Animation & Robotics Summary

Lecture 01 – Introduction To Optimization

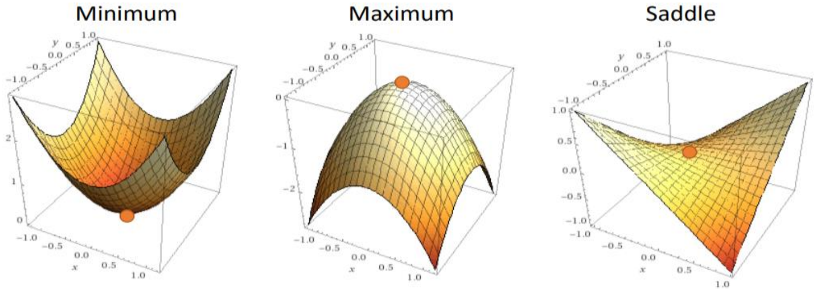
Optimization Problems:

- $\min_x f(x) - The\ Minimal\ Value\ of\ f$
- $\max_x f(x) - The\ Maximal\ Value\ of\ f$
- $\underset{x}{\operatorname{argmin}} f(x) - x^* where\ f(x^*)\ is\ minimal$
- $\underset{x}{\operatorname{argmax}} f(x) - x^* where\ f(x^*)\ is\ maximal$

Let's seek on $\underset{x}{\operatorname{argmin}} f(x) - x^* where\ f(x^*)\ is\ minimal$ problem. Finding $x^*\ s.t\ f(x^*) < f(x)\ \forall x$.

In 1D, $f(x)' = 0$ making the points that fulfill this to be נקודות קיצון where $f''(x) > 0$.

In nD, $\nabla f = \begin{pmatrix} \frac{\vartheta f(x)}{\vartheta x_1} \\ \frac{\vartheta f(x)}{\vartheta x_2} \\ \frac{\vartheta f(x)}{\vartheta x_3} \\ \vdots \\ \frac{\vartheta f(x)}{\vartheta x_n} \end{pmatrix} = 0_n . f(x)'' = hessian\ matrix$

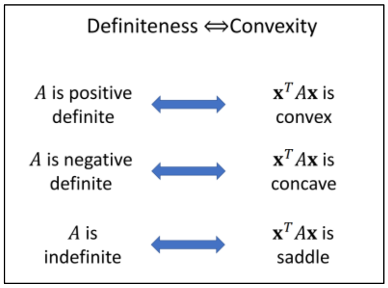


A quadratic form is a function of the form $f(x) = x^T A x$:

I. $A = I \rightarrow f(x) = x^T x = x_1^2 + \cdots + x_n^2$

II. $A = Diagonal(\lambda_1, \dots, \lambda_n) \rightarrow f(x) = \lambda_1 x_1^2 + \cdots + \lambda_n x_n^2$

NOTE : $\begin{cases} \forall \lambda_i > 0 \rightarrow convex \\ \forall \lambda_i < 0 \rightarrow concave \\ otherwise \rightarrow convex \end{cases}$



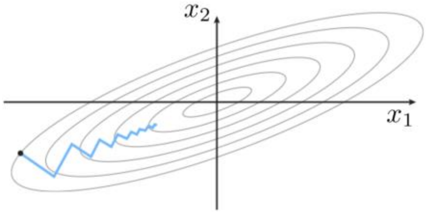
Definiteness ⟺Convexity

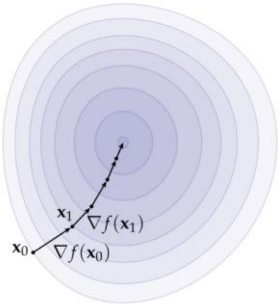
A is positive definite ⟷ $\mathbf{x}^T A\mathbf{x}$ is convex

A is negative definite ⟷ $\mathbf{x}^T A\mathbf{x}$ is concave

A is indefinite ⟷ $\mathbf{x}^T A\mathbf{x}$ is saddle

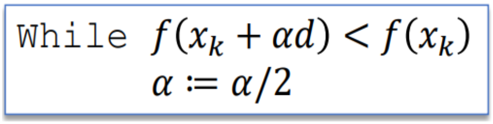
1. Gradient Descent
   Objective : Gradient is direction of steepest ascent. Function value gets largest if we move in direction of gradient, it doesn't change if we move orthogonally but it decreases fastest if we move exactly in opposite direction



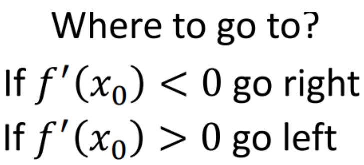


2. Line Search
   Objective : Given x_k and a step direction d, determine step size α such that :

$$f(x_k + \alpha d) < f(x_K)$$



```
While $f(x_k + \alpha d) < f(x_k)$
    $\alpha := \alpha/2$
```



Where to go to?

If $f'(x_0) < 0$ go right

If $f'(x_0) > 0$ go left

3. Newton Step
   Objective : Start with Taylor expansion, $f(x) = f(x_0) + \nabla f(x_0)^T p + \frac{1}{2} p^T \nabla^2 f(x_0) p$ then We want the minimum, so we take the gradient and look for 0.

$$\nabla f(x_0) + \nabla^2 f(x_0) p = 0 \rightarrow p = -\left(\nabla^2 f(x_0)\right)^{-1} \nabla f(x_0)$$

NOTES:

1. In general, Newton's method requires many less iterations to converge.
2. Newton's method only work if the Hessian is positive-definite.
3. In general, Newton's method is preferable.

