Perlin Noise II

CMSC425.01 fall 2019

Administrivia

- Final project
  - Final project demonstration schedule (Monday, Dec. 16th, 1:30-3:30)
  - Alternative dates: Thursday, Dec. 12
    Friday, Dec. 13
    Monday, Dec. 16 (other times)

Final project rubric – from proposal handout

- A quality of planning and execution that can't be achieved in the last week.
- Work by all members of the team, documented by some record of your work schedule and individual contributions.
- Some innovation beyond copying an existing game, although it's not easy to be fully new in this space.
- Achievement relative to ambition. Try for something ambitious, and lack a little polish, ok. Try for less ambitious results, then make it look good.
- Non-trivial scripting, and scripts that aren't just copied as assets. Shapes and animations can be assets (although adding your own terrain or animation script would good.)

Final project rubric

Topic	Scoring 5/5	Weights
Concept – Clear, consistent, not just copy*	/5	15
Artistic – Consistent, good look (not mixed assets)	/5	15
Algorithmic – Non-trivial scripting somewhere	/5	15
Team work – everyone contributed, documented	/5	15
Completeness – All of it works, relative to ambition*	/5	20
Group size – more people, higher expectations	/5	10
Video – video is submitted, clear	/5	5
Report – report is submitted, clear and complete	/5	5
Intangibles – instructor overall opinion	/5	5
Total		100

Unity VR job

- ONE YEAR Temporary role- Great way to gain experience in VR!
- Space Telescope Science Institute is seeking a temporary Unity3D software developer to develop simulations and scenarios in virtual reality (VR). The successful applicant would work with experts in astrophysics and public outreach to develop immersive astrophysical scenes. No prior knowledge of astrophysics is required. We utilize accurate 3D models of space telescopes and simulated environments to interact with in real time. The Unity3D developer will be building the simulation apps for the HTC Vive and Oculus platforms, but may explore other platforms as required.

- As a part of the VR team, you will be:
  • Pioneering the future of astronomy learning and outreach by working on cutting-edge technologies
  • Building Virtual simulations and experimental prototypes
  • Working in C# and Unity3D to create clean, organized, optimized, reusable code modules that can be adapted for multiple projects
  • Working with multiple 3rd Party SDKs to integrate VR tracking, computer vision, and haptics into Unity
  • Building amazing 3D simulations using the latest in VR/ AR/MR technology

- Basic Qualifications:

  • Bachelor’s degree in Computer Science, Human Computer Interaction, or related field experience
  • Experience developing for Virtual Reality technologies.
  • Experience working in Unity3D and C#
  • Experience implementing 3D games and/or 3D simulations in Unity3D
  • Experience working with animated and rigged 3D models in Unity3D
  • Experience with 3D user interface design
  • Strong knowledge of OOP, design patterns, event delegates, asynchronous functions, data structures and algorithms
  • Experience working with a team in a professional capacity,
  • High standards for work quality, self-motivated, able to work with minimal supervision

- Desired Skills:
  • Experience with rendering pipelines on different platforms, profiling and optimization techniques in VR
  • MS degree in Computer Science, Human Computer Interaction, or related field experience
  • Experience with 3D Graphics
  • Experience with Shader Programming
  • Experience with 3D modeling tools (ex: Maya, 3D Max, or Blender)
  • Experience with Unity3D interfacing other external data elements
  • Experience with Augmented Reality and Android / iOS development a bonus

Today’s question

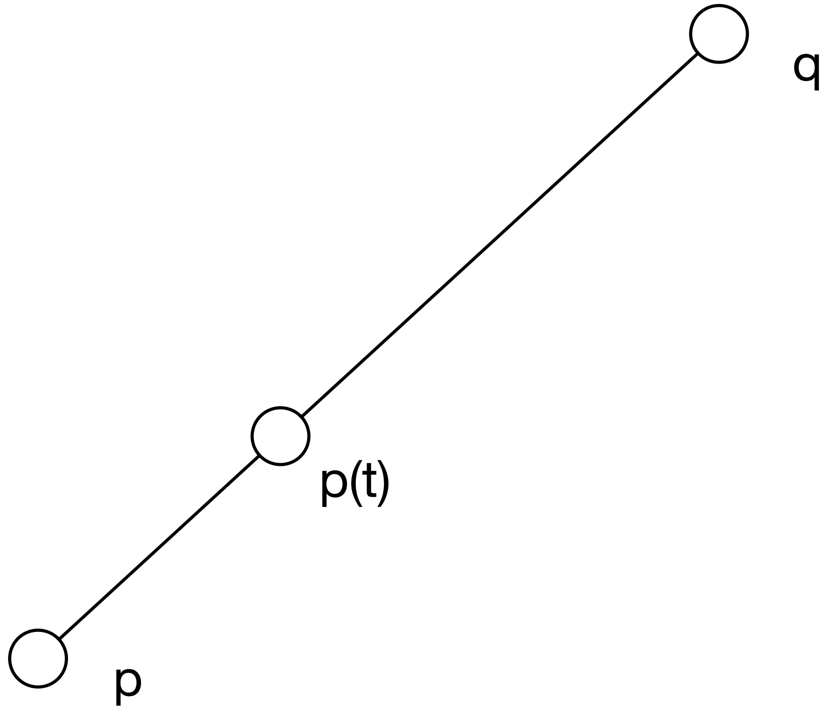
Perlin noise in 2D?

Parametric line segments

$$p(t) = p + tv$$

with $v = q - p$

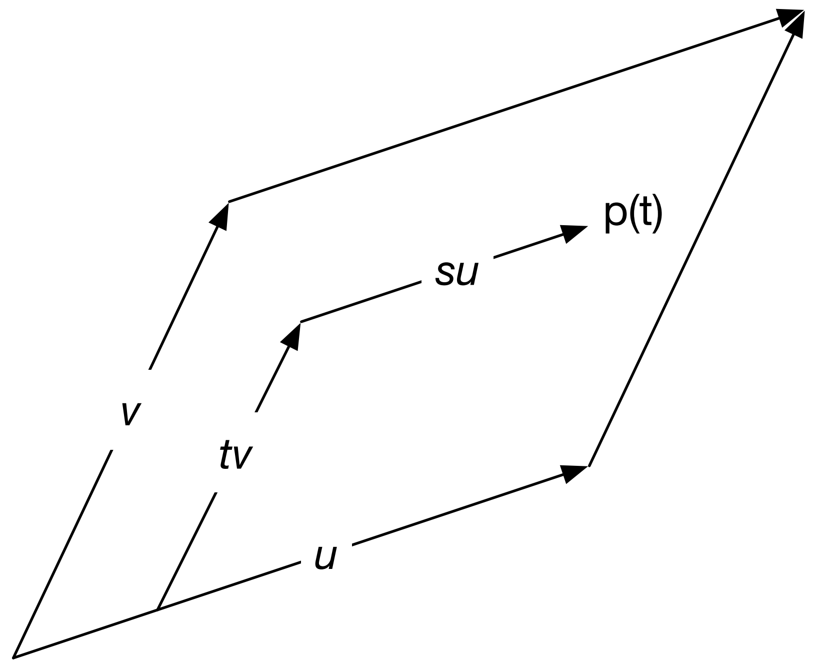
```
for t = 0 to 1 by deltat
     x = px + t * vx
     y = py + t * vy
     plot(x,y) // or line(lx,ly,x,y)
```



