CS 277 - Experimental Haptics
Lecture 12

Haptic Illusions

A list of illusions that we will cover

- Rendering 3D shapes using 2 DOFs
  - i.e. how to project positions and forces on smaller rank vectorials spaces
- Rendering 2D shapes using 1 DOF
  - i.e. how work can be your ally (and your enemy)
- Rendering small bumps to feel **large**
  - i.e. how our sensitivity to force direction is not that good
- Rendering **large** virtual environments using small devices
  - i.e. how to take advantage of humans' poor perception of position
- Rendering fast cars without moving much
  - i.e. our vestibular sense is also pretty limited

2-DOF Haptics

DEMO

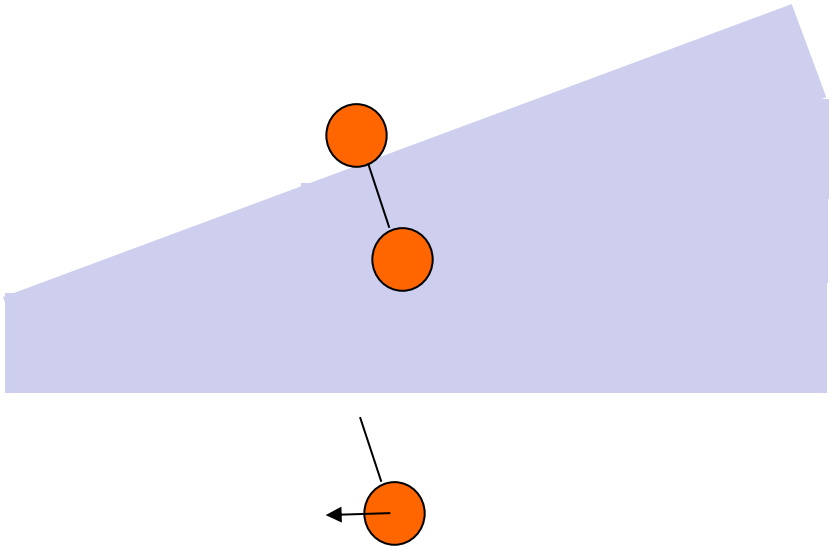
Rendering 3D shapes using 2 DOFs

Why would you want to do this?

- Assume a 2 sensor 2 actuator device
- Try to render a 3D world through it

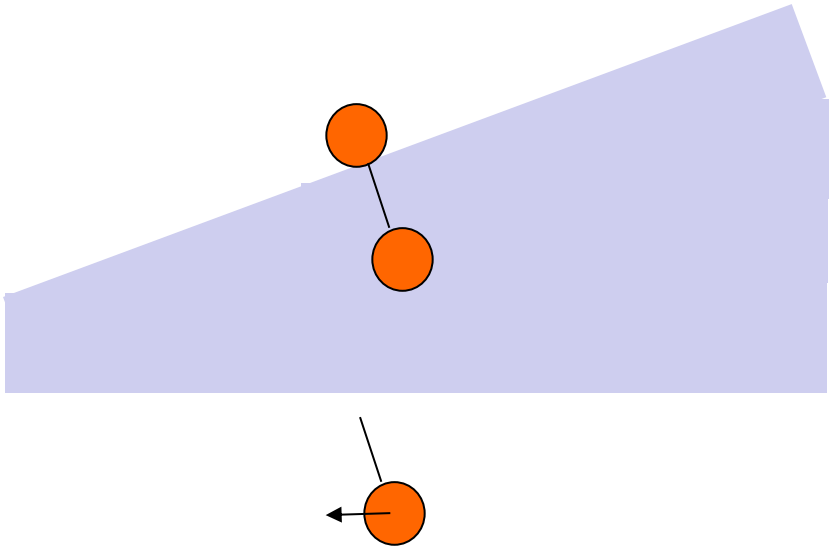
Rendering 3D shapes using 2 DOFs

- Very similar to what we just described for textures
- Basically, assuming interaction with surface z = g(x, y)
  - 3DOF device: F = -r(p) n
  - 2DOF device: $F_{lateral}$ = -r(p) grad(g)



Rendering 3D shapes using 2 DOFs

- Assuming constant penetration > 0
  - you can feel a lateral force proportional to the steepness of you plane
  - as you move your device across the surface
- In other words, we're projecting the force on the display of the device



A list of illusions that we will cover

- Rendering 3D shapes using 2 DOFs
  - i.e. how to project positions and forces on smaller rank vectorials spaces
- Rendering 2D shapes using 1 DOF
  - i.e. how work can be your ally (and your enemy)
- Rendering small bumps to feel **large**
  - i.e. how our sensitivity to force direction is not that good
- Rendering **large** virtual environments using small devices
  - i.e. how to take advantage of humans' poor perception of position
- Rendering fast cars without moving much
  - i.e. our vestibular sense is also pretty limited

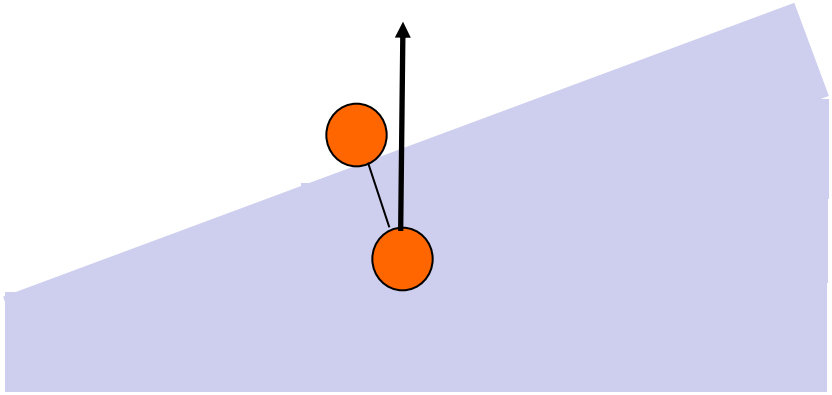
1-DOF Haptics

DEMO

Rendering 2D shapes using 1 DOF

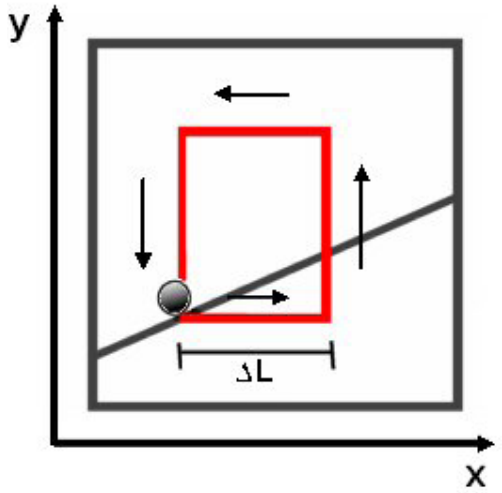
Why would you want to do this?