Lecture 02 – Soft Body Simulation

Definition *: Elasticity The ability of a material to resist a deforming force and to return to its original size and shape when that force is removed.*

- Hooke's Law : $F = -k(l - L)$

- L *is the length of the undeformed spring*
- l *is the length of deformed spring*
- k *is spring stifness*

- Spring Energy : $E = \frac{1}{2}k(l - L)^2$

Equilibrium : A physical system will always reach a state which no further change is possible. Newton Law on Equilibrium : $F_i = \sum_{e_{i,j}} F_{i,j} = 0$, $\forall i$ and the energy reaches a minimum state : $argmin(E) = \sum_{e_{i,j}} E_{i,j}$ whereas $e_{i,j}$ present a spring between vertices i, j.

Spring System :

$$E\begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \frac{1}{2}k\left(\left||x_1 - x_2|\right| - L\right)^2$$

$$G\begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \vartheta_{x_1}E \\ \vartheta_{y_1}E \\ \vartheta_{x_2}E \\ \vartheta_{y_2}E \end{pmatrix} = \frac{k}{\left||x_1 - x_2|\right|}\begin{pmatrix} x_1 - x_2 \\ x_2 - x_1 \end{pmatrix}\left[\left||x_1 - x_2|\right| - L\right]$$

Gradient/Hessian Matrix Calculation :

```
$init\ G = 0_{2n\times1}, H = 0_{2n\times2n}$
$for\ each\ edge\ e_{i,j}$ :
$1. Call\ G_i = ComputeGradient(x_i, x_j)$
$2. Add\ G_i\ To\ G(2i, 2i + 1, 2j, 2j + 1)$
$3. Call\ H_i = ComputeHessian(x_i, x_j)$
$4. Add\ H_i\ To\ The\ Correct\ Places\ in\ H$
```

Definition: <u>Compressed Column Storage (CCS)</u> ,a sparse matrix or sparse array is a matrix in which most of the elements are zero.

Lecture 03 – Deformations

There are three common Metaphors : Point, Skeleton and Cage.

Map 2D representation : $f(x,y) \to \mathbb{R}^2$; for example, $f(x,y) = \left(x + \frac{x}{x^2+y^2}, y + \frac{y}{x^2+y^2}\right)$.
The problem with this representation is that setting maps is big too handle. Therefore, representing maps as linear combination of basic functions.

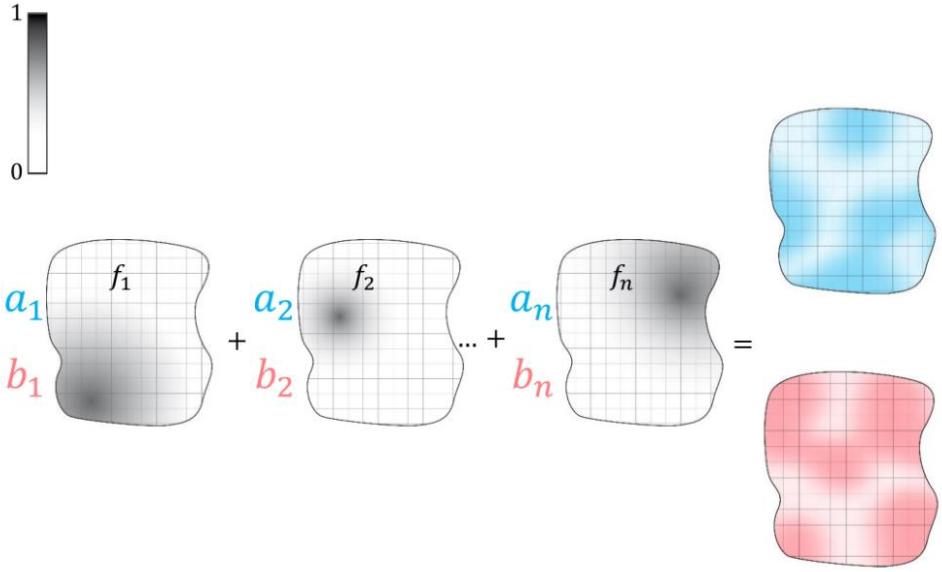


Figure 1: linear combination of basic functions

It can also be presented as $f(x \in \mathbb{R}^n) = \begin{pmatrix} \sum a_i f_i(x) \\ \sum b_i f_i(x) \end{pmatrix} = (\sum c_i f_i(x)\ )$.

Locally Bijective versus Globally Bijective;

***Definition** : A function is said to be bijective or bijection, if a function $f: A \ \to\ B$ satisfies both the injective (one-to-one function) and surjective function (onto function) properties.*

Question: if a f present a linear combination of 2d mapping considered to be locally bijective (Each interior function is bijective) would the function f consider to be globally bijective? Vice versa?

Answer : No, if you are interested “Global Inversion Theorems” even though there are some cases that fulfill that. But the other direction is yes(!).