Parametric planar patches

$$p(s,t) = p + tv + su$$

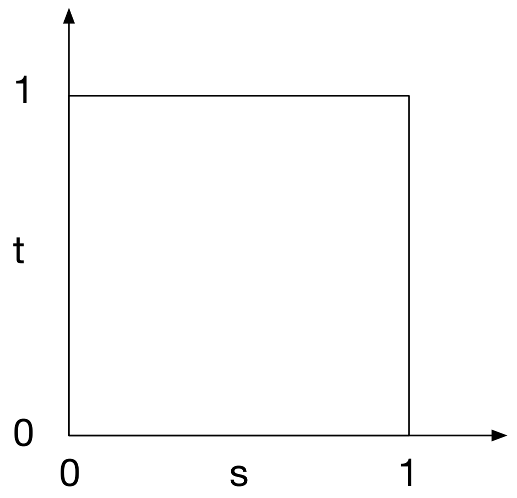
```
for t = 0 to 1 by deltat
     x = px + t * vx + s * ux
     y = py + t * vy + s * uy
     z = pz + t * vz + s * uz
     plot(x,y,z)
```

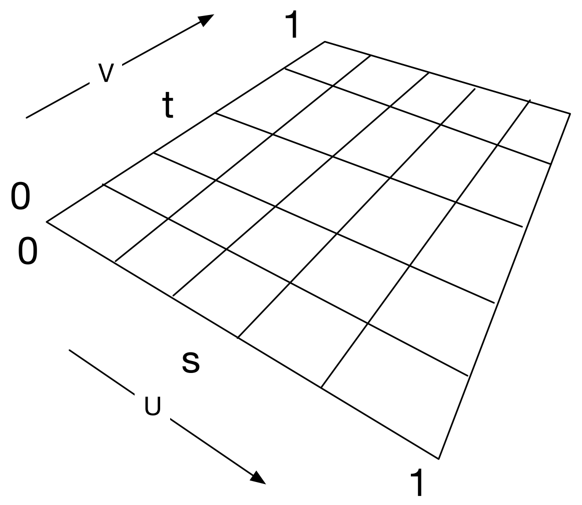


Creating planar mesh

- How create mesh data structure from parametric patch?

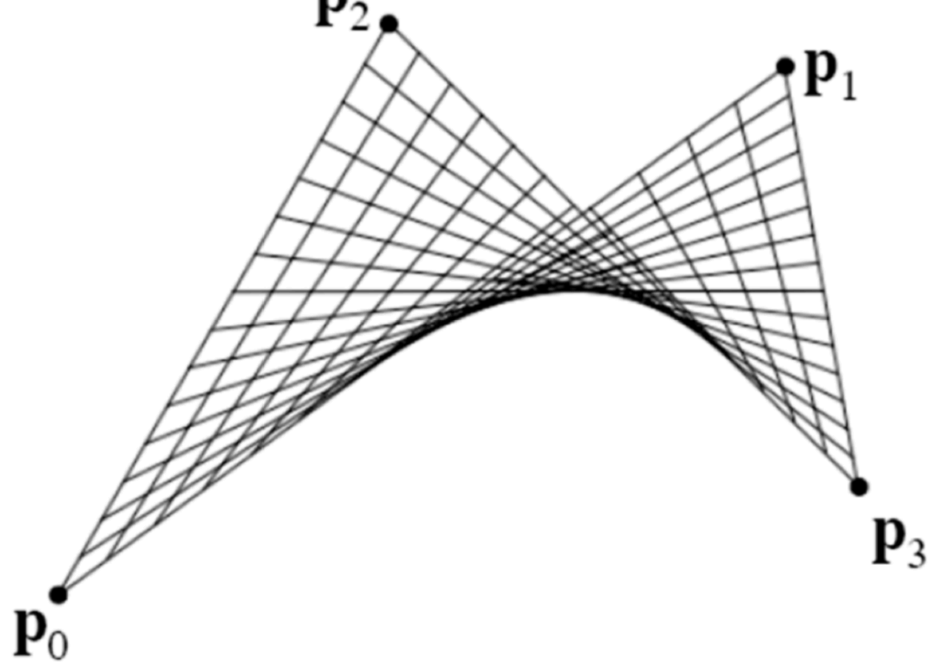
- List of vertices
- List of edges
- List of faces





Bilinear patches and interpolation

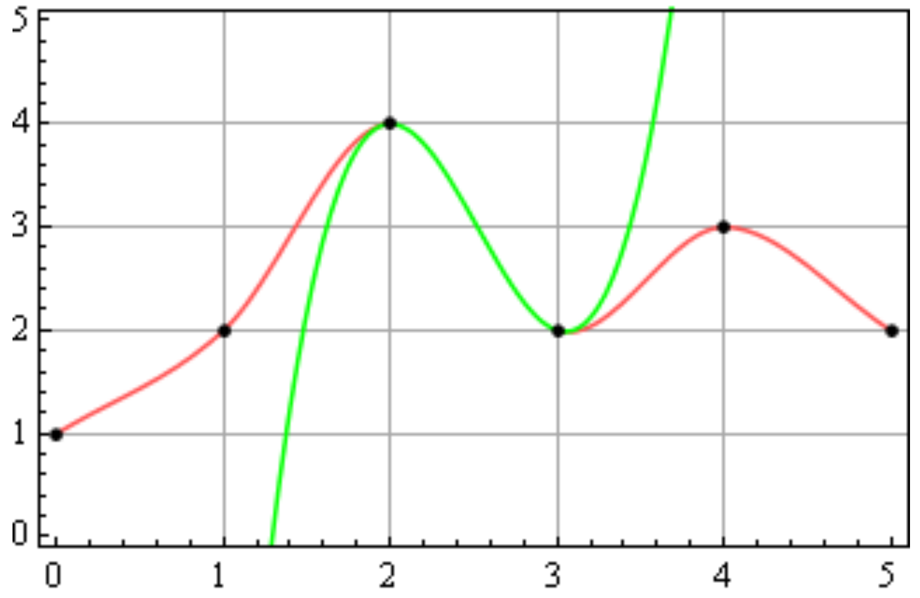
- Interpolation of four points
  - May not be co-planar
- Ruled surface – swept out by straight line
- Will develop equations in class