- Assume a 2 sensor 1 actuator device (asymmetric device)
- Try to render a 3D world through it



Rendering 2D shapes using 1 DOF

- When moving to the left you load a spring for free
- Good news: this extra work gives you the impression of touching a 2D object
- Bad news: surface will feel extra active

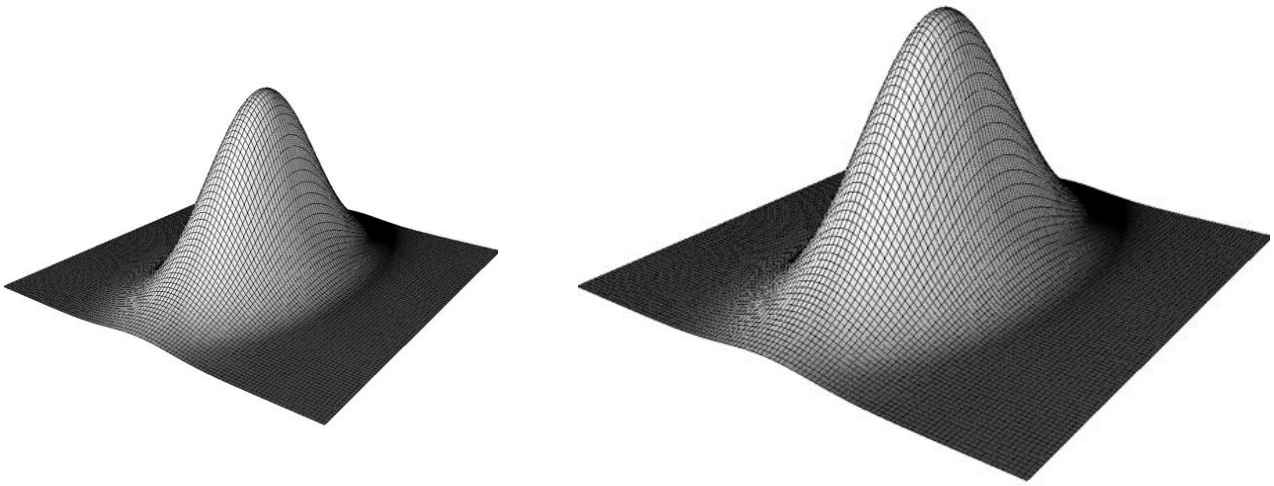


A list of illusions that we will cover

- Rendering 3D shapes using 2 DOFs
  - i.e. how to project positions and forces on smaller rank vectorials spaces
- Rendering 2D shapes using 1 DOF
  - i.e. how work can be your ally (and your enemy)
- Rendering small bumps to feel **large**
  - i.e. how our sensitivity to force direction is not that good
- Rendering **large** virtual environments using small devices
  - i.e. how to take advantage of humans' poor perception of position
- Rendering fast cars without moving much
  - i.e. our vestibular sense is also pretty limited

Rendering small Bumps to Feel Large

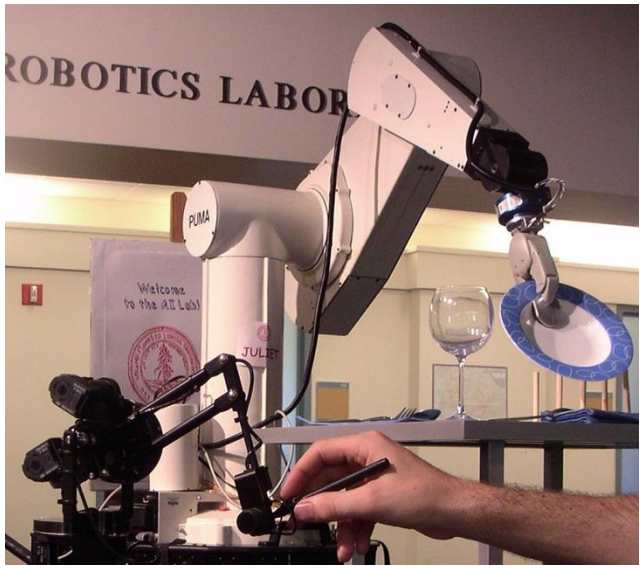
- our force direction perception
  - is poor
  - can be greatly influenced by visual feedback
- You can take advantage of this and make smaller bumps feel larger by “cheating” visually



A list of illusions that we will cover

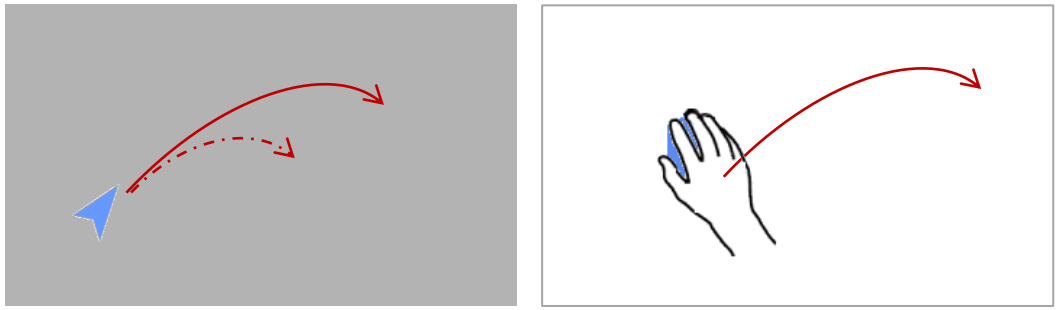
- Rendering 3D shapes using 2 DOFs
  - i.e. how to project positions and forces on smaller rank vectorials spaces
- Rendering 2D shapes using 1 DOF
  - i.e. how work can be your ally (and your enemy)
- Rendering small bumps to feel **large**
  - i.e. how our sensitivity to force direction is not that good
- Rendering **large** virtual environments using small devices
  - i.e. how to take advantage of humans' poor perception of position
- Rendering fast cars without moving much
  - i.e. our vestibular sense is also pretty limited