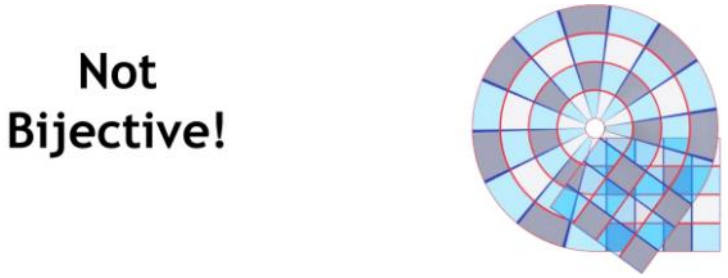


Figure 2 : This Mapping cosidered to be not globally bijective but locally bijective.

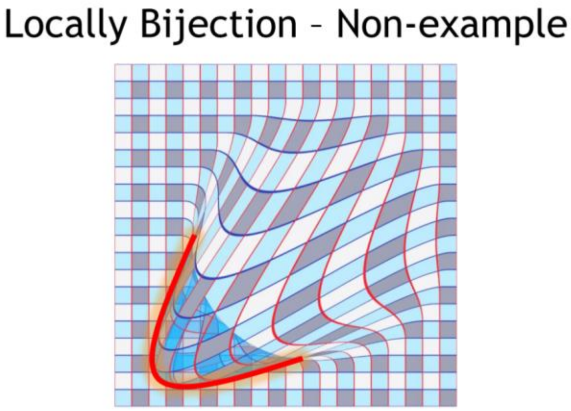


Figure 3 : Not Locally Bijective.

Condition for Locally Bijective; suppose the function of a deformation is $f(x) = \begin{pmatrix} u(x) \\ v(x) \end{pmatrix}$. The Jacobian Matrix would be $\Im f(x) = \begin{pmatrix} \nabla u(x) \\ \nabla u(x) \end{pmatrix} = \begin{pmatrix} \vartheta_x u(x) & \vartheta_y u(x) \\ \vartheta_x v(x) & \vartheta_y v(x) \end{pmatrix}$. The function $f(x)$ is locally bijective if $\det(\Im f(x)) > 0\,, \forall x$

Good Mapping : an example,

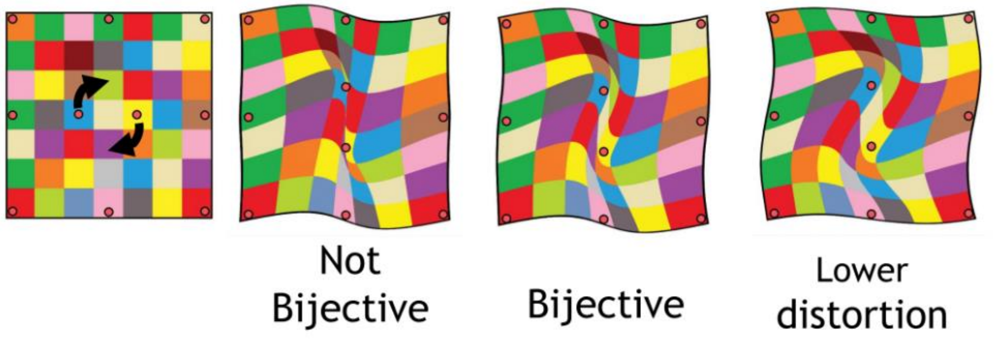


Figure 4 : in the second map is not bijective due to data loss

Definition *: Distortion is a function of the Jacobian at a point. There two types of distortion; Conformal distortion and Isometric distortion.*

Visually : The lengths in Conformal distortion are changed and the angles of the square/rectangle remained 90º.

Visually : The lengths in Isometric distortion are kept and the angles of the square/rectangle has shifted to parallelogram.

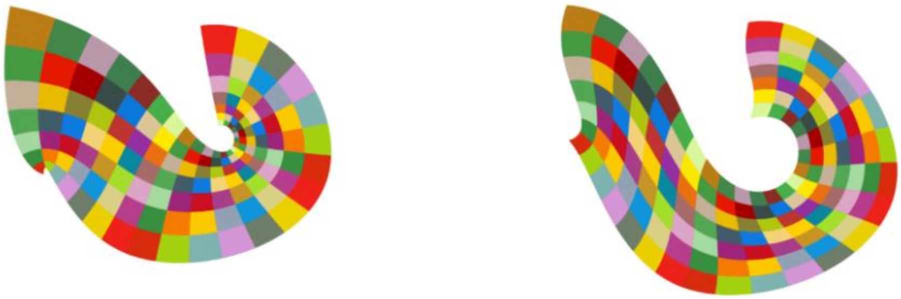
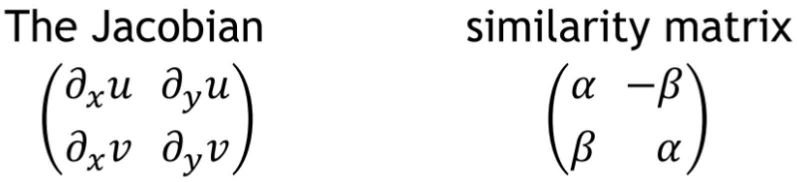


Figure 5 : Conformal distortion (on left) versus Isometric distortion (on right)

NOTE : D represents a deformation map.