Cubic interpolation

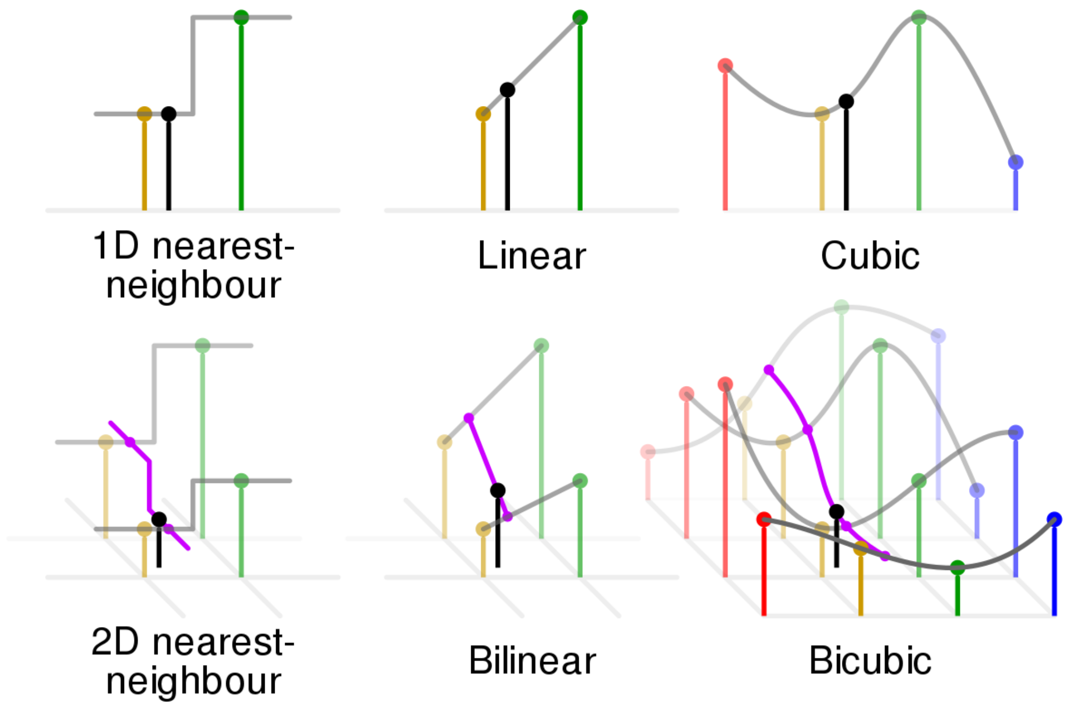
- $P(t) = ax^3 + bx^2 + cx + d$

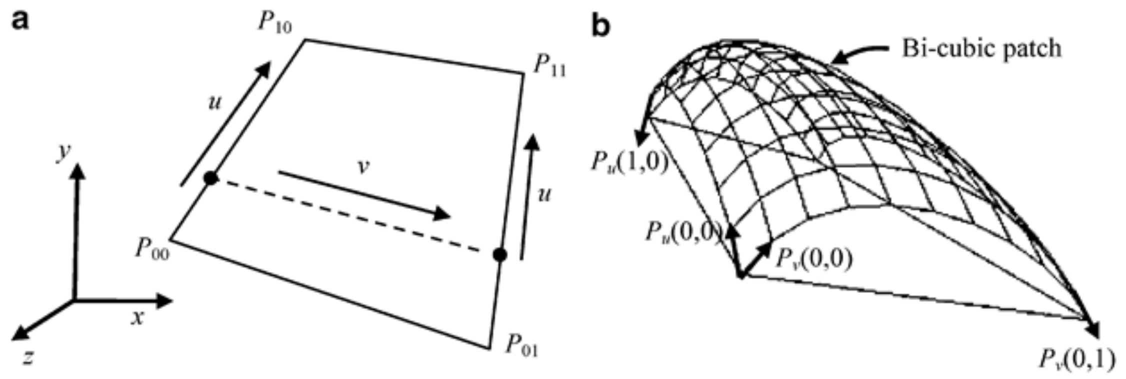
- Can match tangents at ends
- Good enough for human eye

