potential

Hamilton's Principle of Least Action:

Physically plausible motions do minimal work

Lagrangian Mechanics

Turns physics into an optimization problem.

Given a configuration curve q at time t, define the *Lagrangian*

$$\mathcal{L}(q, \dot{q}, t) = T(\dot{q}) - U(q, t)$$

Where $T(\dot{q}) = \frac{1}{2}\dot{q}^T M\dot{q}$ is the kinetic energy and U is a potential

Hamilton's Principle of Least Action:

Physically plausible motions do minimal work

In other words:

$$\underset{q:[t_0,t_1)\rightarrow SE(d)}{\operatorname{argmin}} \int_{t_0}^{t_1} L(q, \dot{q}, t)dt$$

Forward Kinematics

Solve:

$$\operatorname*{argmin}_{q:[t_0,t_1)\to SE(d)} \int_{t_0}^{t_1} L(q, \dot{q}, t)dt$$

Forward Kinematics

Solve:

$$\underset{q:[t_0,t_1)\to SE(d)}{\operatorname{argmin}} \int_{t_0}^{t_1} L(q,\dot{q},t)dt$$

This is a 1st order variational problem, so use Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L(q,\dot{q},t)}{\partial \dot{q}}\right) = \frac{\partial L(q,\dot{q},t)}{\partial q}$$

Forward Kinematics

Solve:

$$\underset{q:[t_0,t_1)\rightarrow SE(d)}{\operatorname{argmin}} \int_{t_0}^{t_1} L(q, \dot{q}, t)dt$$

This is a 1st order variational problem, so use Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L(q, \dot{q}, t)}{\partial \dot{q}}\right) = \frac{\partial L(q, \dot{q}, t)}{\partial q}$$

Unpack definitions and simplify:

$$\frac{d}{dt}\left(\frac{\partial \frac{1}{2}\dot{q}^T M \dot{q}}{\partial \dot{q}}\right) \quad = \quad \frac{\partial U(q, t)}{\partial q}$$

Forward Kinematics

Solve:

$$\underset{q:[t_0,t_1)\to SE(d)}{\operatorname{argmin}} \int_{t_0}^{t_1} L(q,\dot{q},t)dt$$

This is a 1st order variational problem, so use Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L(q,\dot{q},t)}{\partial \dot{q}}\right) = \frac{\partial L(q,\dot{q},t)}{\partial q}$$

Unpack definitions and simplify:

$$\begin{aligned} \frac{d}{dt}\left(\frac{\partial \frac{1}{2}\dot{q}^T M \dot{q}}{\partial \dot{q}}\right) &= \frac{\partial U(q,t)}{\partial q} \\ M\ddot{q} &= \nabla U \end{aligned}$$

Newton's equations!

Multiple Bodies

Q: How to deal with multiple independent bodies?

Multiple Bodies

Q: How to deal with multiple independent bodies?

A: Tensor sum

Let B_i, B_j be independent rigid bodies with motions q_i, q_j

Configuration space	$SE(d)^2 \cong SE(d) \oplus SE(d)$
Motion	$q(t) \cong q_i(t) \oplus q_j(t)$
Lagrangian	$L(q, \dot{q}, t) = L(q_i, \dot{q}_i, t) + L(q_j, \dot{q}_j, t)$

Multiple Bodies

Q: How to deal with multiple independent bodies?

A: Tensor sum

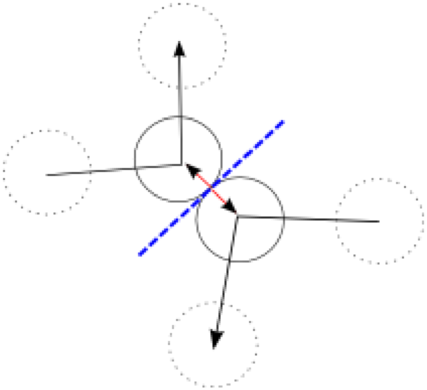
Let B_i, B_j be independent rigid bodies with motions q_i, q_j

Configuration space	$SE(d)^2 \cong SE(d) \oplus SE(d)$
Motion	$q(t) \cong q_i(t) \oplus q_j(t)$
Lagrangian	$L(q, \dot{q}, t) = L(q_i, \dot{q}_i, t) + L(q_j, \dot{q}_j, t)$

Scales to n bodies, get Lagrangian in $SE(d)^n$

Collisions

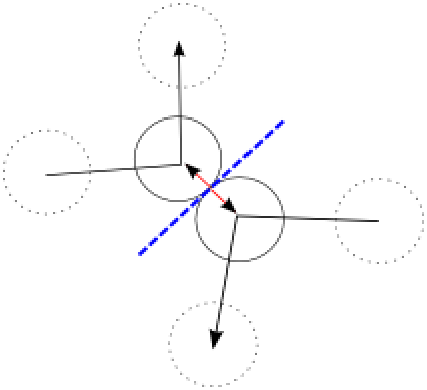
Solids can't physically occupy the same space.



Need to keep them separated

Collisions

Solids can't physically occupy the same space.



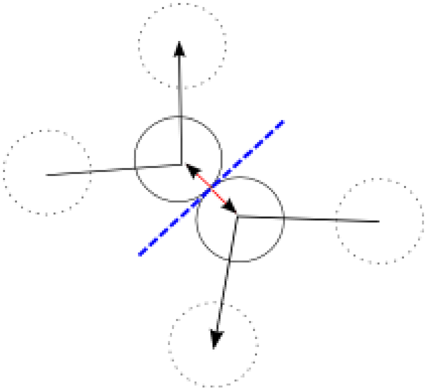
Need to keep them separated

Standard method:

- Time step to point of impact

Collisions

Solids can't physically occupy the same space.