Exploring Large Workspaces

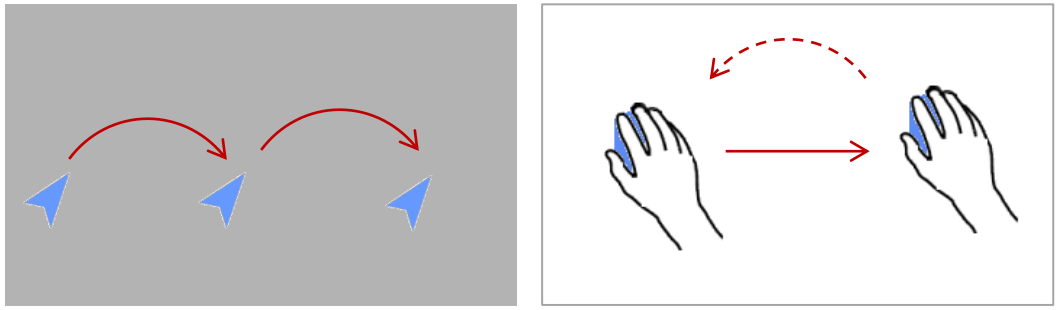


Control Paradigms

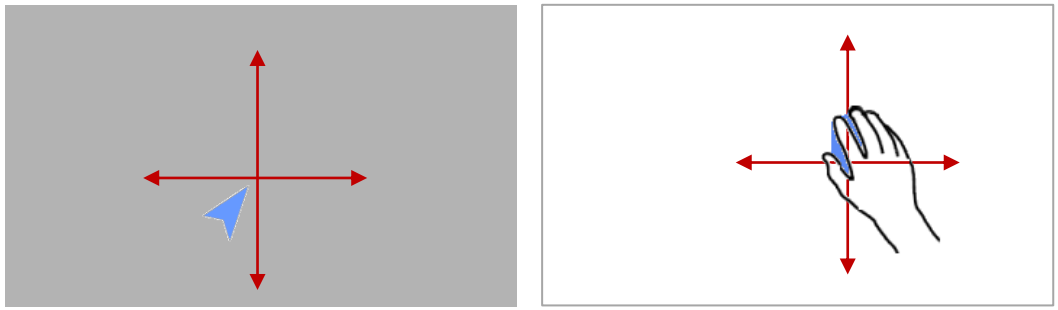
Position Scaling



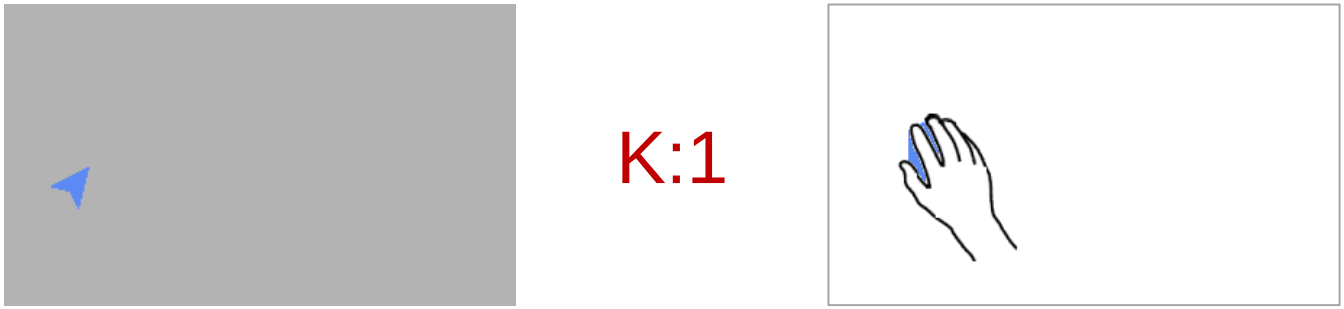
Position Indexing



Rate Control

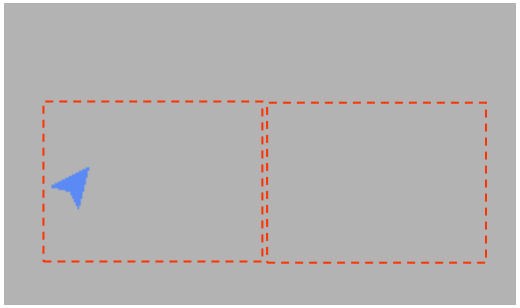


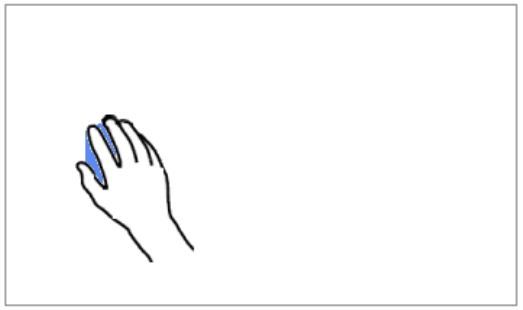
Position and Scaling



- Most common control paradigm.
- Bijective mapping between physical and virtual workspace.
- Loss of spatial resolution when high scaling factors are used.

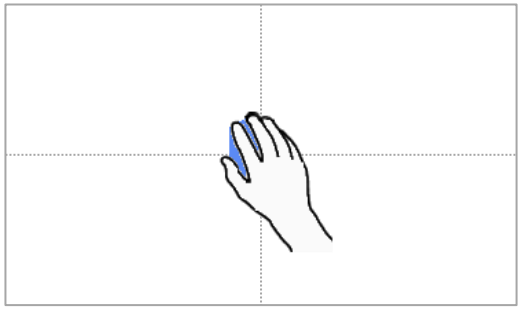
Indexing





- Manual shifting of workspace through clutching or lifting of device.
- Additional user button required.
- Not optimal for small devices.

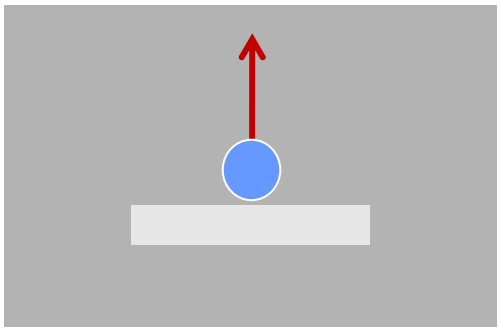
Rate Control

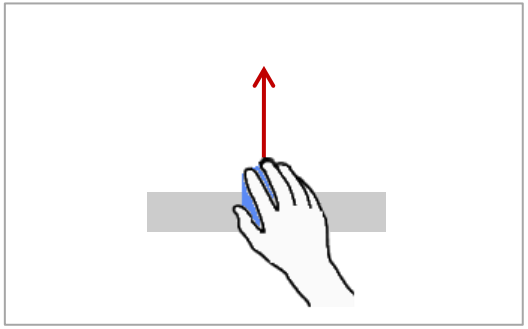


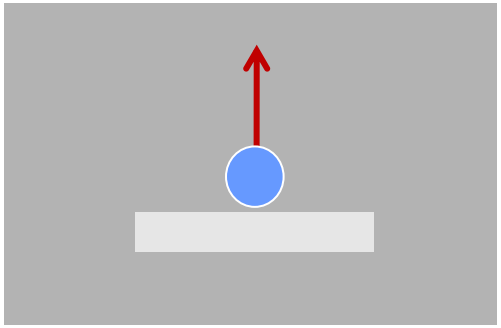
- Position command is translated into a desired velocity of the cursor.
- Low bandwidth.
- Access to unlimited workspace