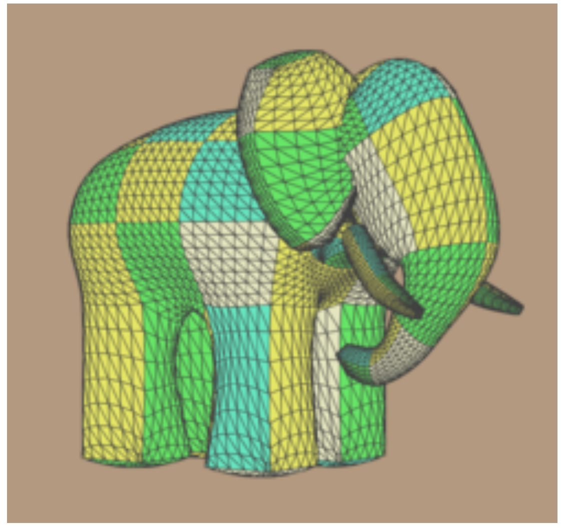


Bicubic surface patch

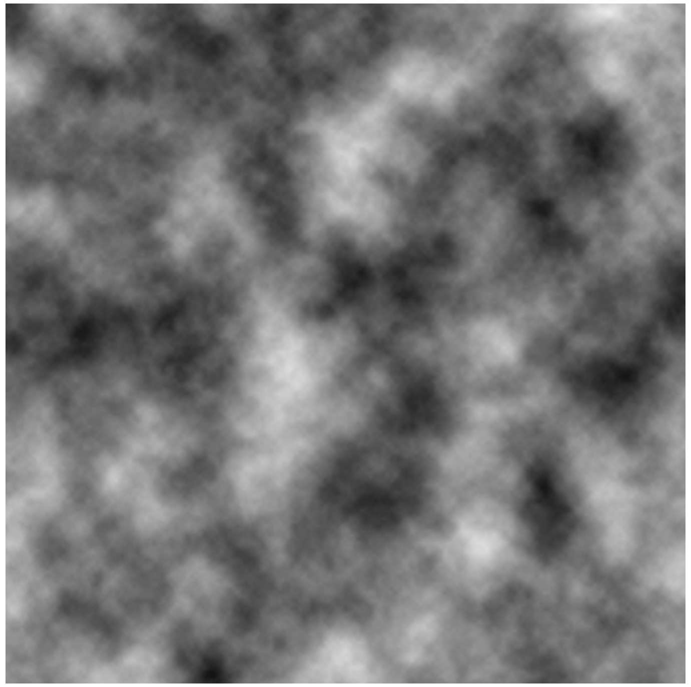
- Cubic curve in both directions

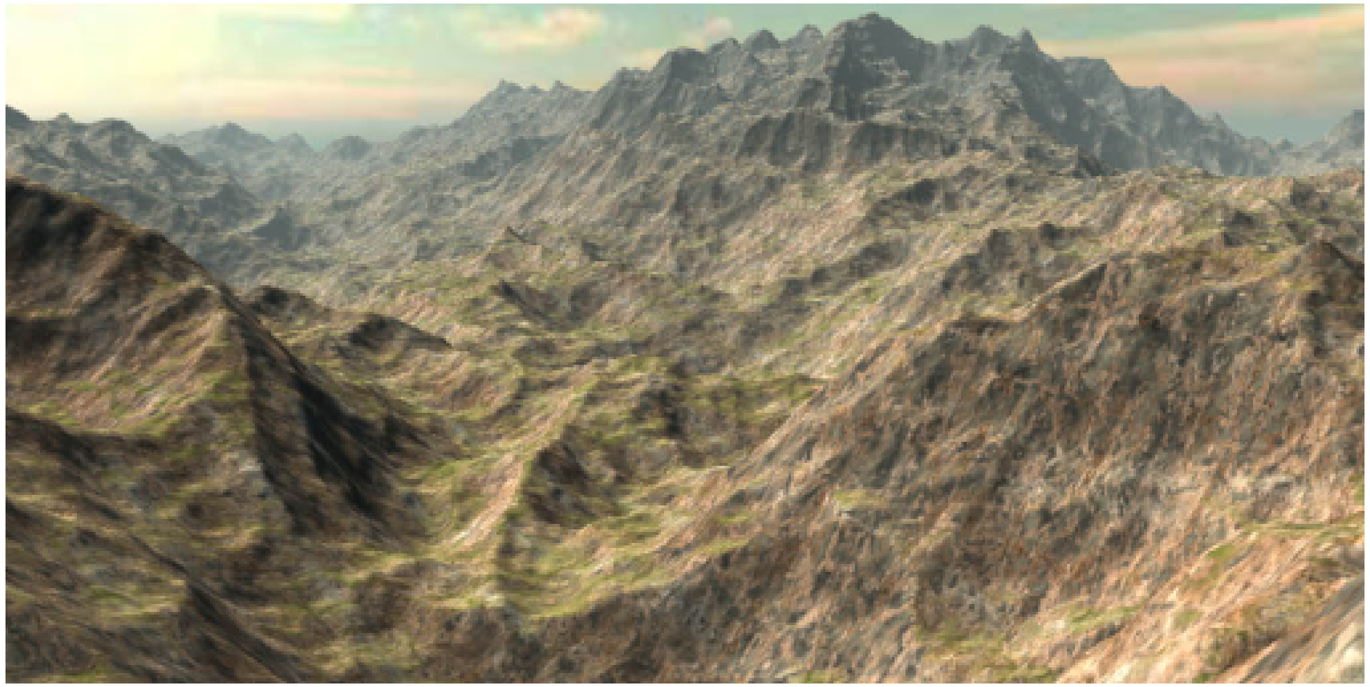




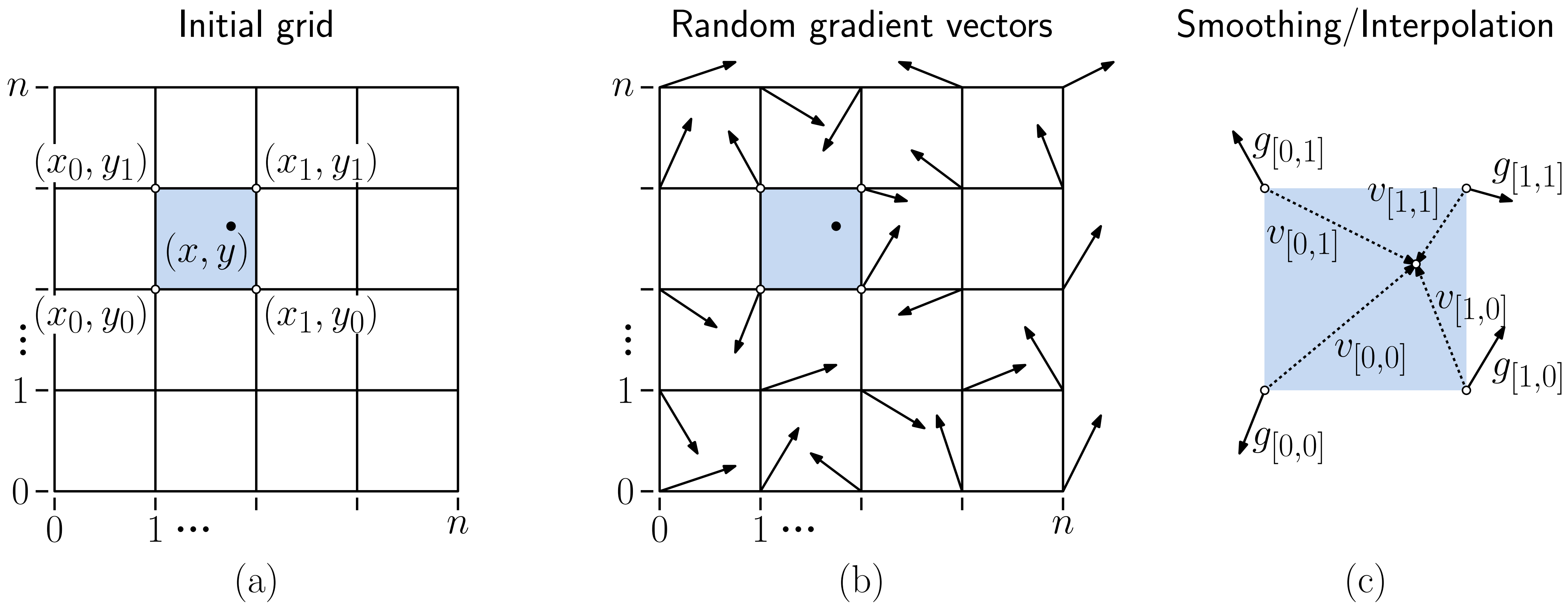


2D Perlin noise


(a)

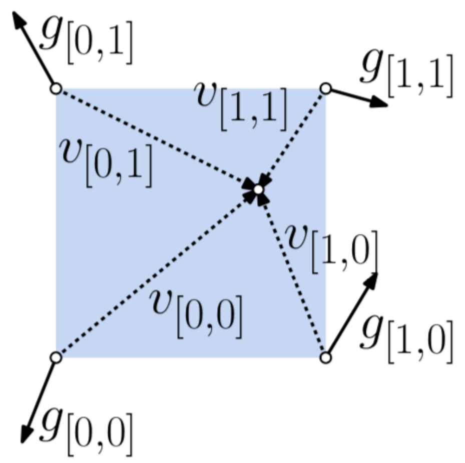

(b)