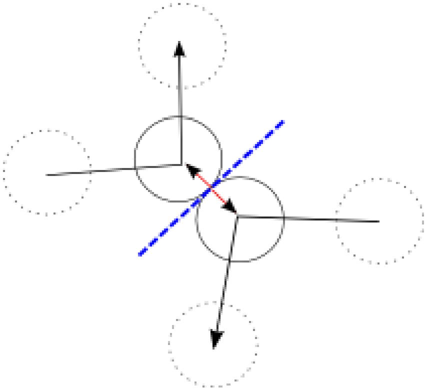
Need to keep them separated

Standard method:

- Time step to point of impact
- Determine contact manifold

Collisions

Solids can't physically occupy the same space.



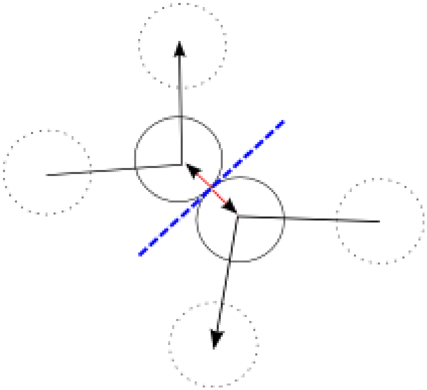
Need to keep them separated

Standard method:

- Time step to point of impact
- Determine contact manifold
- Apply penalty force/impulse to separate objects

Collisions

Solids can't physically occupy the same space.



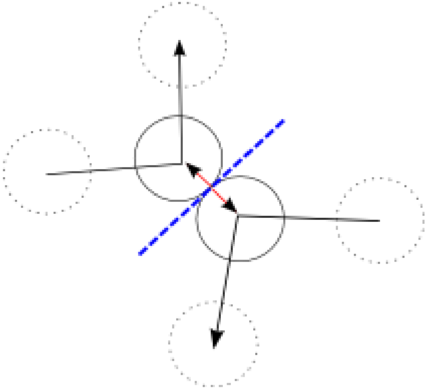
Need to keep them separated

Standard method:

- Time step to point of impact
- Determine contact manifold
- Apply penalty force/impulse to separate objects
- Iterate until all collisions are resolved

Collisions

Solids can't physically occupy the same space.



Need to keep them separated

Standard method:

- Time step to point of impact
- Determine contact manifold
- Apply penalty force/impulse to separate objects
- Iterate until all collisions are resolved

+ Just like high school physics

Problems with typical collision methods

Handling collisions this way is hard

Problems with typical collision methods

Handling collisions this way is hard

- - Finding exact point/time of impact is expensive
- - Contact manifolds can be difficult to classify
- - Normal forces are ambiguous for curvy shapes
- - Impulses are discontinuous; results in stiff system
- - Penalty methods don't gaurantee separation
- - Convergence properties not well understood
- - Unclear how to handle sharp corners
- - Discontinuity in simulation
- - Complicated, ad-hoc

Problems with typical collision methods

Handling collisions this way is hard

- - Finding exact point/time of impact is expensive
- - Contact manifolds can be difficult to classify
- - Normal forces are ambiguous for curvy shapes
- - Impulses are discontinuous; results in stiff system
- - Penalty methods don't gaurantee separation
- - Convergence properties not well understood
- - Unclear how to handle sharp corners
- - Discontinuity in simulation
- - Complicated, ad-hoc

But can be made to work with enough hacking...

There has got to be an easier way...

There has got to be an easier way...

Minimal requirement for physical plausibility

At all times no two solids intersect

There has got to be an easier way...

Minimal requirement for physical plausibility

At all times no two solids intersect

Is this really all there is to it?

Constraint Based Impacts

For any body B_i with mass field φ_i, define the set:

$$A_i = \kappa\iota \ \mathsf{supp}\ \varphi$$

Constraint Based Impacts

For any body B_i with mass field φ_i, define the set:

$$A_i = \kappa\iota \text{ supp } \varphi$$

So two solids, A_i, A_j, *collide* at a configuration q_i, q_j iff:

$$\text{collide}(q_i A_i, q_j A_j) \Leftrightarrow \iota(q_i A_i \cap q_j A_j) \neq \emptyset$$

Constraint Based Impacts

For any body B_i with mass field φ_i, define the set:

$$A_i = \kappa\iota \text{ supp } \varphi$$

So two solids, A_i, A_j, *collide* at a configuration q_i, q_j iff:

$$\text{collide}(q_i A_i, q_j A_j) \Leftrightarrow \iota(q_i A_i \cap q_j A_j) \neq \emptyset$$

But,

$$\iota(q_i A_i \cap q_j A_j) \neq \emptyset \Leftrightarrow \text{vol } q_i A_i \cap q_j A_j > 0$$

Constraint Based Impacts

For any body B_i with mass field φ_i, define the set:

$$A_i = \kappa\iota \text{ supp } \varphi$$

So two solids, A_i, A_j, *collide* at a configuration q_i, q_j iff:

$$\text{collide}(q_i A_i, q_j A_j) \Leftrightarrow \iota(q_i A_i \cap q_j A_j) \neq \emptyset$$

But,

$$\iota(q_i A_i \cap q_j A_j) \neq \emptyset \Leftrightarrow \text{vol } q_i A_i \cap q_j A_j > 0$$