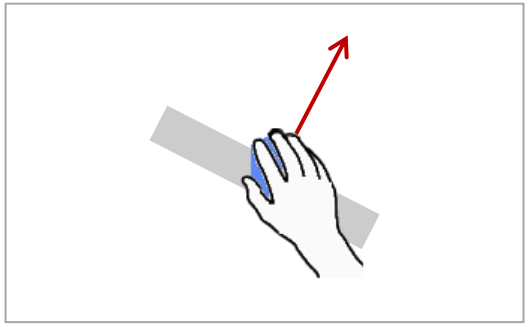
Haptic Perception

Discrimination of Force Direction



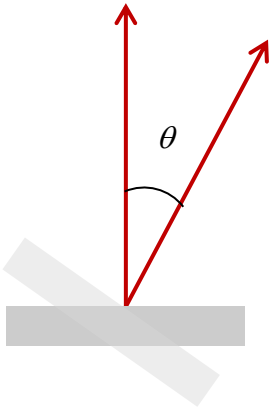






visual perception

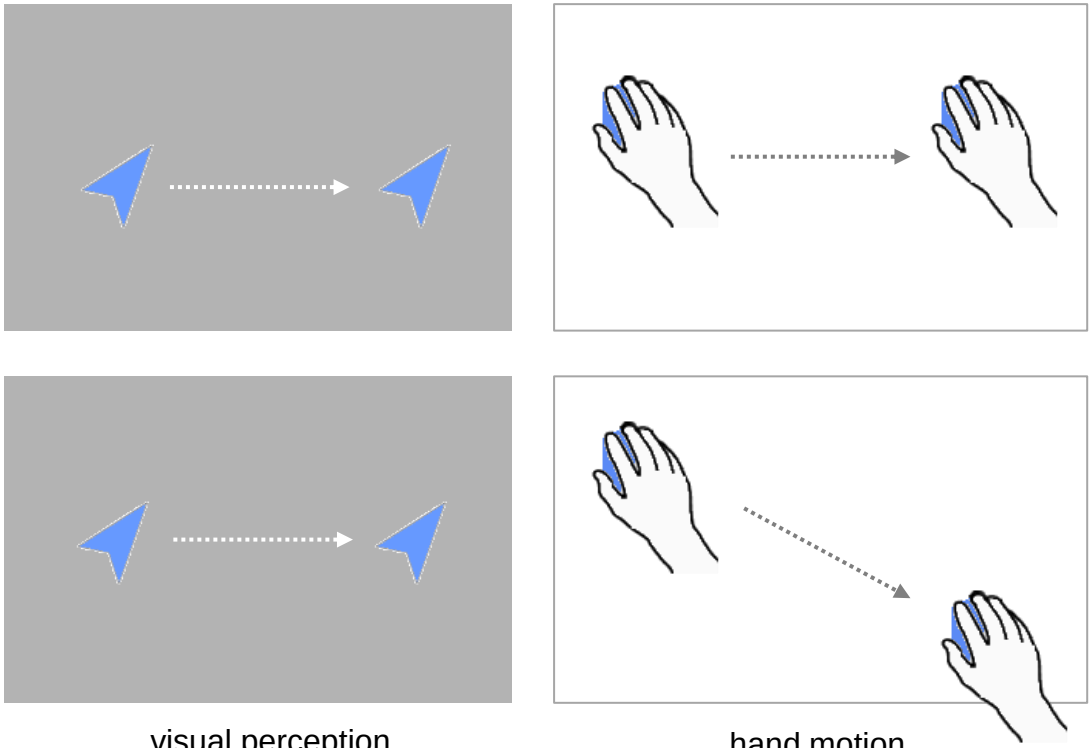
haptic perception



discrimination angle

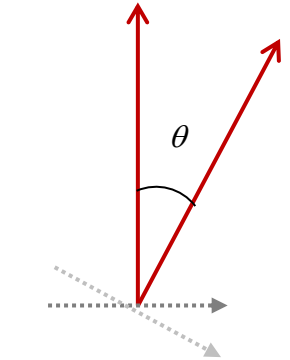
Haptic Perception

Discrimination of Hand Motion Direction in Free Space



visual perception

hand motion

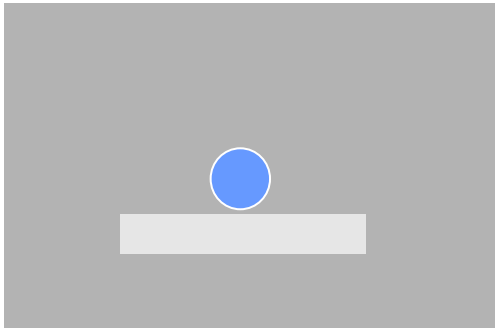


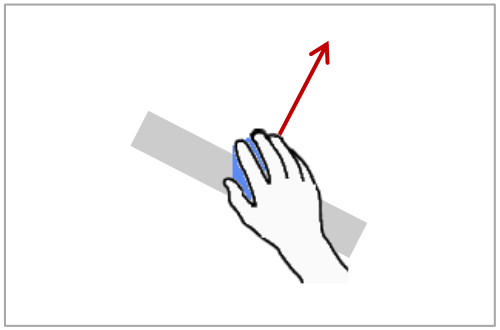
discrimination angle

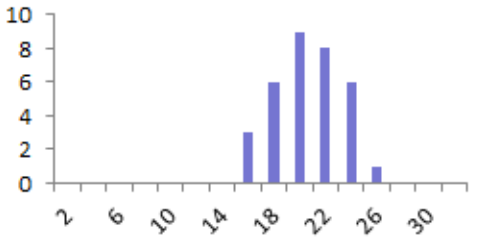
Haptic Perception

Experimental Results

Contact Forces

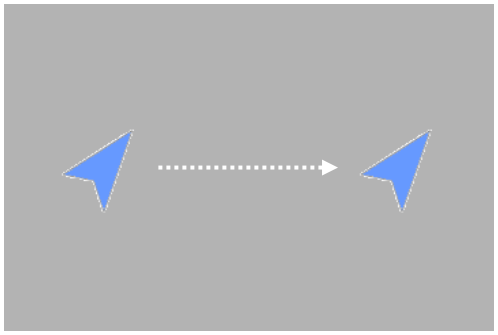


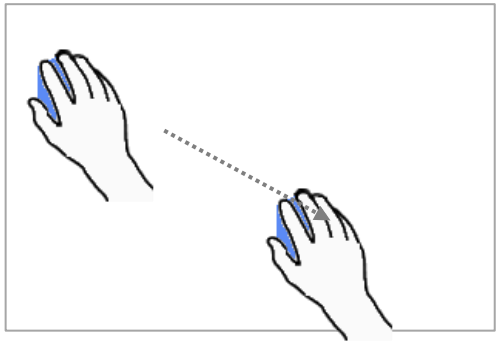


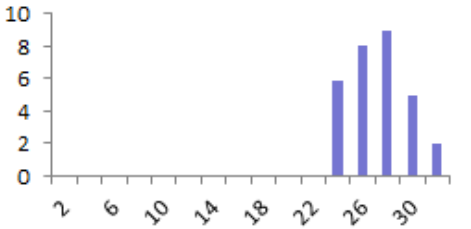


Median: 19 deg (C.I 95% [17, 22])