2D Perlin algorithm



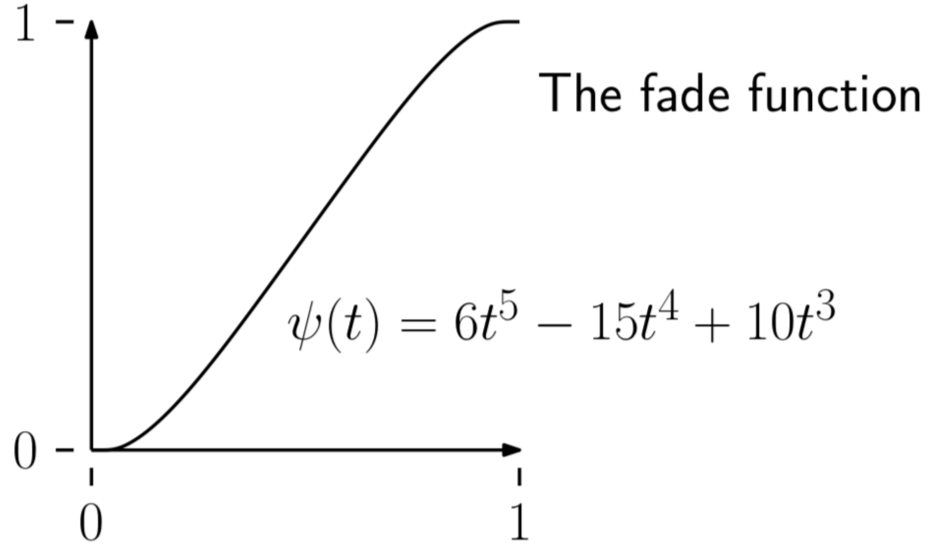
2D Perlin algorithm – gradient factor

$$\begin{aligned} \delta_{[0,0]} &= (v_{[0,0]} \cdot g_{[0,0]}) \quad \text{and} \quad & \delta_{[0,1]} &= (v_{[0,1]} \cdot g_{[0,1]}) \\ \delta_{[1,0]} &= (v_{[1,0]} \cdot g_{[1,0]}) \quad \text{and} \quad & \delta_{[1,1]} &= (v_{[1,1]} \cdot g_{[1,1]}). \end{aligned}$$



2D Perlin – fade function

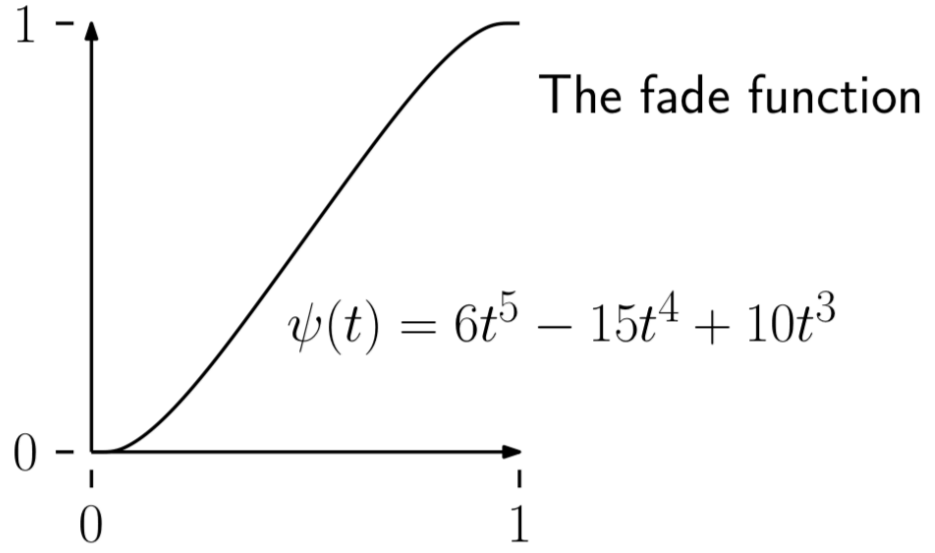
- $\psi(0) = 0$
- $\psi(1) = 1$
- $\psi'(0) = \psi'(1) = ??$



2D Perlin – fade function

- $\psi(0) = 0$
- $\psi(1) = 1$
- $\psi'(0) = \psi'(1) = ??$

- $\Psi(1) = \psi(s)\,\psi(t)$



2D Perlin – noise function

$$\text{noise}(x, y) \;=\; \Psi(1-x, 1-y)\delta_{[0,0]} + \Psi(x, 1-y)\delta_{[1,0]} + \Psi(1-x, y)\delta_{[0,1]} + \Psi(x, y)\delta_{[1,1]}$$

- Link: https://cpetry.github.io/TextureGenerator-Online/
- https://docs.unity3d.com/ScriptReference/Mathf.PerlinNoise.html

2D Perlin – noise function

$$\text{noise}(x,y) \;=\; \Psi(1-x,1-y)\delta_{[0,0]} + \Psi(x,1-y)\delta_{[1,0]} + \Psi(1-x,y)\delta_{[0,1]} + \Psi(x,y)\delta_{[1,1]}$$

$$\text{perlin}(x,y) \;=\; \sum_{i=0}^{k} p^i \cdot \text{noise}(2^i \cdot x, 2^i \cdot y)$$

- P is what parameter?

Crowd motion

- How do multiple agents move without colliding?
- Detect collisions in advance

Crowd motion

- How do multiple agents move without colliding?
- Detect collisions in advance

- Dynamic obstacles – anticipate where someone will be

Crowd motion

- Model each agent with

- Current position $P_i(0)$
- Current velocity $\vec{v}_i(0)$
- Target velocity $\vec{v}_i{}^0(0)$
  - Towards goal

- Forces $\vec{F}_i(0)$ push on agent

Possible forces

- Like boid flocking?

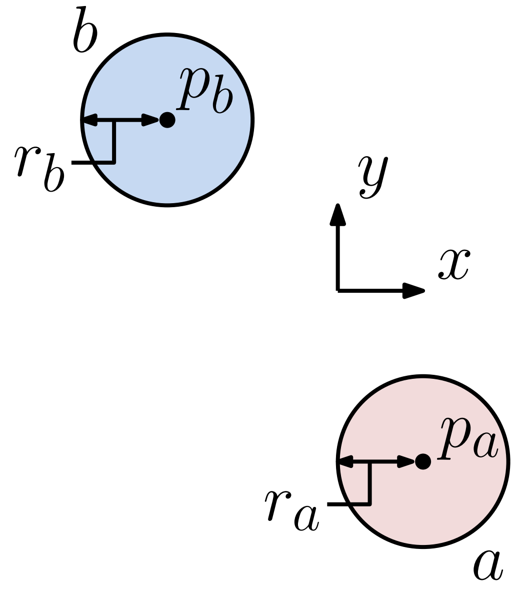
Possible forces

- Like boid flocking?