Define

$$C_{i,j}(q_i, q_j) \overset{def}{\equiv} \text{vol } q_i A_i \cap q_j A_j$$

And so we replace the impact forces with a system of differentiable holonomic inequality constraints:

$$C_{i,j} \leq 0$$

Equations of motion revisited

New problem:

$$\text{minimize} \int_{t_0}^{t_1} L(q, \dot{q}, t)\,dt$$

$$\text{subject to } C_{i,j}(q_i, q_j) \leq 0 \quad \forall t \in [t_0, t_1), i \neq j$$

Equations of motion revisited

New problem:

$$\text{minimize} \int_{t_0}^{t_1} L(q, \dot{q}, t) dt$$

$$\text{subject to } C_{i,j}(q_i, q_j) \leq 0 \quad \forall t \in [t_0, t_1), i \neq j$$

Apply KKT conditions + Euler-Lagrange to get complementarity problem:

$$\frac{d}{dt}\left(\frac{\partial T(\dot{q}_i)}{\partial \dot{q}_i}\right) - \frac{\partial U(q, t)}{\partial q_i} + \sum_{j \neq i} \mu_{i,j} \frac{\partial C_{i,j}(q_i, q_j)}{\partial q_i} = 0$$

$$0 \leq \mu_{i,j} \perp -C_{i,j} \geq 0$$

Equations of motion revisited

New problem:

$$\text{minimize} \int_{t_0}^{t_1} L(q, \dot{q}, t)dt$$

$$\text{subject to } C_{i,j}(q_i, q_j) \leq 0 \quad \forall t \in [t_0, t_1), i \neq j$$

Apply KKT conditions + Euler-Lagrange to get complementarity problem:

$$\frac{d}{dt}\left(\frac{\partial T(\dot{q}_i)}{\partial \dot{q}_i}\right) - \frac{\partial U(q,t)}{\partial q_i} + \sum_{j \neq i} \mu_{i,j} \frac{\partial C_{i,j}(q_i, q_j)}{\partial q_i} = 0$$

$$0 \leq \mu_{i,j} \perp -C_{i,j} \geq 0$$

Exactly elastic collision response!

Slack variables are impulse forces

Calculating $C_{i,j}$

Need to compute:

$$C_{i,j}(q_i, q_j) = \text{vol } q_i A_i \cap q_j A_j$$

Calculating $C_{i,j}$

Need to compute:

$$C_{i,j}(q_i, q_j) = \text{vol } q_i A_i \cap q_j A_j$$

Construct the *indicator* on A

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

Calculating $C_{i,j}$

Need to compute:

$$C_{i,j}(q_i, q_j) = \text{vol } q_i A_i \cap q_j A_j$$

Construct the *indicator* on A

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

Observe:

$$\mathbf{1}_{A_i \cap A_j}(x) = \mathbf{1}_{A_i}(x)\mathbf{1}_{A_j}(x)$$

$$\text{vol } A_i = \int_{\mathbb{R}^d} \mathbf{1}_{A_i}(x)dx$$

Calculating $C_{i,j}$

Need to compute:

$$C_{i,j}(q_i, q_j) = \text{vol } q_i A_i \cap q_j A_j$$

Construct the *indicator* on A

$$\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}$$

Observe:

$$\mathbf{1}_{A_i \cap A_j}(x) = \mathbf{1}_{A_i}(x)\mathbf{1}_{A_j}(x) \qquad \text{vol } A_i = \int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(x) dx$$

So:

$$C_{i,j}(q_i, q_j) = \int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(q_i^{-1} x)\mathbf{1}_{A_j}(q_j^{-1} x) dx$$

Calculating $C_{i,j}$ (cont.)

$$C_{i,j}(q_i, q_j) = \int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(q_i^{-1}x)\mathbf{1}_{A_j}(q_j^{-1}x)dx$$

Calculating $C_{i,j}$ (cont.)

$$C_{i,j}(q_i, q_j) = \int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(q_i^{-1}x)\mathbf{1}_{A_j}(q_j^{-1}x)dx$$

Push terms to one side:

$$\int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(x)\mathbf{1}_{A_j}(q_j^{-1}q_i x)dx$$

Calculating $C_{i,j}$ (cont.)

$$C_{i,j}(q_i, q_j) = \int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(q_i^{-1}x)\mathbf{1}_{A_j}(q_j^{-1}x)dx$$

Push terms to one side:

$$\int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(x)\mathbf{1}_{A_j}(q_j^{-1}q_i x)dx$$

Substitute $q_j^{-1}q_i x \mapsto Rx - y$ and let $\widetilde{\mathbf{1}_{A_j}}(x) = \mathbf{1}_{A_j}(-x)$:

$$\int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(x)\widetilde{\mathbf{1}_{A_j}}(y - Rx)dx = \int\limits_{\mathbb{R}^d} \mathbf{1}_{A_i}(R^{-1}x)\widetilde{\mathbf{1}_{A_j}}(y - x)dx$$

Calculating $C_{i,j}$ (cont.)

$$C_{i,j}(q_i, q_j) = \int_{\mathbb{R}^d} \mathbf{1}_{A_i}(q_i^{-1}x)\mathbf{1}_{A_j}(q_j^{-1}x)dx$$

Push terms to one side:

$$\int_{\mathbb{R}^d} \mathbf{1}_{A_i}(x)\mathbf{1}_{A_j}(q_j^{-1}q_i x)dx$$

Substitute $q_j^{-1}q_i x \mapsto Rx - y$ and let $\widetilde{\mathbf{1}_{A_j}}(x) = \mathbf{1}_{A_j}(-x)$:

$$\int_{\mathbb{R}^d} \mathbf{1}_{A_i}(x)\widetilde{\mathbf{1}_{A_j}}(y - Rx)dx = \int_{\mathbb{R}^d} \mathbf{1}_{A_i}(R^{-1}x)\widetilde{\mathbf{1}_{A_j}}(y - x)dx$$