Free Space







Median: 27 deg (C.I 95% [25, 30])

visual perception

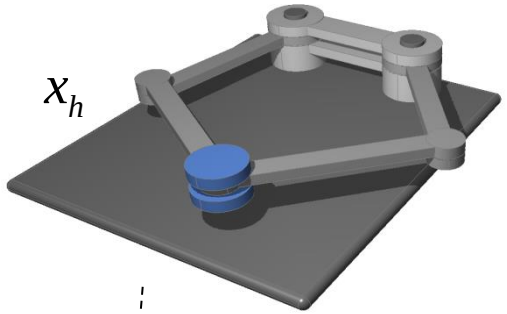
haptics perception

discrimination angles

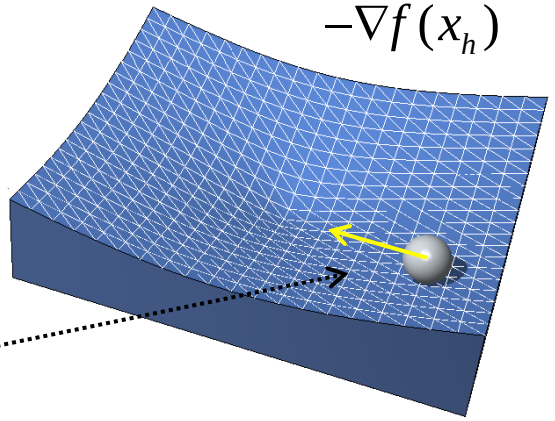
Workspace Expansion

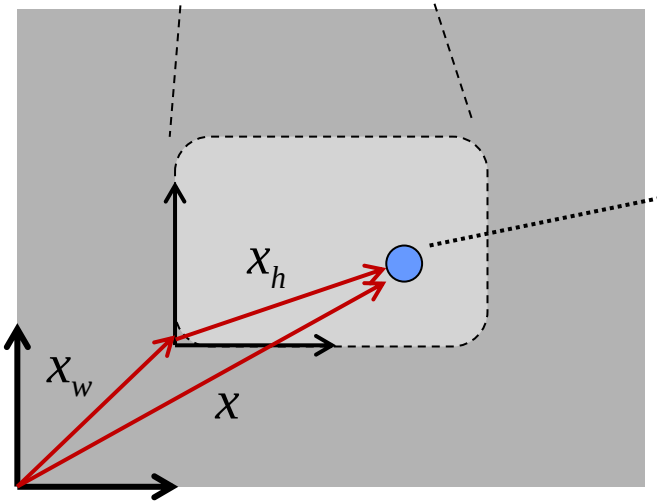
Workspace Mapping

Haptic Interface



Haptic Device Criterion Function: $f(x_h)$

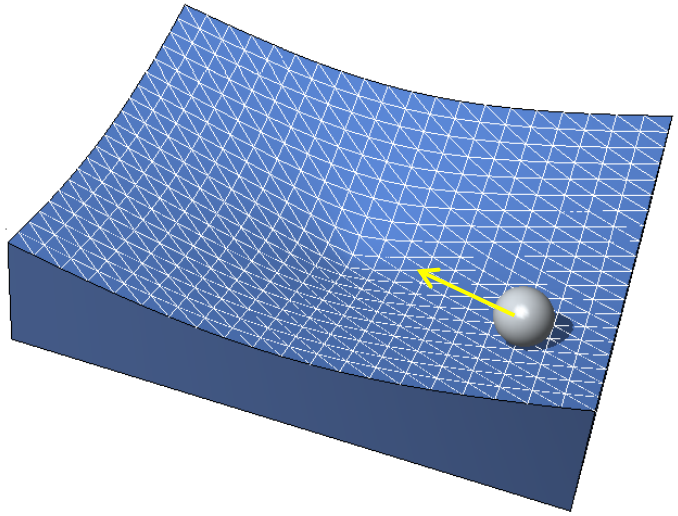


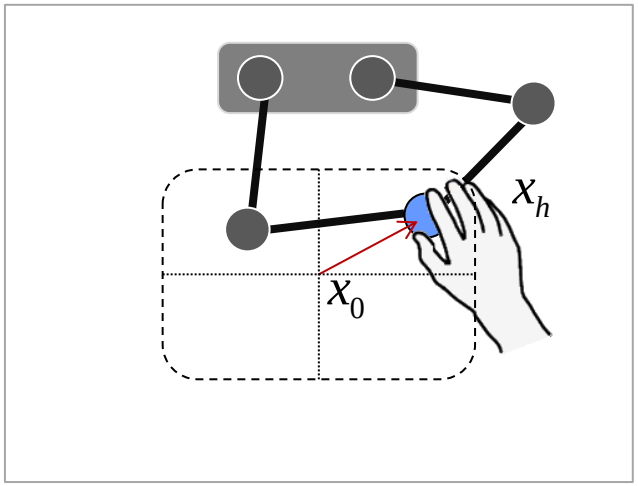


Simulation Environment

Workspace Expansion

Device Dependent Haptic Criteria





Normalized Distance from center of workspace:

$$f(x_h) = \frac{1}{2}\frac{(x_h - x_0)^2}{r_w}$$

$$-\nabla f(x_h) = -\frac{x_h - x_0}{r_w}$$

r_w : workspace radius

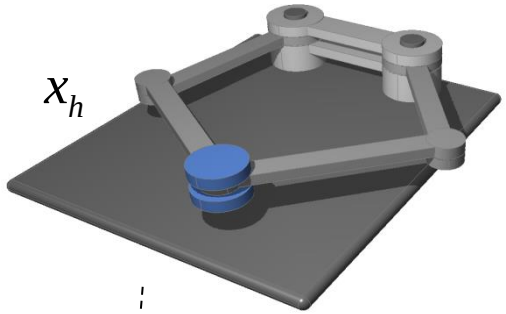
x_h : current position of haptic device

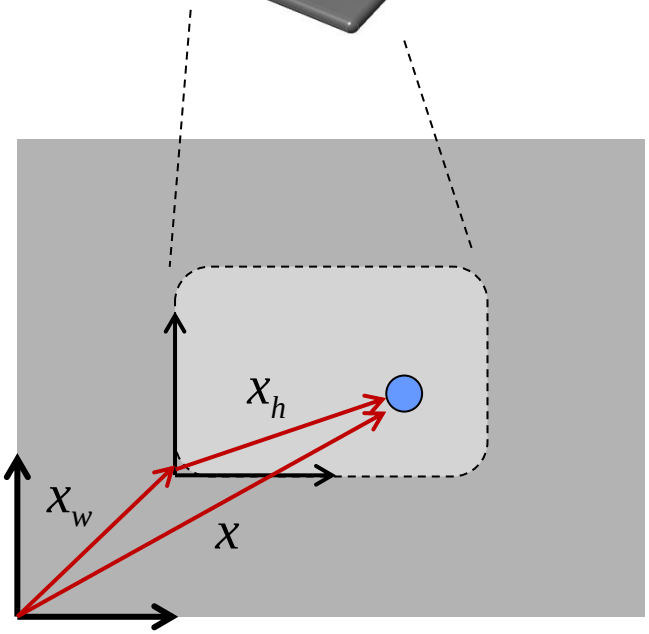
x_0 : position of origin

Workspace Expansion

Drifting the Workspace

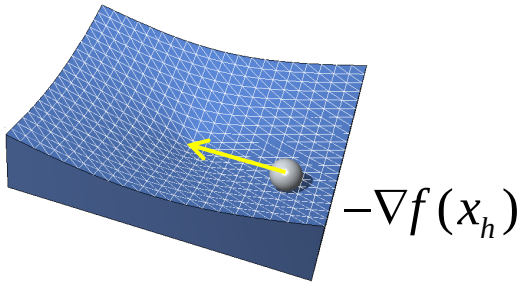
Haptic Interface





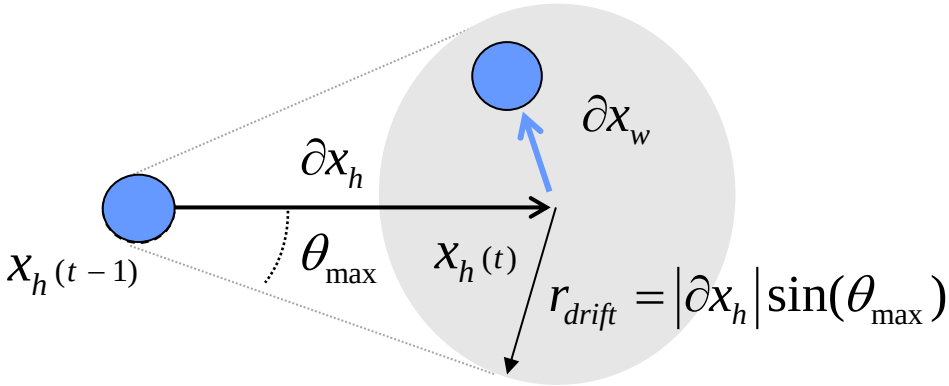
Simulation Environment

Criterion Function



Workspace Drift

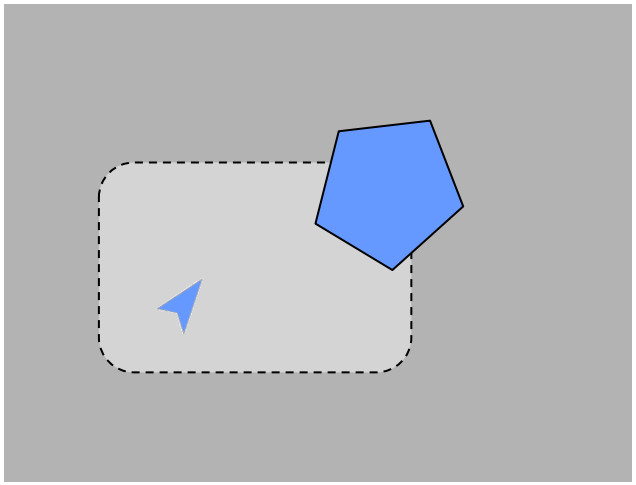
$$\partial x_w = r_{drift} \nabla f(x_h) \quad = \sin(\theta_p) \left| \partial x_h \right| \nabla f(x_h)$$