- Separation
- Obstacle Avoidance
- Attraction
- Traffic signals and social conventions
- Individual variations

Velocity obstacles

- Compute forbidden velocities
  - That would lead to collision
- Example
  - Agent **a** walking towards obstacle **b**
  - What velocities at time i cause collision?



Velocity obstacles

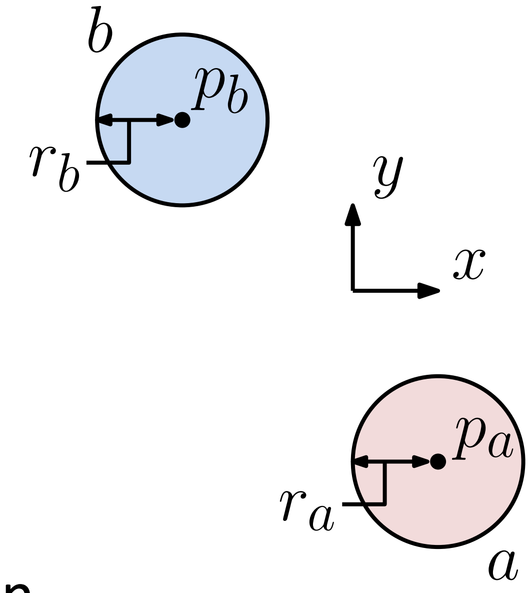
- Compute forbidden velocities
    - That would lead to collision

- Example
    - Agent **a** walking towards obstacle **b**

    - What velocities at time i cause collision?

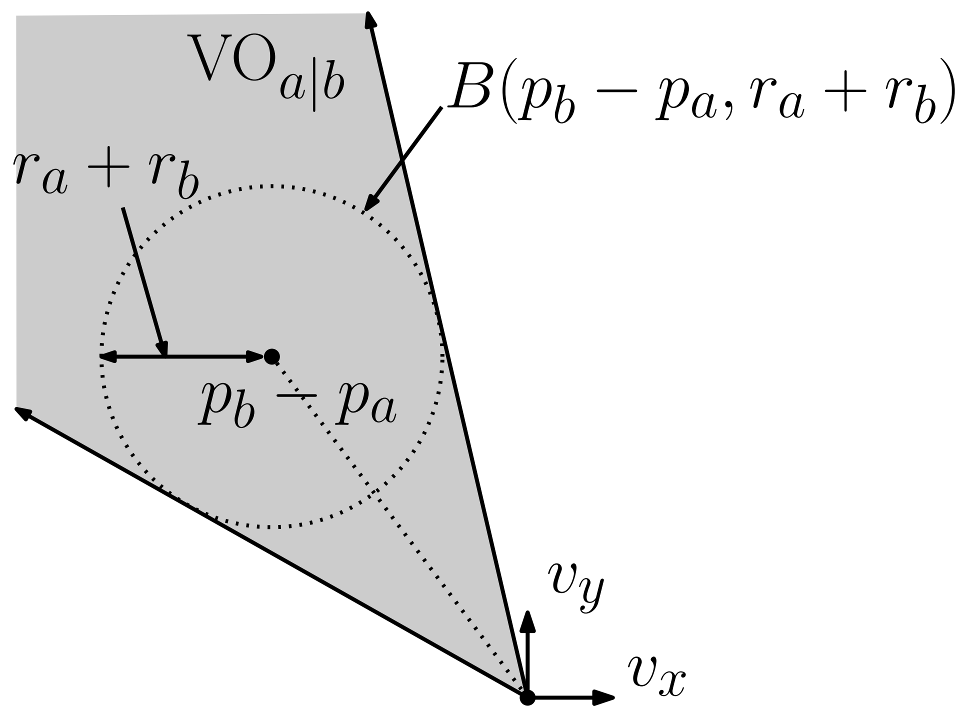
    - Velocity v = (pb-pa) causes immediate collision

    - Others?



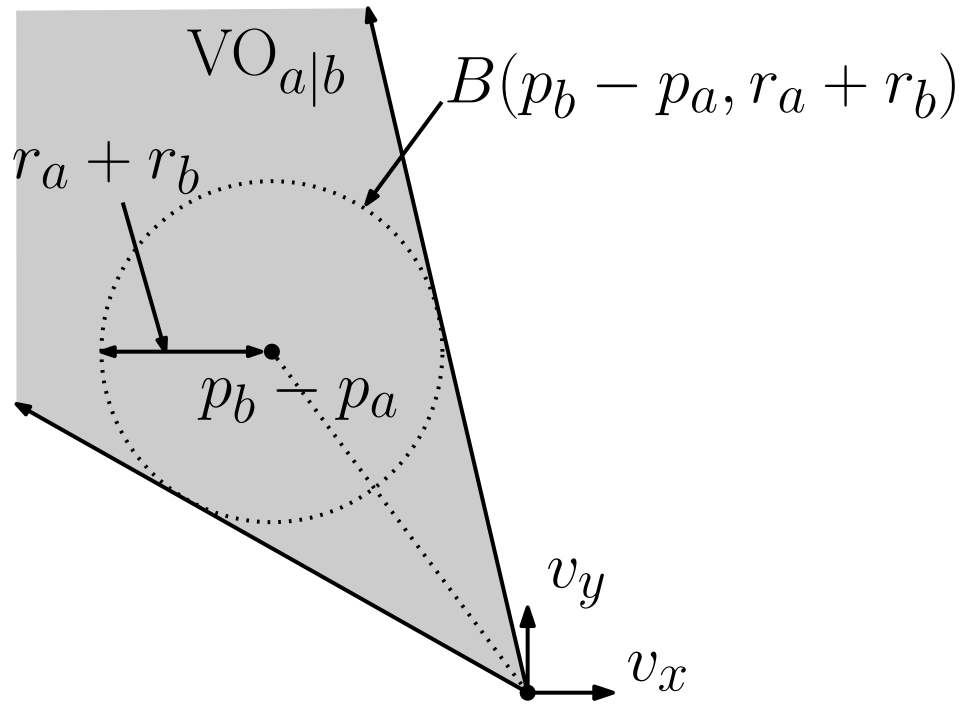
Velocity obstacles

- Region V0$_{(a|b)}$ of forbidden velocities
- Cone around Ball B(pb-pa, ra+rb)