So,

$$C_{i,j}(q_i, q_j) = ((\mathbf{1}_{A_i} \circ R^{-1}) \star \widetilde{\mathbf{1}_{A_j}})(y)$$

Fourier Methods

Convolution?

$$C_{i,j}(q_i, q_j) = ((\mathbf{1}_{A_i} \circ R^{-1}) \star \widetilde{\mathbf{1}_{A_j}})(y)$$

Fourier Methods

Convolution? Take a Fourier transform!

$$C_{i,j}(q_i, q_j) = ((\mathbf{1}_{A_i} \circ R^{-1}) \star \widetilde{\mathbf{1}_{A_j}})(y) = \mathcal{F}^{-1}\left((\widehat{\mathbf{1}_{A_i}} \circ R^{-1}) \overline{\widehat{\mathbf{1}_{A_j}}} \right)(y)$$

Fourier Methods

Convolution? Take a Fourier transform!

$$C_{i,j}(q_i, q_j) = ((\mathbf{1}_{A_i} \circ R^{-1}) \star \widetilde{\mathbf{1}_{A_j}})(y) = \mathcal{F}^{-1}\left((\widehat{\mathbf{1}_{A_i}} \circ R^{-1})\overline{\widehat{\mathbf{1}_{A_j}}}\right)(y)$$

What about gradients? (ie $\frac{\partial C_{i,j}(q_i,q_j)}{\partial q_i}$)

Fourier Methods

Convolution? Take a Fourier transform!

$$C_{i,j}(q_i, q_j) = ((\mathbf{1}_{A_i} \circ R^{-1}) \star \widetilde{\mathbf{1}_{A_j}})(y) = \mathcal{F}^{-1}\left((\widehat{\mathbf{1}_{A_i}} \circ R^{-1})\overline{\widehat{\mathbf{1}_{A_j}}}\right)(y)$$

What about gradients? (ie $\frac{\partial C_{i,j}(q_i,q_j)}{\partial q_i}$)

Fix parameters $q_i = (R_i, t_i)$,

$$q_i x = R_i x + t_i$$

$$q_i^{-1} = (R_i^{-1}, -R_i^{-1} t_i)$$

$$q_j q_i^{-1} x = Rx - y$$

Fourier Methods

Convolution? Take a Fourier transform!

$$C_{i,j}(q_i, q_j) = ((\mathbf{1}_{A_i} \circ R^{-1}) \star \widetilde{\mathbf{1}_{A_j}})(y) = \mathcal{F}^{-1}\left((\widehat{\mathbf{1}_{A_i}} \circ R^{-1})\overline{\widehat{\mathbf{1}_{A_j}}}\right)(y)$$

What about gradients? (ie $\frac{\partial C_{i,j}(q_i,q_j)}{\partial q_i}$)

Fix parameters $q_i = (R_i, t_i)$,

$$q_i x = R_i x + t_i$$

$$q_i^{-1} = (R_i^{-1}, -R_i^{-1} t_i)$$

$$q_j q_i^{-1} x = Rx - y$$

Solve for R, y,

$$R = R_j R_i^{-1}$$

$$y = t_j - R_j R_i^{-1} t_i$$

Fourier Methods

Convolution? Take a Fourier transform!

$$C_{i,j}(q_i, q_j) = ((\mathbf{1}_{A_i} \circ R^{-1}) \star \widetilde{\mathbf{1}_{A_j}})(y) = \mathcal{F}^{-1}\left((\widehat{\mathbf{1}_{A_i}} \circ R^{-1})\overline{\widehat{\mathbf{1}_{A_j}}}\right)(y)$$

What about gradients? (ie $\frac{\partial C_{i,j}(q_i,q_j)}{\partial q_i}$)

Fix parameters $q_i = (R_i, t_i)$,

$$q_i x = R_i x + t_i$$

$$q_i^{-1} = (R_i^{-1}, -R_i^{-1} t_i)$$

$$q_j q_i^{-1} x = Rx - y$$

Solve for R, y,

$$R = R_j R_i^{-1}$$

$$y = t_j - R_j R_i^{-1} t_i$$

Need to compute $\frac{\partial C_{i,j}(R,y)}{\partial y}$, $\frac{\partial C_{i,j}(R,y)}{\partial R}$ then use chain rule

Or by symmetry: $C_{i,j}(q_i, q_j) = C_{j,i}(q_j, q_i)$

Translational Gradient

Start with the translational case first.

Let y^k denote the k^{th} component of y,

Translational Gradient

Start with the translational case first.

Let y^k denote the k^{th} component of y,then

$$\frac{\partial C_{i,j}}{\partial y^k} = \frac{\partial}{\partial y^k}\left(\int\limits_{\mathbb{R}^d} \widehat{\mathbf{1}_{A_i}}(R^{-1}\omega)\overline{\widehat{\mathbf{1}_{A_j}}(\omega)}e^{2\pi i\langle \omega, y\rangle} d\omega \right)$$

Translational Gradient

Start with the translational case first.