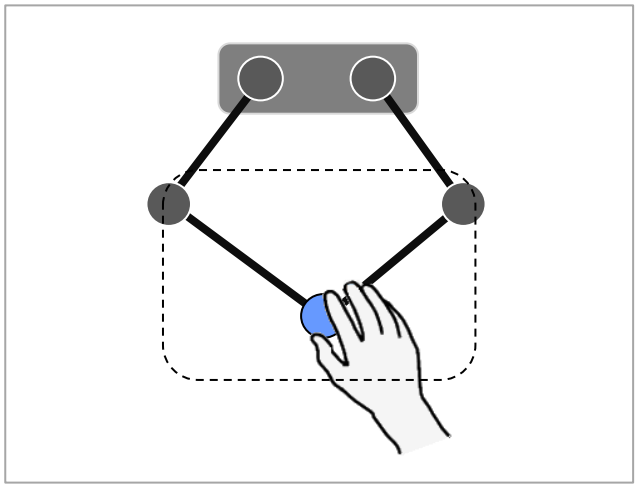


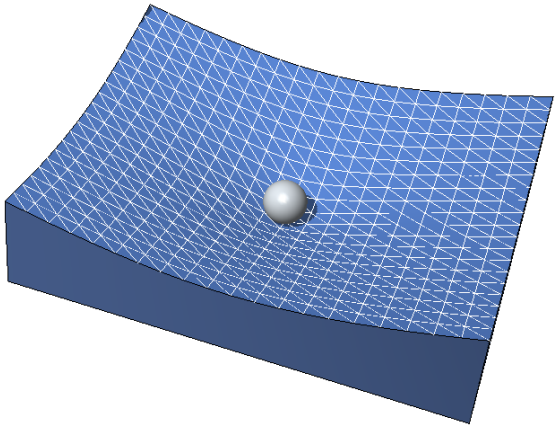
Perception Cone

Workspace Expansion

Drifting the Workspace

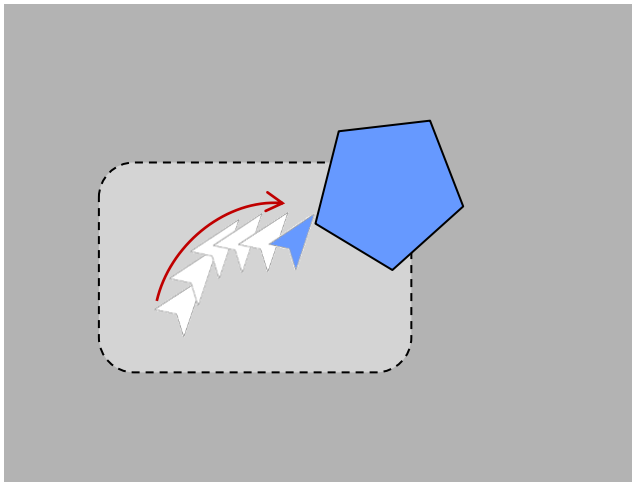


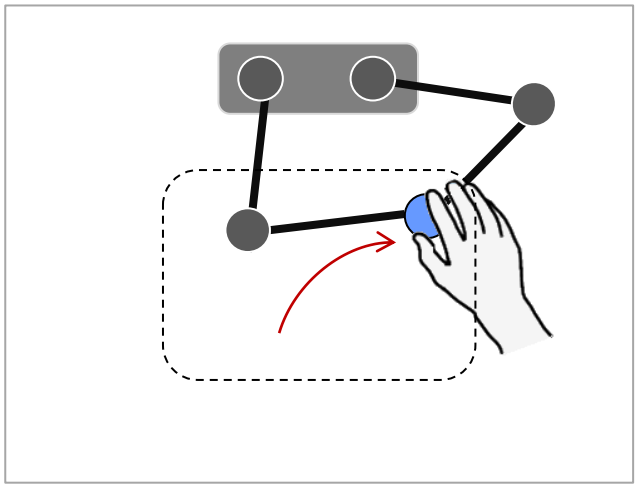


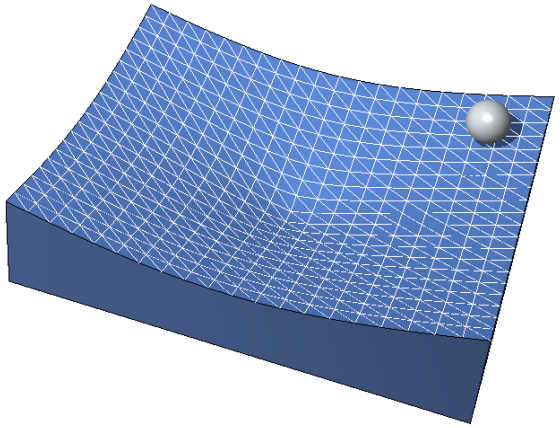


Workspace Expansion

Drifting the Workspace

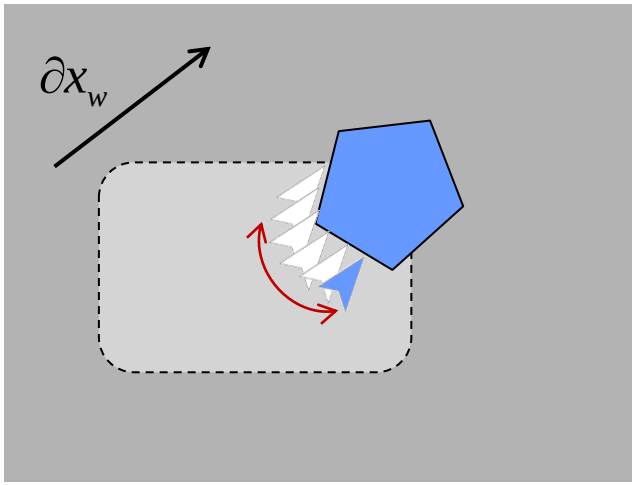


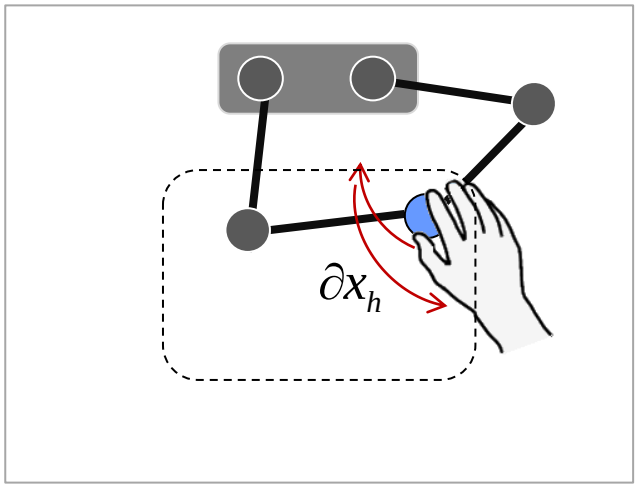




Workspace Expansion

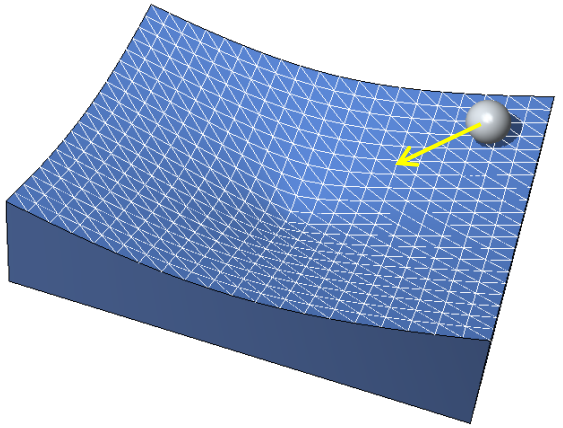
Drifting the Workspace





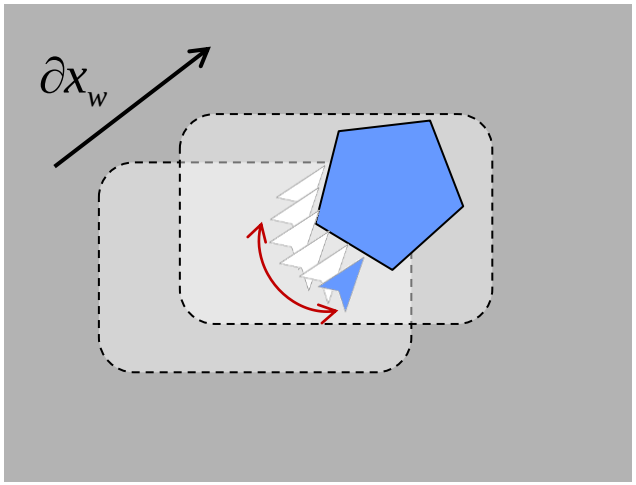
Workspace Drift:

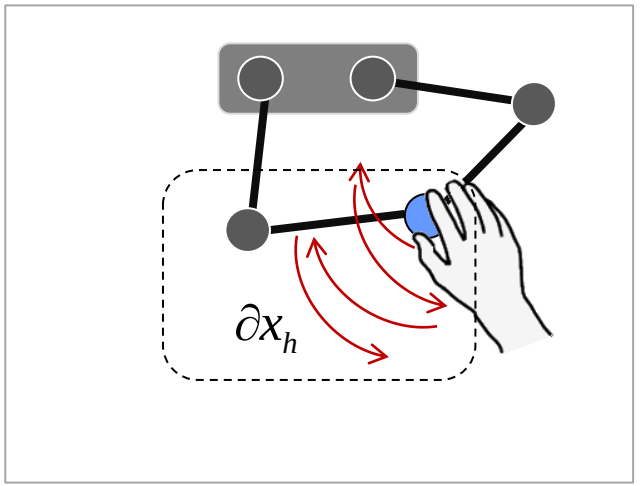
$$\partial x_w = \sin(\theta_p)\left|\partial x_h\right|\nabla f(x_h)$$



Workspace Expansion

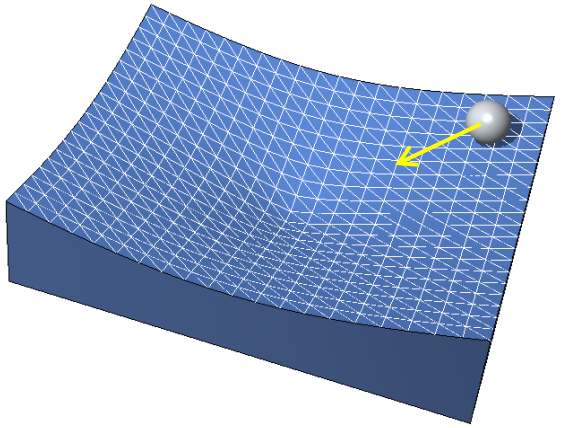
Drifting the Workspace





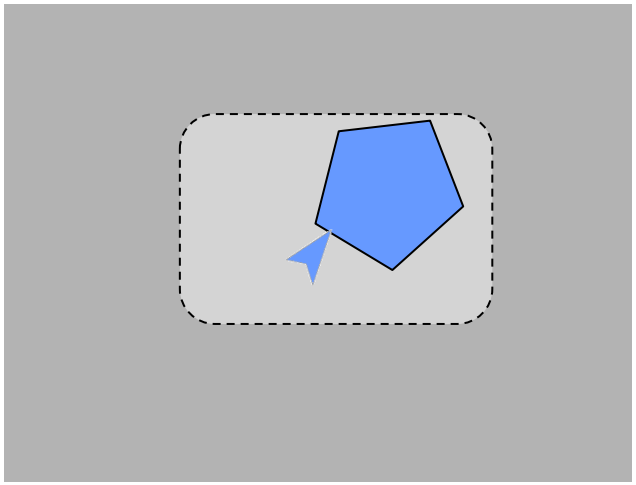
Workspace Drift:

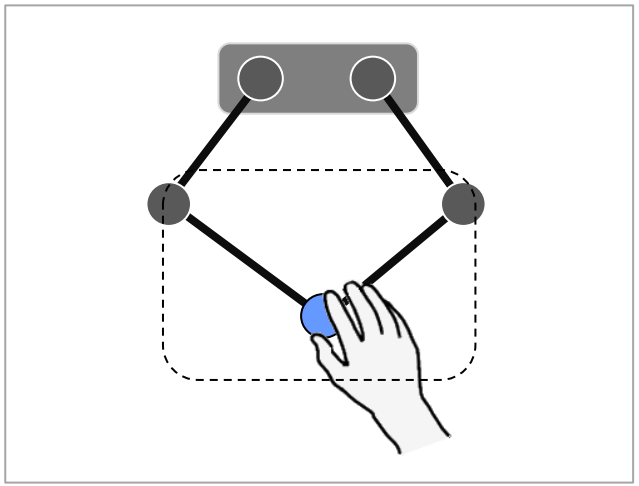
$$\partial x_w = \sin(\theta_p)\left|\partial x_h\right|\nabla f(x_h)$$

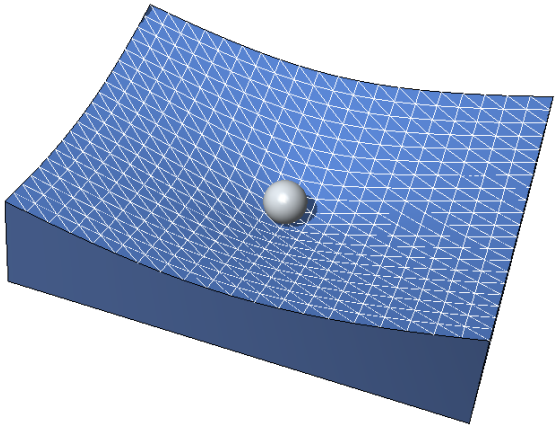


Workspace Expansion

Drifting the Workspace





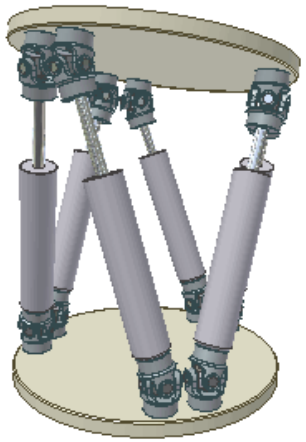


A list of illusions that we will cover

- Rendering 3D shapes using 2 DOFs
  - i.e. how to project positions and forces on smaller rank vectorials spaces
- Rendering 2D shapes using 1 DOF
  - i.e. how work can be your ally (and your enemy)
- Rendering small bumps to feel **large**
  - i.e. how our sensitivity to force direction is not that good
- Rendering **large** virtual environments using small devices
  - i.e. how to take advantage of humans' poor perception of position
- Rendering fast cars without moving much
  - i.e. our vestibular sense is also pretty limited

Simulators

Rendering fast vehicles without moving much



Simulators

Rendering fast vehicles without moving much

Basic issue:

- Platform (left) has limited workspace, plane (right) doesn't

Simulators

Rendering fast vehicles without moving much

Washout filters trick the user's vestibular sense by

- Splitting accelerations into low and high frequency components
- Applying high frequency components directly (as they require limited workspace)
- Applying low frequency components by taking advantage of gravity (tilting the platform)

Simulators

Rendering fast vehicles without moving much

Why does this work?

- Platform tilting is not perceived if it happens slower than 3 DEG/sec