- LSCM (Least Squares Conformal Map)



The Jacobian

$$\begin{pmatrix} \partial_x u & \partial_y u \\ \partial_x v & \partial_y v \end{pmatrix}$$

similarity matrix

$$\begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix}$$

Figure 6 : Jacobian Vs Similarity Matrices

$$\vartheta_x u = \vartheta_y v \rightarrow \vartheta_x u - \vartheta_y v = 0$$

$$-\vartheta_x v = \vartheta_y u \rightarrow \vartheta_x v + \vartheta_y u = 0$$

Cauchy Riemann Equations

- ASAP (As Similar As Possible) :

$$Min\left|\left|\Im - S\right|\right|_F^2 \ where\ S = \begin{pmatrix} \alpha & -\beta \\ \beta & \alpha \end{pmatrix} = \frac{1}{2}\begin{pmatrix} a+d & c-b \\ b-c & a+d \end{pmatrix}$$

$$\boxed{ASAP\ =\ LSCM}$$

- ARAP (As Rigid As Possible) :

$$D_{ARAP} = \left|\left|\mathfrak{I} - R_A\right|\right|_F^2 = (\sigma_1 - 1)^2 + (\sigma_2 - 1)^2$$

$$D_{ASAP} = \left|\left|\mathfrak{I} - S_A\right|\right|_F^2 = (\sigma_1 - \sigma_2)^2$$

Distortion On Triangle Meshes:

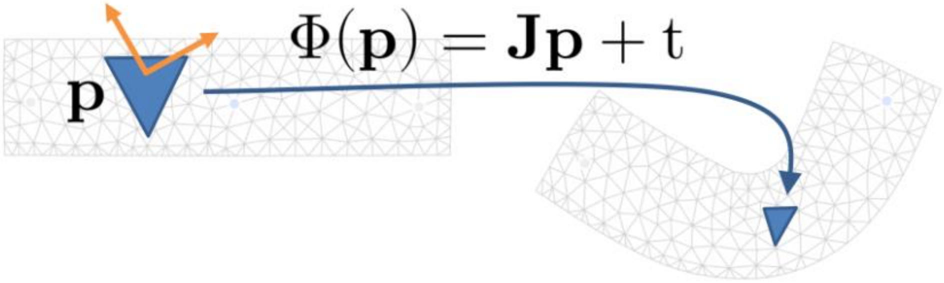


Figure 7 : Distortion On Triangle Meshes

- Conformal (LCSM) :

$$D(\mathfrak{I}) = \left|\left|\mathfrak{I} + \mathfrak{I}^2 - (tr\ \mathfrak{I})I\right|\right|_F^2 = (\sigma_1 - \sigma_2)^2 \mid \sigma_1 = \sigma_2$$

- Isometric (ARAP) :

$$D(\mathfrak{I}) = \left|\left|\mathfrak{I} - R\right|\right|_F^2 = (\sigma_1 - 1)^2 + (\sigma_2 - 1)^2 \mid \sigma_1, \sigma_2 = 1$$

- Authalic :

$$D(\mathfrak{I}) = \left|\left|\mathrm{Det}(\mathfrak{I}) - 1\right|\right|_F^2 = (\sigma_1\sigma_2 - 1)^2 \mid \sigma_1\sigma_2 = 1$$

Lecture 04 – Transformation:

1. Scaling : Let $v = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$ a vertex, Scaled vertex would be like $S: \mathbb{R}^2 \to \mathbb{R}^2$ s.t $S^{s_y, s_x}(v) = \begin{pmatrix} s_x v_x \\ s_y v_y \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}\begin{pmatrix} v_x \\ v_y \end{pmatrix}$. A case when $s_x = s_y$ called uniform/isotropic scaling(preserve 1:1 ratio).
2. Rotation : Let $v = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$ a vertex, Rotated vertex would be like $R: \theta \to \mathbb{R}^2$ s.t $w^\theta(v) = \begin{pmatrix} w_x \\ w_y \end{pmatrix} = \begin{pmatrix} v_x cos\theta - v_y sin\theta \\ v_x sin\theta + v_y cos\theta \end{pmatrix}$ whereas matrix representation be like $R^\theta = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix}$

3. Translation : Let $v = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$ a vertex, Translated vertex would be like $T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ s.t $T^{t_x,t_y}(v) = \begin{pmatrix} v_x + t_x \\ v_y + t_y \end{pmatrix}$ whereas matrix representation requires shifting to Homogeneous Coordination by added a z-value for v. Therefore, Matrix transformation would shape like:
$$\begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}$$
4. Shear : $(\begin{matrix} 1 & \alpha \\ 0 & 1 \end{matrix})$

NOTE : Matrices Order is IMPORTANT!

Singular Value Decomposition (SVD): Every Matrix M has factorization of the form $M = USV^T$ *where* $S = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}$ $s.t\ \sigma_1 > \sigma_2$.

Lecture 05 – Rigid Motion:

1. Translation : upon $t_a = 0$ the mesh located at point a such that $a = (0,0)$. At $t_b = 1$ mesh is transferred to point b which located at $b = (4,0)$.