Let y^k denote the k^{th} component of y,then

$$\begin{aligned}
\frac{\partial C_{i,j}}{\partial y^k} &= \frac{\partial}{\partial y^k}\left(\int\limits_{\mathbb{R}^d} \widehat{\mathbf{1}_{A_i}}(R^{-1}\omega)\overline{\widehat{\mathbf{1}_{A_j}}(\omega)}e^{2\pi i\langle\omega,y\rangle}d\omega\right) \\
\ldots &= \int\limits_{\mathbb{R}^d} 2\pi i\left\langle \omega, v^k\right\rangle \widehat{\mathbf{1}_{A_i}}(R^{-1}\omega)\overline{\widehat{\mathbf{1}_{A_j}}(\omega)}e^{2\pi i\langle\omega,y\rangle}d\omega
\end{aligned}$$

Where v^k denotes the k^{th} basis vector

Translational Gradient

Start with the translational case first.

Let y^k denote the k^{th} component of y,then

$$\begin{aligned}\frac{\partial C_{i,j}}{\partial y^k} &= \frac{\partial}{\partial y^k}\left(\int\limits_{\mathbb{R}^d} \widehat{\mathbf{1}_{A_i}}(R^{-1}\omega)\overline{\widehat{\mathbf{1}_{A_j}}(\omega)}e^{2\pi i\langle\omega,y\rangle}d\omega\right)\\ \dots &= \int\limits_{\mathbb{R}^d} 2\pi i\left\langle\omega, v^k\right\rangle \widehat{\mathbf{1}_{A_i}}(R^{-1}\omega)\overline{\widehat{\mathbf{1}_{A_j}}(\omega)}e^{2\pi i\langle\omega,y\rangle}d\omega\end{aligned}$$

Where v^k denotes the k^{th} basis vector

Conclusion: Translational gradient is just a multiplier

Rotational Gradient

Parameterize $R = \exp(\mathfrak{r})$, where $\mathfrak{r} \in \mathfrak{so}(d)$ with basis $\mathfrak{r}_{k,l}$

In otherwords $d \times d$ skew symmetric matrices, $\mathfrak{r}^T = -\mathfrak{r}$

Rotational Gradient

Parameterize $R = \exp(\mathfrak{r})$, where $\mathfrak{r} \in \mathfrak{so}(d)$ with basis $\mathfrak{r}_{k,l}$

In otherwords $d \times d$ skew symmetric matrices, $\mathfrak{r}^T = -\mathfrak{r}$

$$\frac{\partial C_{i,j}}{\partial \mathfrak{r}_{k,l}} = \frac{\partial}{\partial \mathfrak{r}_{k,l}} \left(\int_{\mathbb{R}^d} \widehat{\mathbf{1}_{A_i}}(\exp(-\mathfrak{r})\omega)\overline{\widehat{\mathbf{1}_{A_j}}(\omega)} e^{2\pi i \langle \omega, y \rangle} d\omega \right)$$

Rotational Gradient

Parameterize $R = \exp(\mathfrak{r})$, where $\mathfrak{r} \in \mathfrak{so}(d)$ with basis $\mathfrak{r}_{k,l}$

In otherwords $d \times d$ skew symmetric matrices, $\mathfrak{r}^T = -\mathfrak{r}$

$$\begin{aligned}
\frac{\partial C_{i,j}}{\partial \mathfrak{r}_{k,l}} &= \frac{\partial}{\partial \mathfrak{r}_{k,l}} \left(\int\limits_{\mathbb{R}^d} \widehat{\mathbf{1}_{A_i}}(\exp(-\mathfrak{r})\omega)\overline{\widehat{\mathbf{1}_{A_j}}(\omega)} e^{2\pi i \langle \omega, y \rangle} d\omega \right) \\
\dots &= \int\limits_{\mathbb{R}^d} - \left\langle \nabla \widehat{\mathbf{1}_{A_i}}(R^{-1}\omega), R^{-1}\mathfrak{r}_{k,l}\omega \right\rangle \overline{\widehat{\mathbf{1}_{A_j}}(\omega)} e^{2\pi i \langle \omega, y \rangle} d\omega
\end{aligned}$$

Rotational Gradient

Parameterize $R = \exp(\mathfrak{r})$, where $\mathfrak{r} \in \mathfrak{so}(d)$ with basis $\mathfrak{r}_{k,l}$

In otherwords $d \times d$ skew symmetric matrices, $\mathfrak{r}^T = -\mathfrak{r}$

$$\begin{aligned} \frac{\partial C_{i,j}}{\partial \mathfrak{r}_{k,l}} &= \frac{\partial}{\partial \mathfrak{r}_{k,l}} \left(\int\limits_{\mathbb{R}^d} \widehat{\mathbf{1}_{A_i}}(\exp(-\mathfrak{r})\omega) \overline{\widehat{\mathbf{1}_{A_j}}(\omega)} e^{2\pi i \langle \omega, y \rangle} d\omega \right) \\ \dots &= \int\limits_{\mathbb{R}^d} - \left\langle \nabla \widehat{\mathbf{1}_{A_i}}(R^{-1}\omega), R^{-1}\mathfrak{r}_{k,l}\omega \right\rangle \overline{\widehat{\mathbf{1}_{A_j}}(\omega)} e^{2\pi i \langle \omega, y \rangle} d\omega \end{aligned}$$

Not a Fourier multiplier.