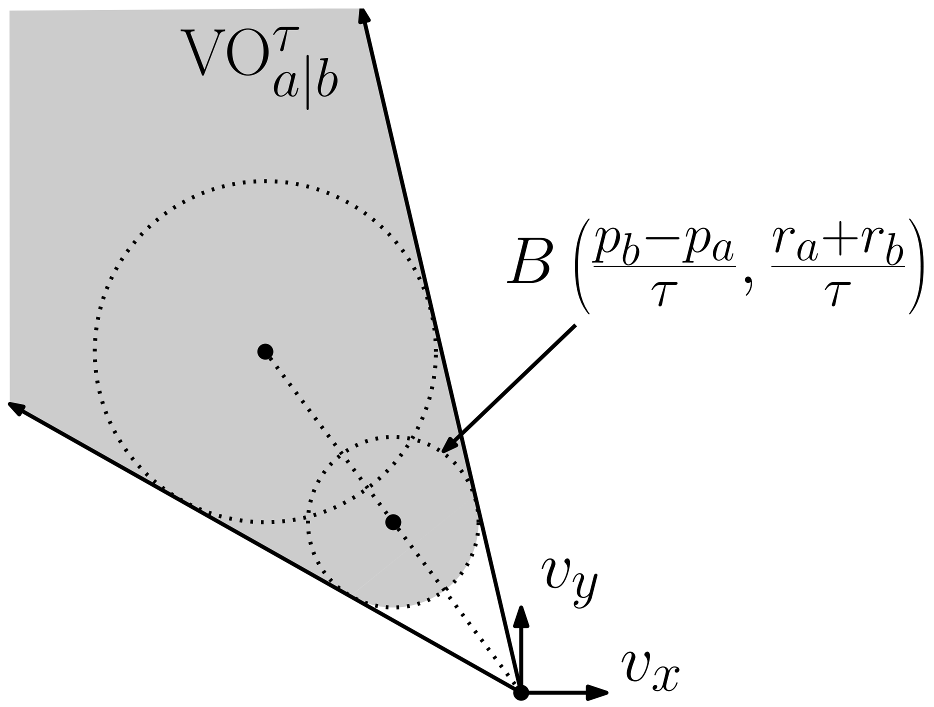
Velocity obstacles

- Region $V0_{(a|b)}$ of forbidden velocities
- Cone around Ball B(pb-pa, ra+rb)
- Apex of cone is what?

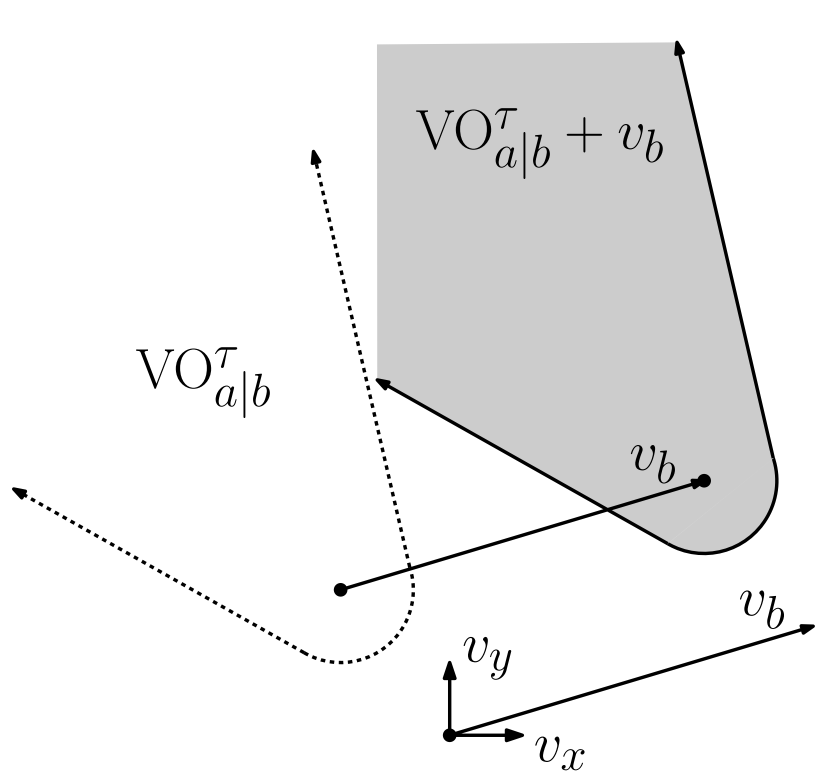


Velocity obstacles

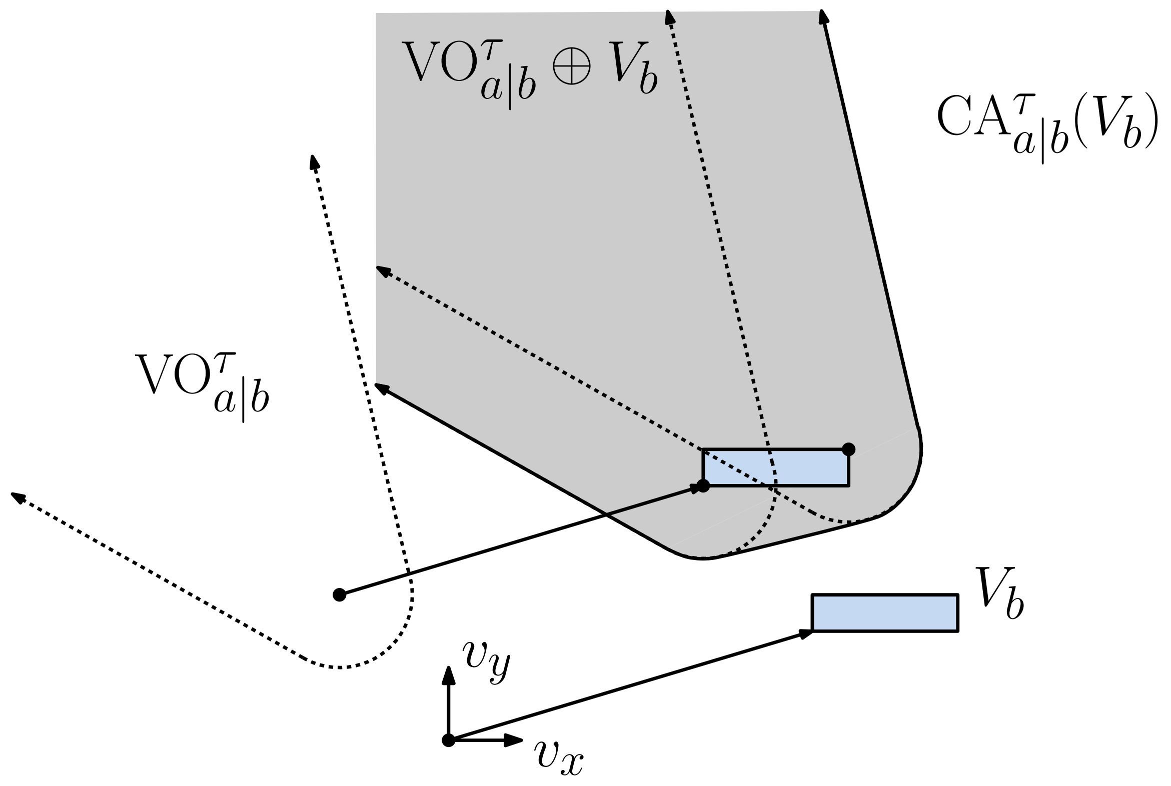
- Region V0$_{(a|b)}$ of forbidden velocities
- Apex of cone is very slow velocities
- Limit length of future time to (0,tau)
- Limiting time truncates cone – why?


$\mathrm{VO}^{\tau}_{a|b}$
$B\left(\frac{p_b-p_a}{\tau}, \frac{r_a+r_b}{\tau}\right)$
v_y
v_x

Obstacle **b** moving?



(a)

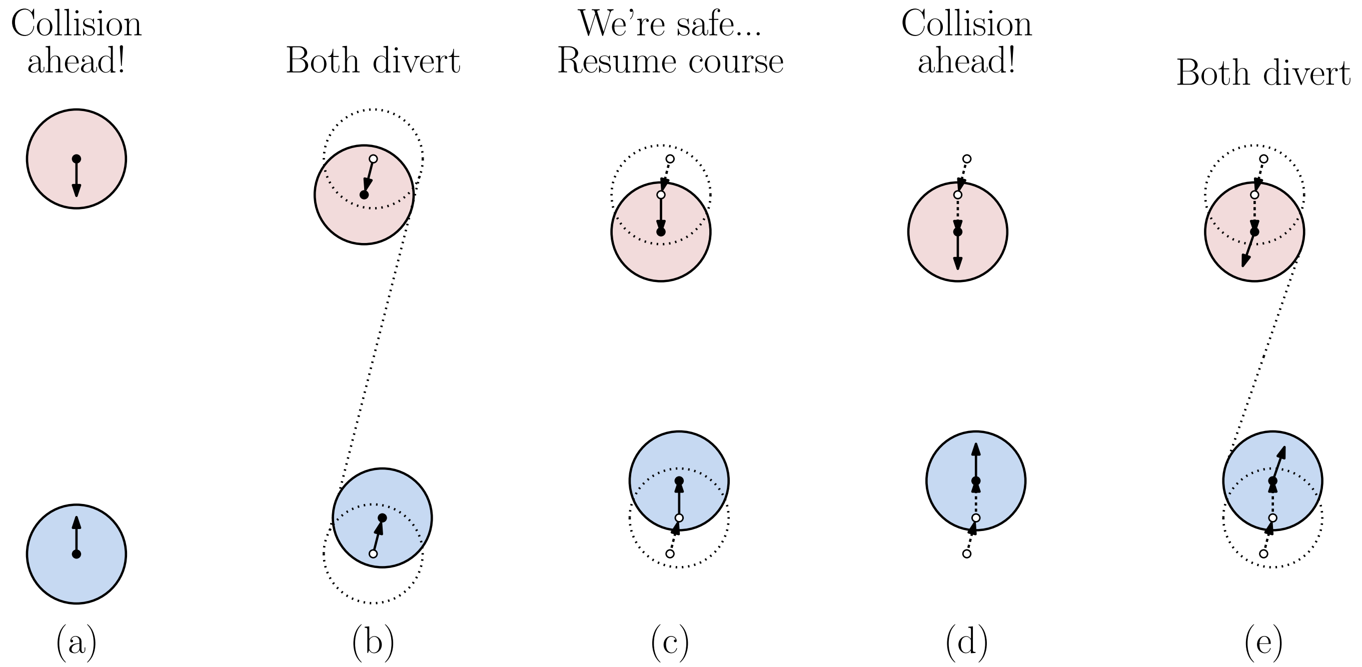

$\mathrm{VO}^{\tau}_{a|b} \oplus V_b$
$\mathrm{CA}^{\tau}_{a|b}(V_b)$
$\mathrm{VO}^{\tau}_{a|b}$
V_b
v_y
v_x

(b)

Who is responsible for avoiding collision?

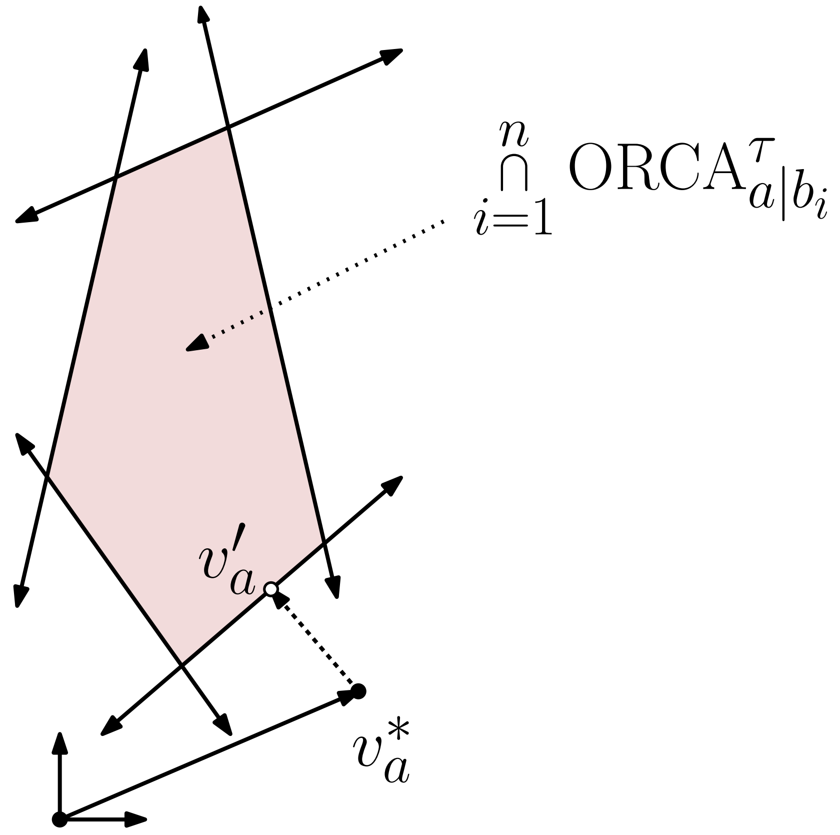
- Both agents fully responsible
- Oscillating motion

- Other avoids
- Your path becomes clear
- You resume original path
- Collision!



Avoiding multiple agents

- Simplify each agent's forbidden velocity region to half plane
- Intersect acceptable velocity regions to get polygon
- Take velocity v'a nearest to target velocity v*a



Lin and Manocha

- https://www.youtube.com/watch?v=lyyyEcy_9so

- https://www.youtube.com/watch?v=xme4pRelwJ0