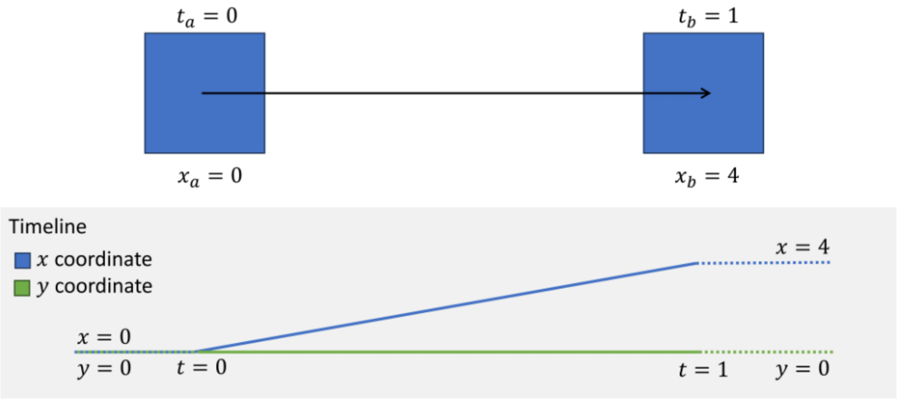


Figure 8 : Translation from A To B in timeline

- At $t_a = 0$ Translate Matrix representation : $T_a = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
- At $t = 0.5$: Matrix would reach $T_a = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, $x_c = \frac{x_a}{2} + \frac{x_b}{2}$
- At $t_b = 1$ Matrix representation : $T_a = \begin{pmatrix} 1 & 0 & 4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

2. Rotation : upon $t_a = 0$ the mesh located at point had $\theta = 0$. Then the mesh is rotated to with $\theta = \pi$.

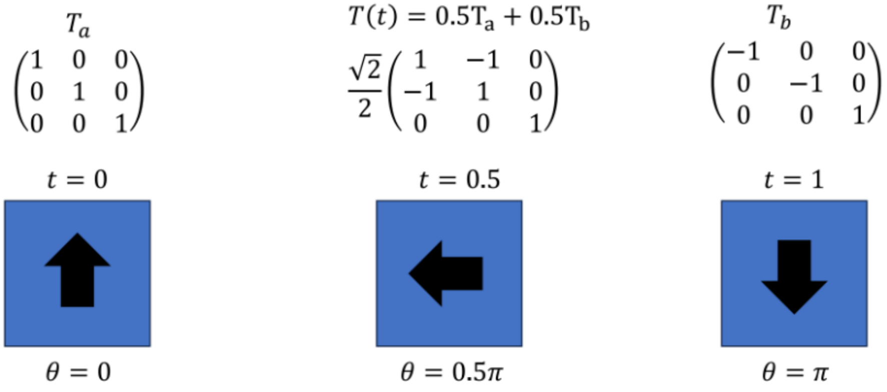
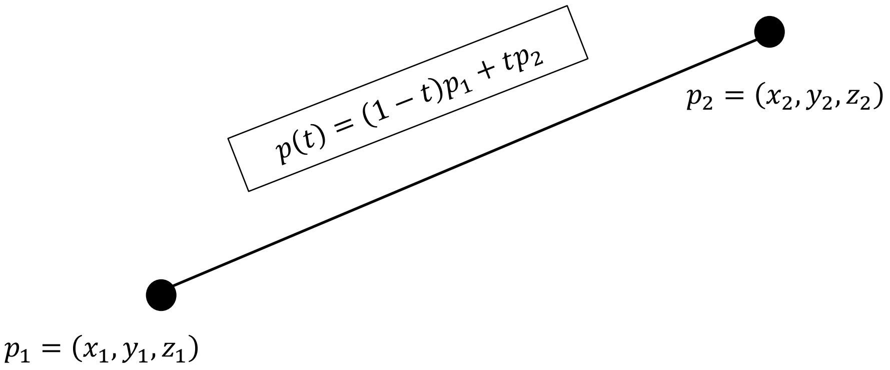

T_a
$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
$T(t) = 0.5\mathrm{T_a} + 0.5\mathrm{T_b}$
$\frac{\sqrt{2}}{2}\begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
T_b
$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
$t = 0$
$t = 0.5$
$t = 1$
$\theta = 0$
$\theta = 0.5\pi$
$\theta = \pi$

Figure 9 : Rotation timeline

As we notice in Translation, representation as vector and interpolation as lines.



What about Orientation? Representation as angles and interpolation as ..??

$$R \cdot R^T = I$$

Leaving 6 equations, 9 unknowns and 3 numbers. Therefore, 3 angles.

1. *X rotation by* α .
2. *Y rotation by* β .

3. $Z\ rotation\ by\ \gamma$.

Note that the order is IMPORTANT but ARBITRARY.

Definition : Extrinsic or intrinsic with respect to world coordinates or local coordinates. Extrinsic rotations about the axes xyz of the original coordinate system, which is assumed to remain motionless. Intrinsic rotations about the axes of the rotating coordinate system XYZ, solidary with the moving body, which changes its orientation with respect to the extrinsic frame after each elemental rotation.