Rotational Gradient

Parameterize $R = \exp(\mathfrak{r})$, where $\mathfrak{r} \in \mathfrak{so}(d)$ with basis $\mathfrak{r}_{k,l}$

In otherwords $d \times d$ skew symmetric matrices, $\mathfrak{r}^T = -\mathfrak{r}$

$$\begin{aligned} \frac{\partial C_{i,j}}{\partial \mathfrak{r}_{k,l}} &= \frac{\partial}{\partial \mathfrak{r}_{k,l}} \left(\int\limits_{\mathbb{R}^d} \widehat{\mathbf{1}_{A_i}}(\exp(-\mathfrak{r})\omega) \overline{\widehat{\mathbf{1}_{A_j}}(\omega)} e^{2\pi i \langle \omega, y \rangle} d\omega \right) \\ \ldots &= \int\limits_{\mathbb{R}^d} - \left\langle \nabla \widehat{\mathbf{1}_{A_i}}(R^{-1}\omega), R^{-1}\mathfrak{r}_{k,l}\omega \right\rangle \overline{\widehat{\mathbf{1}_{A_j}}(\omega)} e^{2\pi i \langle \omega, y \rangle} d\omega \end{aligned}$$

Not a Fourier multiplier.

But can be precalculated with $O(d)$ overhead:

$$\frac{\partial}{\partial \omega^k} \widehat{\mathbf{1}_{A_i}}(\omega) = \mathcal{F}\left(-2\pi i \left\langle x, v^k \right\rangle \mathbf{1}_{A_i} \right)(\omega)$$

Performance

Fourier transform is neat, but what does it get us?

Performance

Fourier transform is neat, but what does it get us?

Full inverse Fourier transform complexity = brute force volume

Though the derivative formulas are cute...

And we do get slightly better cache coherency...

Performance

Fourier transform is neat, but what does it get us?

Full inverse Fourier transform complexity = brute force volume

Though the derivative formulas are cute...

And we do get slightly better cache coherency...

But do we really need the whole spectrum?

Performance

Fourier transform is neat, but what does it get us?

Full inverse Fourier transform complexity = brute force volume

Though the derivative formulas are cute...

And we do get slightly better cache coherency...

But do we really need the whole spectrum?

NO!

Performance

Fourier transform is neat, but what does it get us?

Full inverse Fourier transform complexity = brute force volume

Though the derivative formulas are cute...

And we do get slightly better cache coherency...

But do we really need the whole spectrum?

NO!

Since Fourier series exhibit exponential convergence for suitably regular functions, only need about $O(-log(h))$ terms to get $O(h)$ accuracy (in the L^1 norm).

Performance

Fourier transform is neat, but what does it get us?

Full inverse Fourier transform complexity = brute force volume

Though the derivative formulas are cute...

And we do get slightly better cache coherency...

But do we really need the whole spectrum?

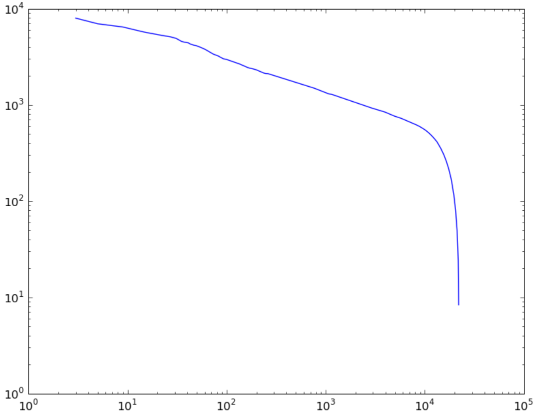
NO!

Since Fourier series exhibit exponential convergence for suitably regular functions, only need about $O(-log(h))$ terms to get $O(h)$ accuracy (in the L^1 norm).

Dramatically reduces space and time complexity

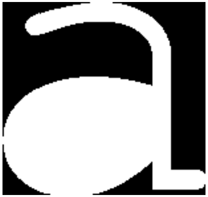
Convergence

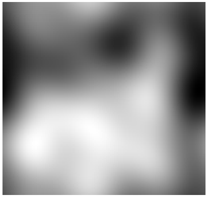
Rate of convergence, number of Fourier terms vs. L^1 error, (log-log plot)

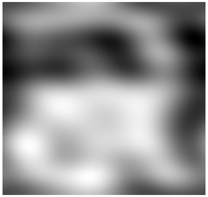


Convergence

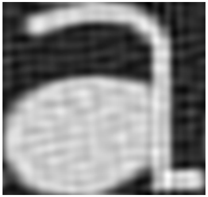
Qualitative effects of cutoff (full spectrum has 22185 terms)

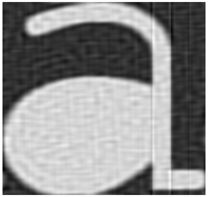

Exact


16


32

64


256


1024

Truncated Spectra

Henceforth $d = 2$

Want: An efficient way to represent truncated spectrum

Need to deal with rotations too

Obvious answer: Polar coordinates

Pick $\omega_r = \sqrt{\omega_x^2 + \omega_y^2}$, $\omega_\theta = \tan^{-1} \frac{y}{x}$

$$\widehat{f}(\omega_x, \omega_y) \mapsto \widehat{f}^p(\omega_r, \omega_\theta)$$

Truncated Spectra

Henceforth $d = 2$

Want: An efficient way to represent truncated spectrum

Need to deal with rotations too

Obvious answer: Polar coordinates

Pick $\omega_r = \sqrt{\omega_x^2 + \omega_y^2}$, $\omega_\theta = \tan^{-1} \frac{y}{x}$

$$\widehat{f}(\omega_x, \omega_y) \mapsto \widehat{f}^p(\omega_r, \omega_\theta)$$

Rotation $\mapsto$ Cyclic shift in θ
Differentiation formulas still work, just do change of variables
Seems easy, right?

Truncated Spectra

Henceforth $d = 2$

Want: An efficient way to represent truncated spectrum

Need to deal with rotations too

Obvious answer: Polar coordinates

Pick $\omega_r = \sqrt{\omega_x^2 + \omega_y^2}$, $\omega_\theta = \tan^{-1} \frac{y}{x}$

$$\widehat{f}(\omega_x, \omega_y) \mapsto \widehat{f}^p(\omega_r, \omega_\theta)$$

Rotation $\mapsto$ Cyclic shift in θ

Differentiation formulas still work, just do change of variables

Seems easy, right?

Problem?

Discrete Cartesian to Polar Conversion

Can't do this exactly!

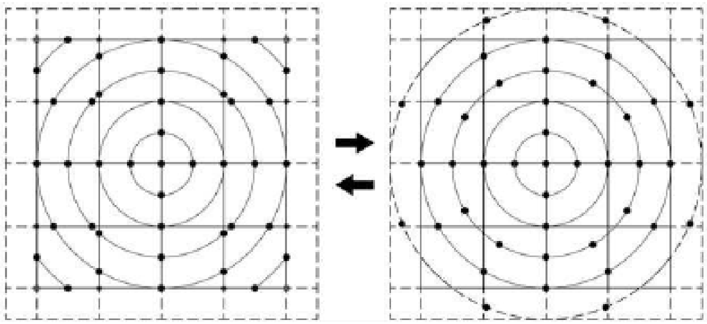


Figure from Chirikjian 2003

Discrete Cartesian to Polar Conversion

Can't do this exactly!

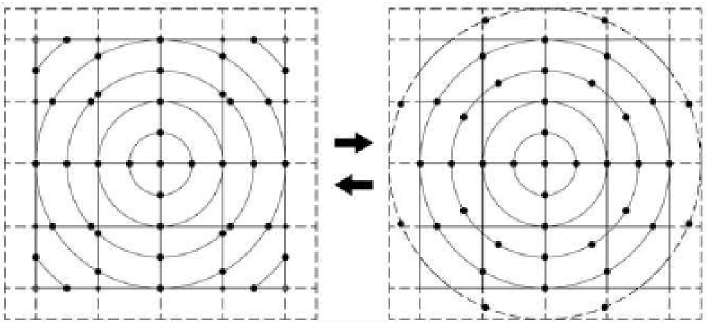
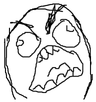


Figure from Chirikjian 2003

Reason: *Only nontrivial discrete subgroups of $SE(2)$ are 17 plane tiling groups!*



Discrete Cartesian to Polar Conversion

Can't do this exactly!

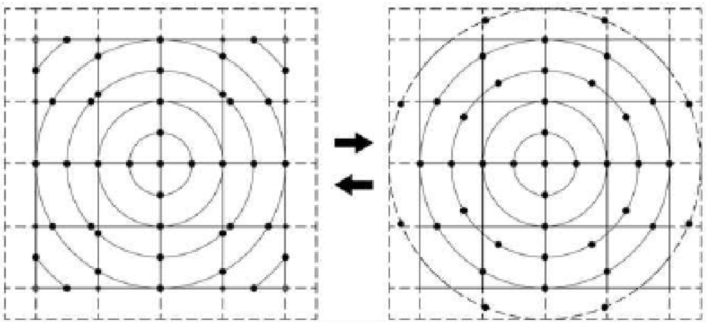
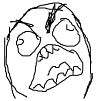


Figure from Chirikjian 2003

Reason: *Only nontrivial discrete subgroups of $SE(2)$ are 17 plane tiling groups!*



So unless you are happy with a hexagonal grid with 6 rotational samples, forget about doing this losslessly!

Bresenham Interpolation

- Approximation is a fact of life – deal with it
  - Might as well do it fast

Bresenham Interpolation

Approximation is a fact of life – deal with it

- Might as well do it fast

Idea: Do circle drawing in reverse, use Bresenham's algorithm

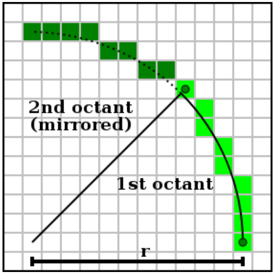


Figure from Wikipedia, 2010

Bresenham Interpolation

Approximation is a fact of life – deal with it

Might as well do it fast

Idea: Do circle drawing in reverse, use Bresenham's algorithm

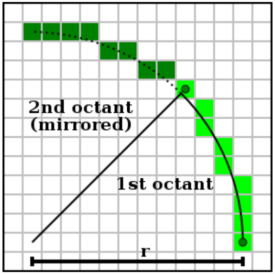


Figure from Wikipedia, 2010

- Sample r at uniform increments
- For each r, trace a circle centered at the origin
- Store $8r + 4$ radial samples in array
- Interpolate to uniform radial samples
- Append to store

Bresenham Interpolation

Approximation is a fact of life – deal with it

Might as well do it fast

Idea: Do circle drawing in reverse, use Bresenham's algorithm

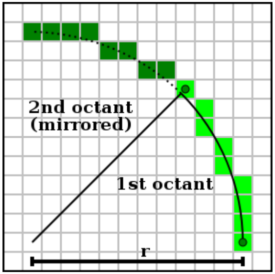


Figure from Wikipedia, 2010

- Sample r at uniform increments
- For each r, trace a circle centered at the origin
- Store $8r + 4$ radial samples in array
- Interpolate to uniform radial samples
- Append to store

Minimal storage, uses only integer arithmetic

Gibbs Phenomenon

- Truncating Fourier series causes ringing artifacts at boundaries
  - Intuitively, same as low pass filtering high curvature

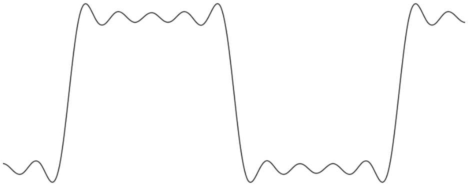


Figure from Wikipedia, 2010

Gibbs Phenomenon

Truncating Fourier series causes ringing artifacts at boundaries

- Intuitively, same as low pass filtering high curvature

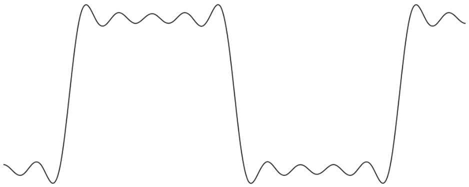


Figure from Wikipedia, 2010

But we don't care, just need separation between interior and exterior.

So pick some cutoff threshold; values above are in, below are out

Optimal Thresholding

Given a region $R \subset \mathbb{R}^d$ and set $A \subseteq R$ approximated by f, define quality, Q_T, of threshold T

$$Q_T = \text{vol}\,\{x \in R | f(x) > T \Leftrightarrow x \in A\}$$

Intuitively, number of points where threshold gives the right answer
Want to find optimal T

$$T = \operatorname*{argmax}_{T' \in \mathbb{R}} Q_{T'}$$

Seems hard to estimate T analytically

Optimal Thresholding

Given a region $R \subset \mathbb{R}^d$ and set $A \subseteq R$ approximated by f, define quality, Q_T, of threshold T

$$Q_T = \text{vol}\,\{x \in R | f(x) > T \Leftrightarrow x \in A\}$$

Intuitively, number of points where threshold gives the right answer
Want to find optimal T

$$T = \operatorname*{argmax}_{T' \in \mathbb{R}} Q_{T'}$$

Seems hard to estimate T analytically

But in practice can compute T combinatorially given f, A, R

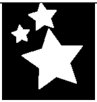
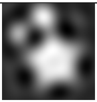
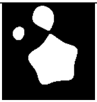
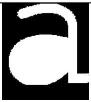
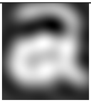
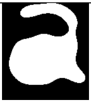
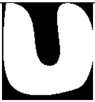
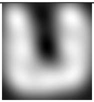
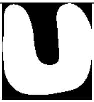
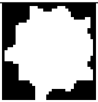
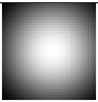
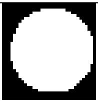
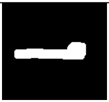
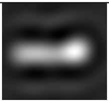
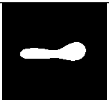
Optimal Thresholding Algorithm

```
v = []
for x in R:
    v.append( (f(x), A(x)) )
v.sort()

fv, Av = unzip(v)
Q = foldl(Av, +, 0) + foldr(Av, +, 0)

return max(zip(Q, fv)).second
```

Greedy/DP method

Takes $O(n\log(n))$ time, uses $O(n)$ space

Thresholding Results

Exact Indicator	Fourier Truncation	Optimal Cutoff
		
		
		
		
		

Radius 14 cutoff, 777 terms

Putting it together

Let $\hat{f}_A$ be the approximate Fourier series for $\mathbf{1}_A$

Then:

$$
\begin{aligned}
C_{i,j}(\rho,\theta,\phi) &= -T + \sum_{r=2}^{N_R}\sum_{t=1}^{8r+4} \widehat{f_{A_i}}\left(r, \phi\frac{dt}{d\theta} + t\right)\overline{\widehat{f_{A_j}}}(r,t) \\
\ldots &\quad \exp\left(\frac{2\pi i}{N}\rho r\cos\left(t\frac{d\theta}{dt} + \theta\right)\right) r\frac{dt}{d\theta}
\end{aligned}
$$

Where:

$$q_j q_i^{-1} = (R, y)$$

$$y = \rho(\cos(\theta)v^x + \sin(\theta)v^y)$$

$$R = \exp(\phi\mathfrak{r}_{0,1})$$

$$dt = 8r + 4$$

$$d\theta = 2\pi$$

Putting it together

Let $\hat{f}_A$ be the approximate Fourier series for $\mathbf{1}_A$

Then:

$$\begin{aligned} C_{i,j}(\rho,\theta,\phi) &= -T + \sum_{r=2}^{N_R} \sum_{t=1}^{8r+4} \widehat{f_{A_i}}\left(r, \phi\frac{dt}{d\theta} + t\right) \overline{\widehat{f_{A_j}}}(r,t) \\ \ldots & \quad \exp\left(\frac{2\pi i}{N}\rho r \cos\left(t\frac{d\theta}{dt} + \theta\right)\right) r\frac{dt}{d\theta} \end{aligned}$$

Where:

$$q_j q_i^{-1} = (R, y)$$

$$y = \rho(\cos(\theta)v^x + \sin(\theta)v^y)$$

$$R = \exp(\phi \mathfrak{r}_{0,1})$$

$$dt = 8r + 4$$

$$d\theta = 2\pi$$

Protip: Can early out using the rearrangement inequality

Prototype Physics Code

Basic implementation so far, just uses implicit Euler.
Doesn't solve for μ correctly (currently is constant)

Insert image

Features:

- OpenGL GUI
- Written 100% in Python
- Reasonable performance (though certainly not fast)
- Arbitrary shapes/mass fields supported

Download Link:

http://github.com/mikolalysenko/Collisions

To Do

- Analysis is still very incomplete
- Need to implement a proper solver for physics system
- Benchmarks, etc.
- More development/write ups needed