- Gimbal Lock :

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix}, R_y(\beta) = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \beta \end{pmatrix}, R_z(\gamma) = \begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

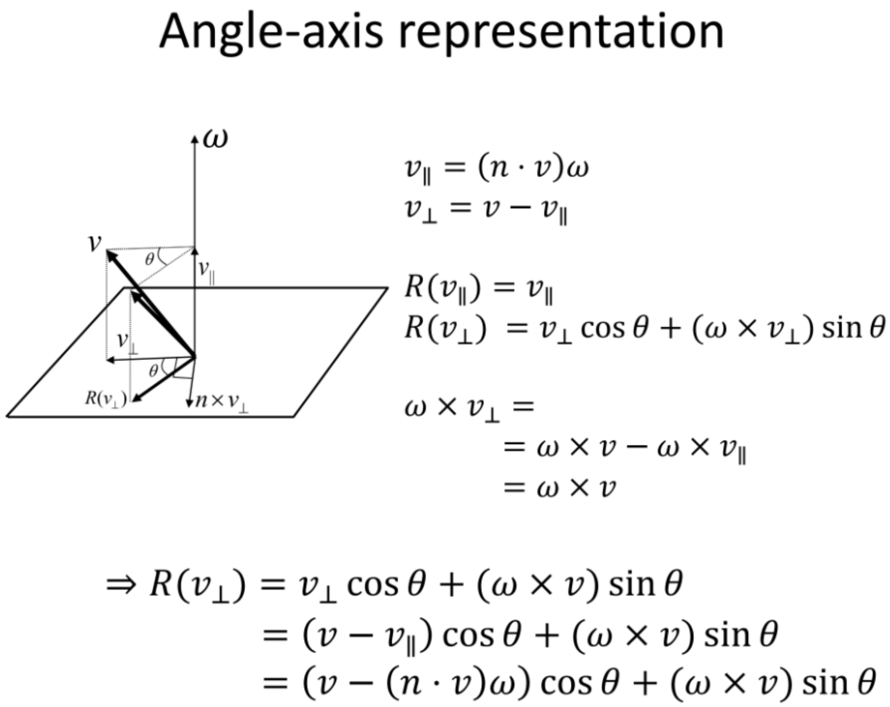
$$R(\alpha, \beta, \gamma) = R_z R_y R_x$$

Upon $\beta = \frac{\pi}{2}$, $R\left(\alpha, \frac{\pi}{2}, \gamma\right) = \begin{pmatrix} 0 & \sin\theta & \cos\theta \\ 0 & \cos\theta & \sin\theta \\ 1 & 0 & 0 \end{pmatrix}$ $s.t\ \theta = \alpha + \gamma$ resulting in loss of one degree of freedom .

Theorem: (Euler): Any orientation can be obtained from a fixed reference orientation by a single

Unique rotation around an appropriate axis in space.

Any orientation may be described by four parameters: angle θ and unit axis $w = (w_x, w_y, w_z)$.



Angle-axis representation

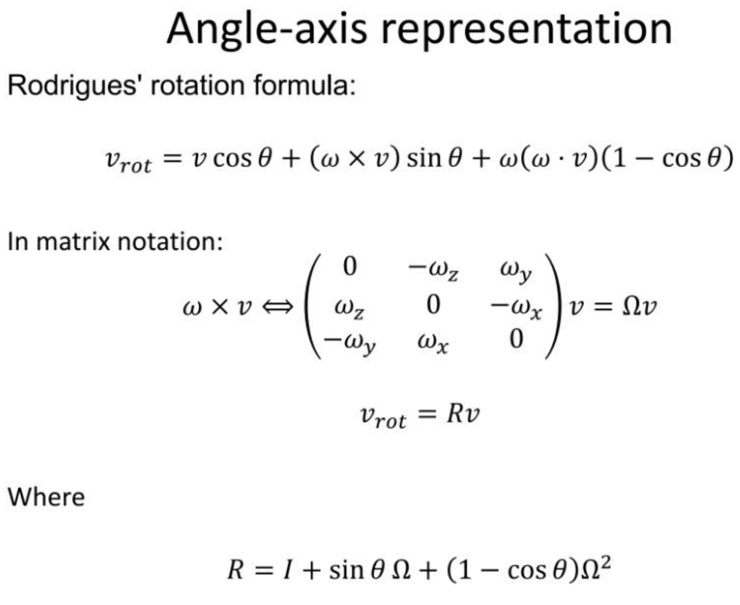
$$v_{\|} = (n \cdot v)\omega$$
$$v_{\perp} = v - v_{\|}$$

$$R(v_{\|}) = v_{\|}$$
$$R(v_{\perp}) \ = v_{\perp}\cos\theta + (\omega \times v_{\perp})\sin\theta$$

$$\begin{aligned} \omega \times v_{\perp} &= \\ &= \omega \times v - \omega \times v_{\|} \\ &= \omega \times v \end{aligned}$$

$$\begin{aligned} \Rightarrow R(v_{\perp}) &= v_{\perp}\cos\theta + (\omega \times v)\sin\theta \\ &= (v - v_{\|})\cos\theta + (\omega \times v)\sin\theta \\ &= (v - (n \cdot v)\omega)\cos\theta + (\omega \times v)\sin\theta \end{aligned}$$

Figure 10 : Angle-Axis representation



Angle-axis representation

Rodrigues' rotation formula:

$$v_{rot} = v\cos\theta + (\omega \times v)\sin\theta + \omega(\omega \cdot v)(1 - \cos\theta)$$

In matrix notation:

$$\omega \times v \Leftrightarrow \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} v = \Omega v$$

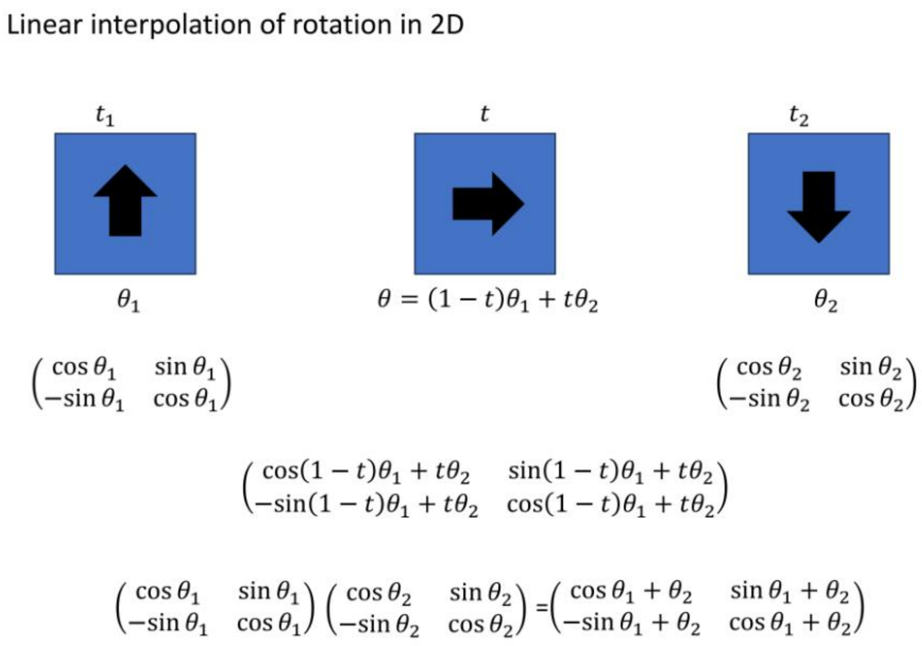
$$v_{rot} = Rv$$

Where

$$R = I + \sin\theta\,\Omega + (1 - \cos\theta)\Omega^2$$

Figure 11 : Angle-Axis representation via Rodrigues formula

So, Back to Rotation Interpolation :



Linear interpolation of rotation in 2D

t_1 $\quad$ t $\quad$ t_2

θ_1 $\quad$ $\theta = (1-t)\theta_1 + t\theta_2$ $\quad$ θ_2

$$\begin{pmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{pmatrix} \qquad \begin{pmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{pmatrix}$$

$$\begin{pmatrix} \cos(1-t)\theta_1 + t\theta_2 & \sin(1-t)\theta_1 + t\theta_2 \\ -\sin(1-t)\theta_1 + t\theta_2 & \cos(1-t)\theta_1 + t\theta_2 \end{pmatrix}$$

$$\begin{pmatrix} \cos\theta_1 & \sin\theta_1 \\ -\sin\theta_1 & \cos\theta_1 \end{pmatrix} \begin{pmatrix} \cos\theta_2 & \sin\theta_2 \\ -\sin\theta_2 & \cos\theta_2 \end{pmatrix} = \begin{pmatrix} \cos\theta_1 + \theta_2 & \sin\theta_1 + \theta_2 \\ -\sin\theta_1 + \theta_2 & \cos\theta_1 + \theta_2 \end{pmatrix}$$

Complex Numbers:

Multiplication : $(a+bi)(c+di) = (ac-bd) + (bc+ad) = r_1 e^{i\theta} \cdot r_2 e^{i\theta} = r_1 r_2 e^{i\theta}$

$$e^{i\theta} \leftrightarrow \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Therefore,

Linear Interpolation of $e^{i\theta_1}$ and $e^{i\theta_2}$:

$$e^{i((1-t)\theta_1 + t\theta_2)}$$

Quaternions – ATTEMPT OF EXTENDED 3D COMPLEX NUMBER

$$i^2 = j^2 = k^2 = ijk = -1$$

Definition : $q_0 + q_1 i + q_2 j + q_3 k = q_0 + \vec{q}$.

✓ Addition Operation

✓ Multiplication Operation

✓ Conjugate Operation

✓ Size

Rotation : Rotation by θ around unit direction w may be represented by the unit quaternion.

Example : $q = \left[\cos\frac{\theta}{2}, w\sin\frac{\theta}{2}\right]$ Therefore, The 3D Vector $[0, v]$ is rotated by q to $R_q(v) = q \cdot v \cdot q^{-1}$.

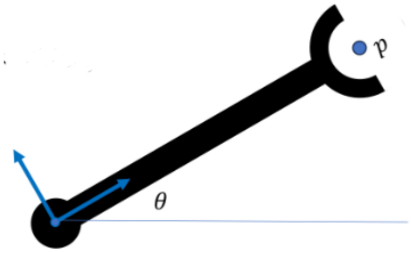
The shortest rotation Given rotation $R_1$and R_2 what is the "shortest" path between them?

For any two quaternions q_1 and q_2 the SLERP between them is :

$$q(t) = \frac{\sin\big((1-t)\varphi\big)}{\sin\varphi}q_0 + \frac{\sin(t\varphi)}{\sin\varphi}q_1$$

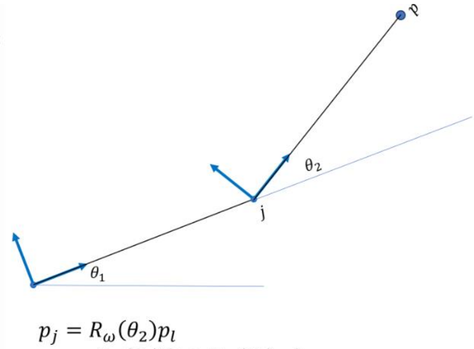
Which is equivalent to : $q(u) = q_0(q_0^{-1}q_1)^T$

Lecture 06 – Kinematics:



- One Link : Local frame aligns with parent joint axis.
  $p\ in\ local\ coordinates : p_l$
  $p\ in\ world\ coordinates : R_w(\theta)p_l = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} p_l$

- Two Link :



$$p_j = R_\omega(\theta_2)p_l$$
$$p_w = R_\omega(\theta_1)(j + R_\omega(\theta_2)p_l)$$

Figure 12 : Two Links

- Three Links : $R_w(\theta_1)\big(j_1 + R_w(\theta_2)(j_2 + R_w(\theta_3)p_l)\big)$
- Four Links : $R_w(\theta_1)\left(j_1 + R_w(\theta_2)\big(j_2 + R_w(\theta_3)(j_3 + R_w(\theta_4))\big)\right)$

Velocity Jacobian: How does a small change in joint angles change the position of p_w?

- One Link : $\frac{dp_w}{d\theta} = \hat{z} \times p_w = \omega \times p_w = \omega \times R_\omega(\theta)p_l$
- Two Link : $p_w = R_\omega(\theta_1)(j + R_\omega(\theta_2)p_l)$
- More than two: $J(\Theta) = \frac{dp_w}{d\Theta} = \nabla_\Theta p_w$

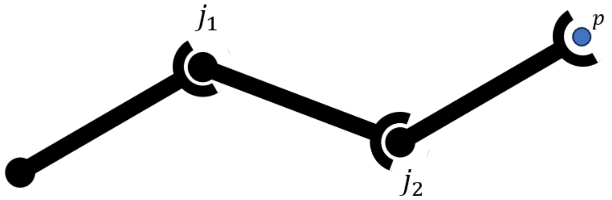
Forward kinematics on three links:

$$p_w = R_\omega(\theta_1)\big(j_1 + R_\omega(\theta_2)(j_2 + R_\omega(\theta_3)p_l)\big)$$

$$\frac{dp_w}{d\theta_1} = \omega \times R_\omega(\theta_1)\big(j_1 + R_\omega(\theta_2)(j_2 + R_\omega(\theta_3)p_l)\big)$$

$$\frac{dp_w}{d\theta_2} = R_\omega(\theta_1)\big(\omega \times R_\omega(\theta_2)(j_2 + R_\omega(\theta_3)p_l)\big)$$

$$\frac{dp_w}{d\theta_2} = R_\omega(\theta_1)R_\omega(\theta_2)(\omega \times R_\omega(\theta_3)p_l)$$

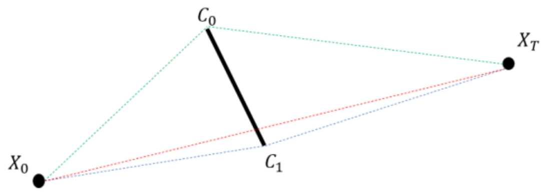


Inverse Kinematics : I will have to watch the lecture again to write it down.

Lecture 07 – Path Planning:

- Analytical method:

$$F(x) = \begin{cases} Red\ PATH & : \quad is\ valid \\ \min\{Blue\ PATH\ , Green\ PATH\} & : Otherwise \end{cases}$$



- Numerical method :
  1. **<u>Search-based algorithms (Grid-based algorithms)</u> :**
     - BFS(Time-Complexity $O(V + E)$) /Dijkstra(Time-Complexity $O(V + Elog(E))$)
     - A^* matrix where we define $g(n), h(n)\ and\ f(n)$ as the following:
       $g(n)$: same as Dijkstra. The real cost to reach a node n.
       $h(n)$: approximate cost from node n to goal node (heuristic function).
       $$f(n) = g(n) + h(n).$$
       Dijkstra is a special case for $A *$ (when the heuristi0cs is zero).

2. **Sampling-based algorithms (graph-based algorithms)**
   - Random Tree(RT) : Doesn’t Explore well.
   - Rapidly Random Tree (RRT) : Rapid Exploring but it is not OPTIMAL
   - Rapidly Random Tree * (RRT*) : is the same as RRT, but two key additions to the algorithm result in significantly different results :
     1. Cost Optimization: It records the travel cost from each vertex to its parent and selects the cheapest neighboring node within a fixed radius for connection, eliminating the cubic structure.
     2. Tree Rewiring: It rewires neighbors to a newly added vertex if this reduces their travel cost, resulting in a smoother path.

     Despite that it gives result closer to optimal solution, it takes High time complexity execution.
   - Probabilistic Roadmaps (PRM) : Builds the graph only once and find multiple paths, Not optimal and Clusters.

Lecture 08 – Trajectory Optimization: