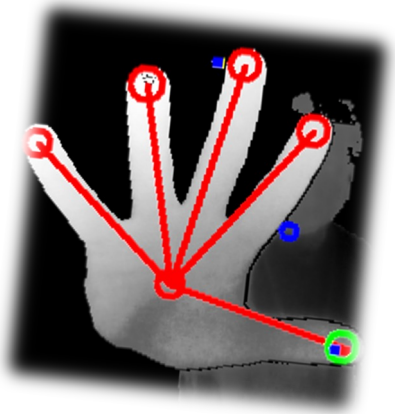
LYU1302

Perceptual Application Development using Intel Perceptual Computing SDK

Agenda

- Introduction
  - Background
  - Motivation
  - Objective
- Design & Implementation
  - Data processing
  - Data to movement
  - Problem solving
- Conclusion
- Demonstration



Introduction

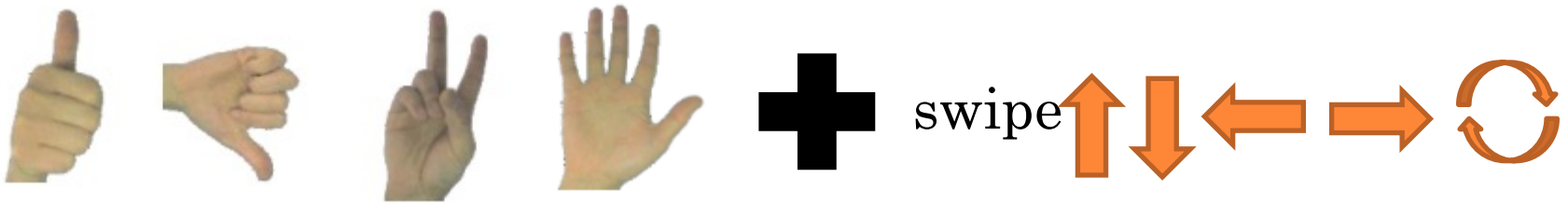
Background---Intel Perceptual Computing

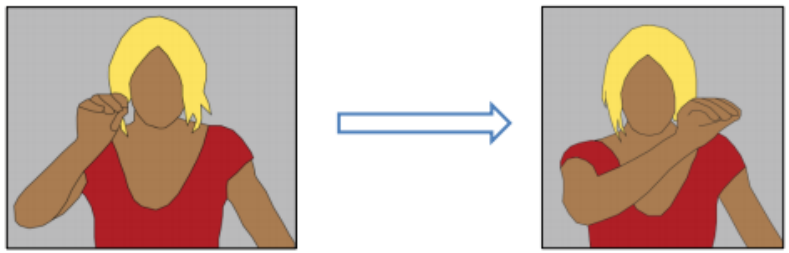
- Perceptual Computing was first introduced in 2012 by Intel Corporation
- For research on user-machine interaction
  - close-range hand and finger gestures
  - speech recognition
  - face recognition and tracking
  - augmented reality
  - background subtraction

close-range hand & finger gestures

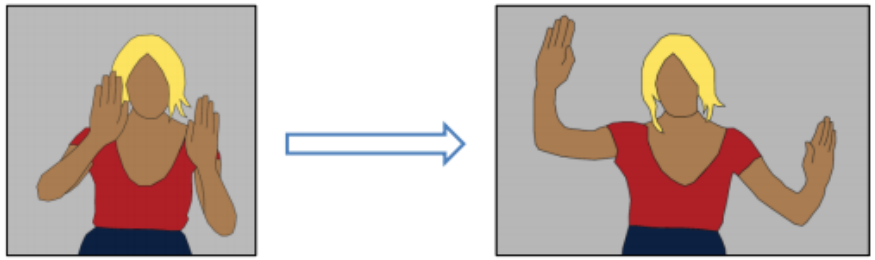
Dramatically change the way human beings interact with machine.

- Close-range hand and finger gestures in SDK
  - *Recognizable Gestures*





A user moves an object.



A user zooms the view.

And more . . . . . .

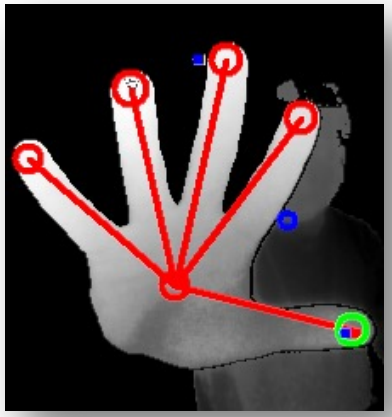
But, we designed our own!

Motivation

- Chinese Lion Dancing
  - Traditional Chinese performance
  - Shown in ceremony or festival party
  - People love it

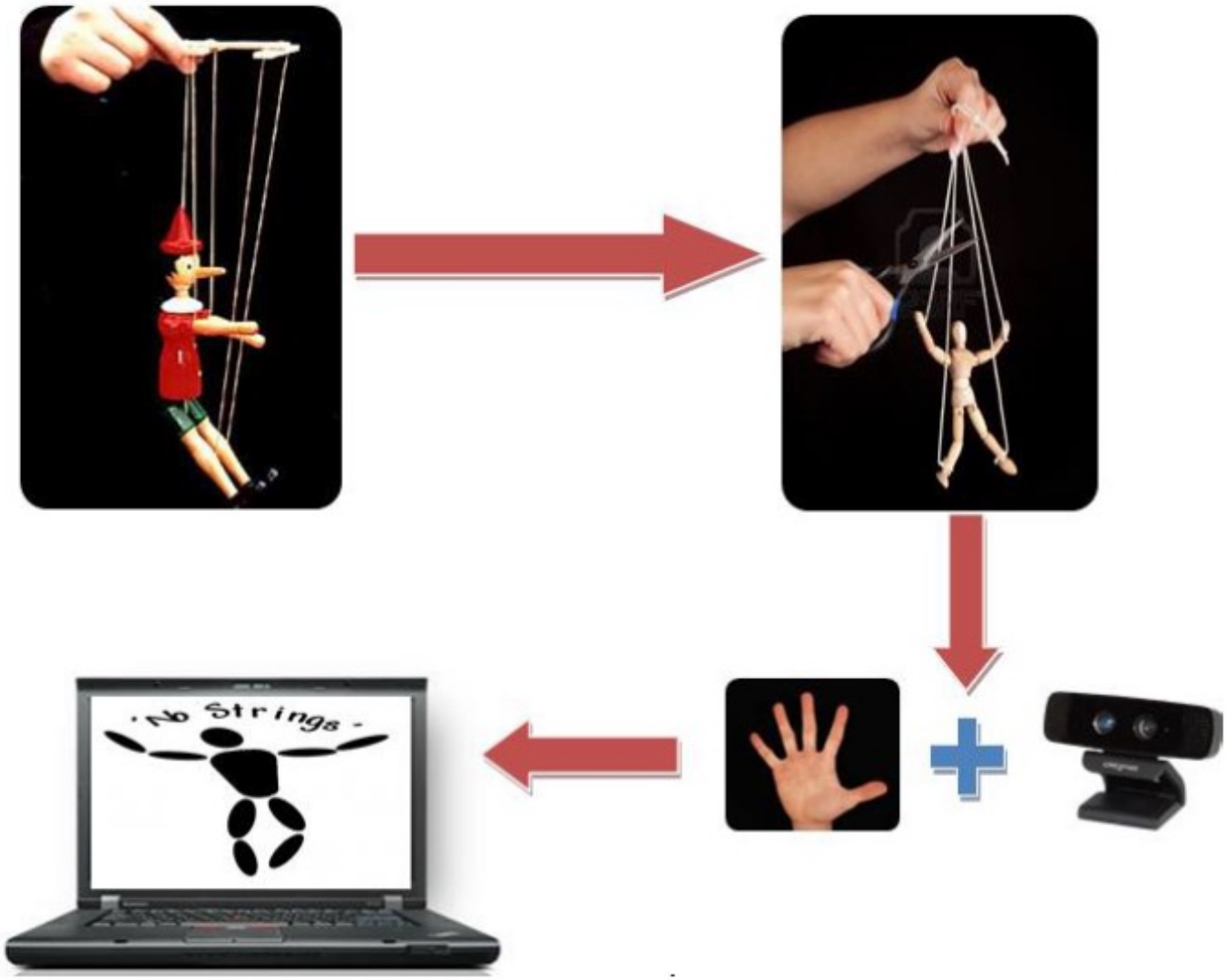
But is it has anything to do with our project?

Hand control + Chinese Lion Dancing



String Lion Puppet

- With Perceptual Computing SDK and Creative* Camera, we can cut the string off

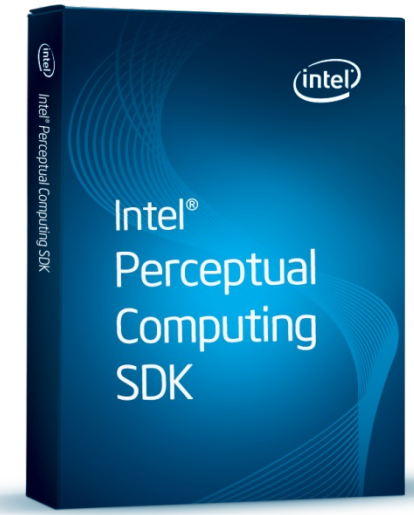


Objective

1. Simulate string lion puppet in virtual world

So that

1. Study Intel Perceptual Computing thoroughly
2. Study model rendering
3. Learn to virtualize realistic behaviors



Expected Result

- String lion puppet's moving direction is correspondent with user hands moving direction
- String lion puppet can make different actions according to user hands gesture

Details are in the next chapter…

Design & implementation

Data processing

- Image Position processing
  - Resolution of depth camera is 320*240
  - Expect string lion puppet moving in the whole screen
  - Mirror effect

- World Position processing-depth coordinate
  - Image position do not have depth information
  - World position with unit meter

```
pxcF32 getxPosition(){
    return (320-this->x)*ofGetWidth()/320;
}
pxcF32 getyPosition(){
    return (this->y)*ofGetHeight()/240;
}
pxcF32 getzPosition(){
    return this->z*5000;
}
```

320-this->x

Mirror effect

*ofGetWidth()/320 & *ofGetHeight()/240

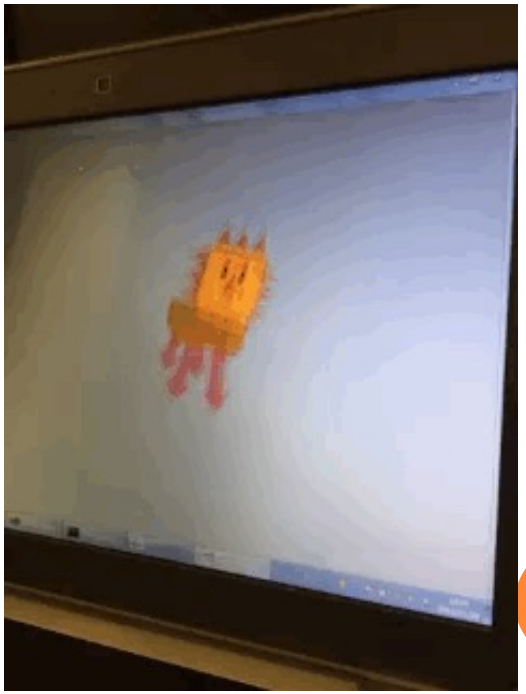
Expand moving range

this->z*5000

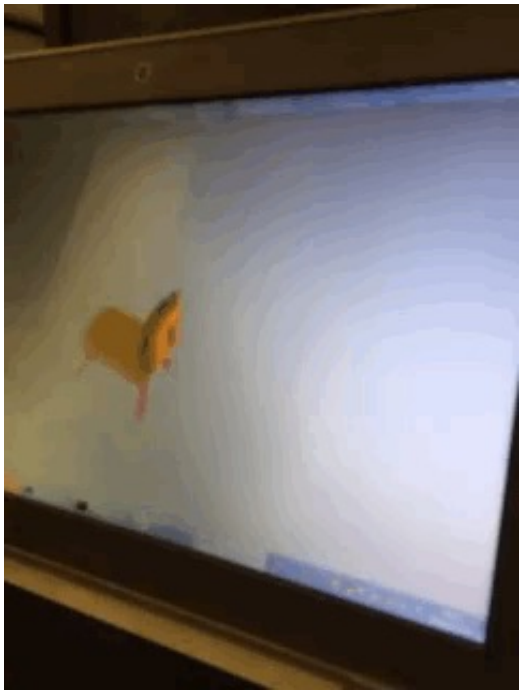
Make a reasonable unit

Data to movement

- Translation
  - Treat a point p(x,y,z) as central position of control hand (* details in next chapter)
  - P(x,y,z) is also lion's position
- Effect exhibition

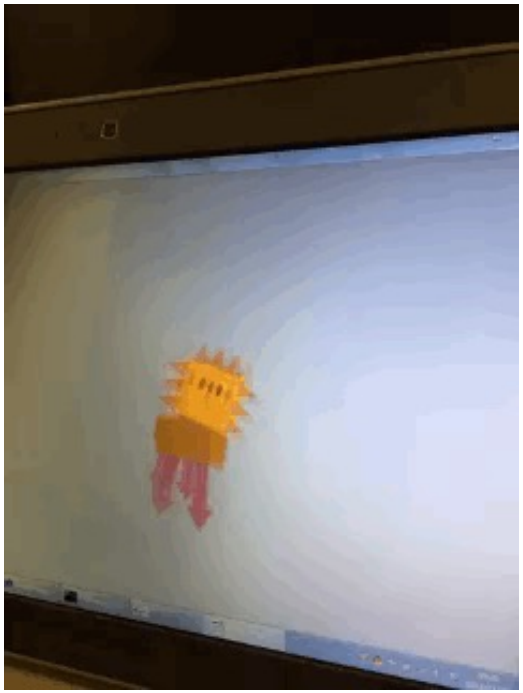


- Rotation
  - Depends on finger and hand's position
  - Rotate with user's hand
  - Decide suitable rotate angle and axis based on fingers position(* details in next chapter)
- Effect exhibition



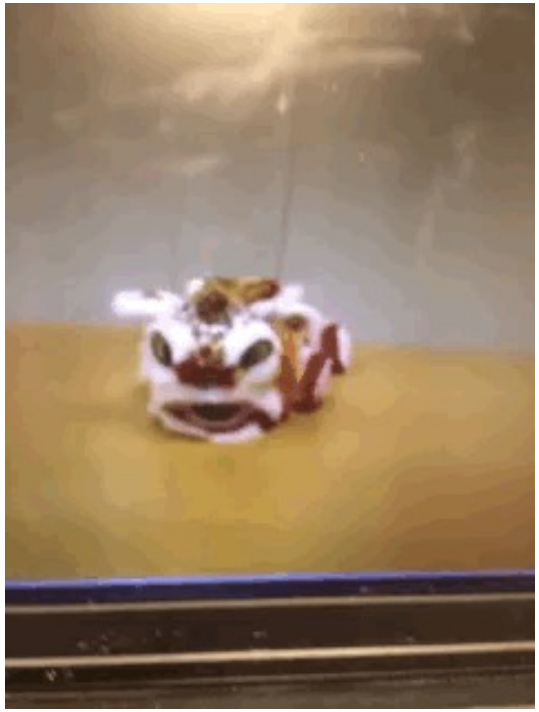
- Leg shaking
  - Shake with user's hand
  - Similar to rotation

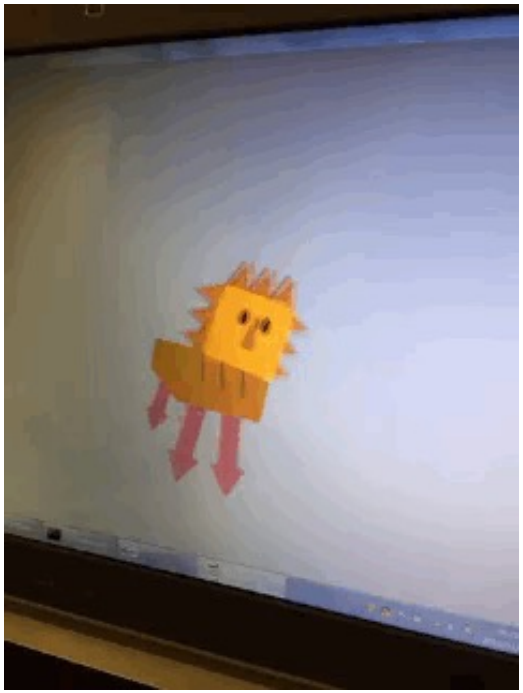
- Effect exhibition



- Head shaking
  - Shake with user's fingers
  - Similar to rotation

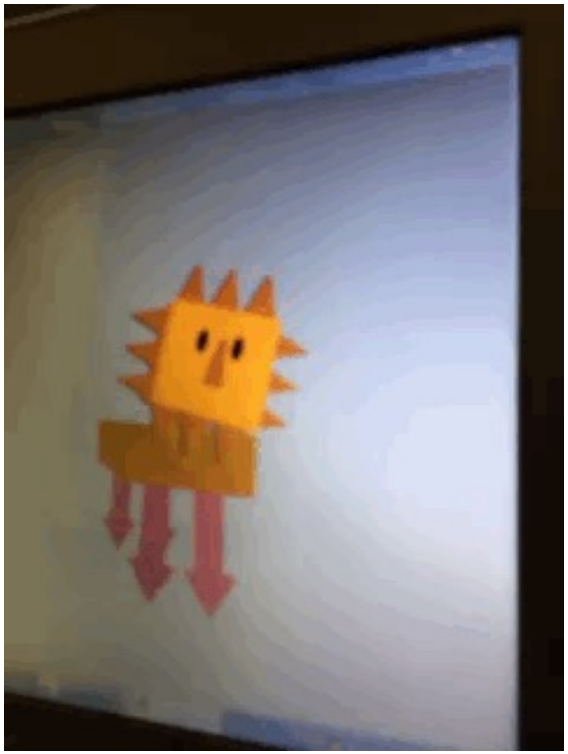
- Effect exhibition



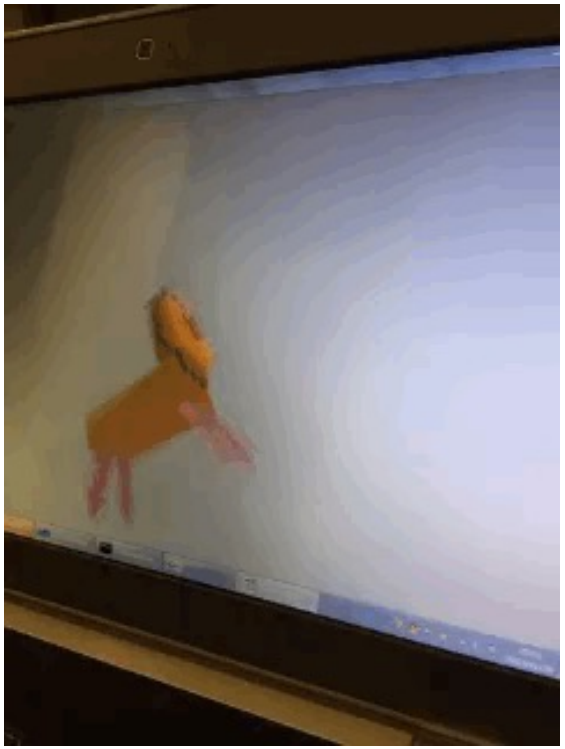


- Some other actions exhibition

Eye changing

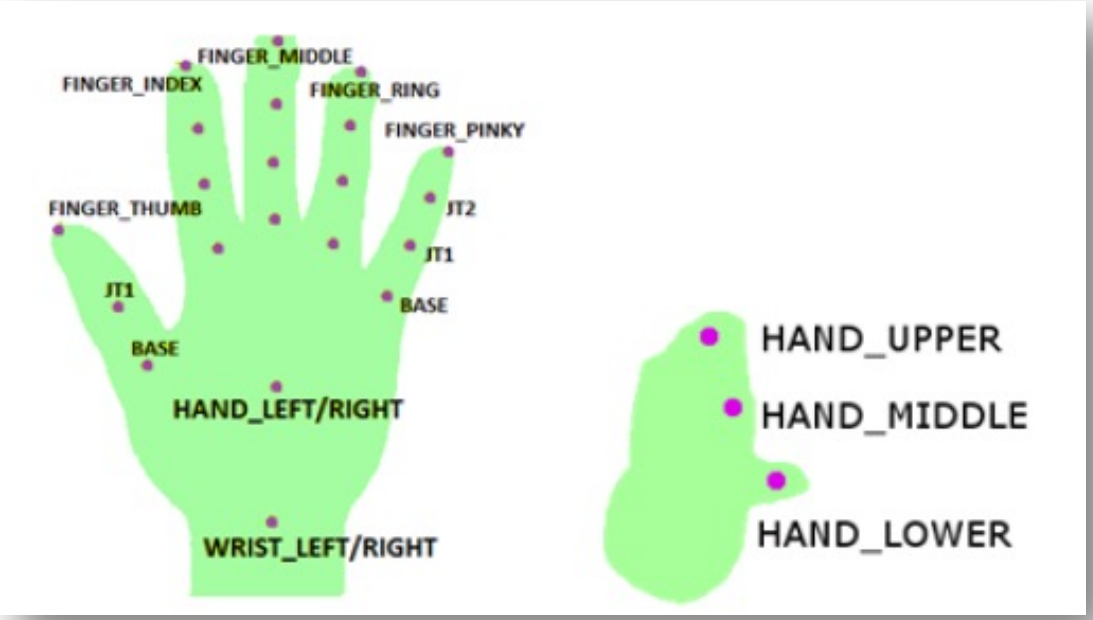


Rotate forward & backward



Problem solving

- Hand central point decision

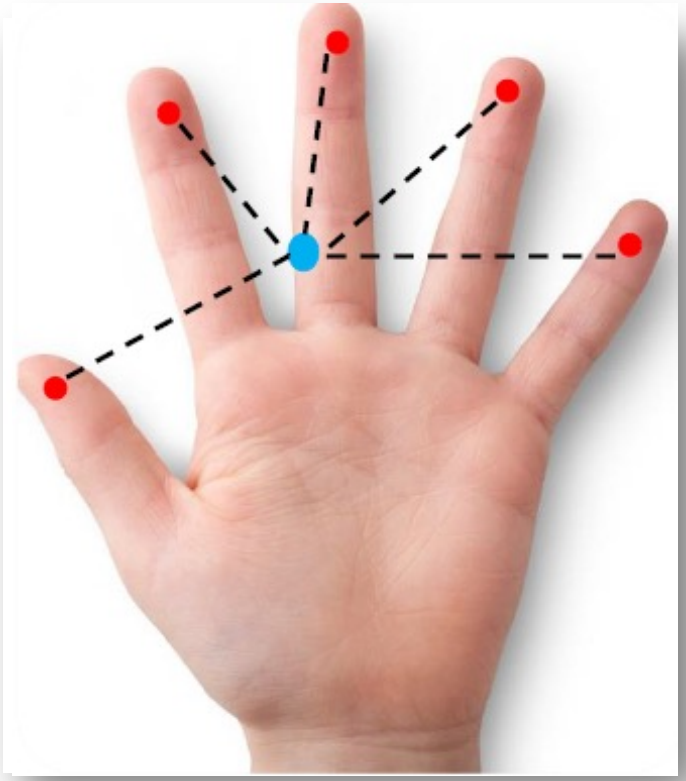


Can we simply choose any one of them?

- Hand central point decision
  - We expect read red position
  - The result can be any one of the green position

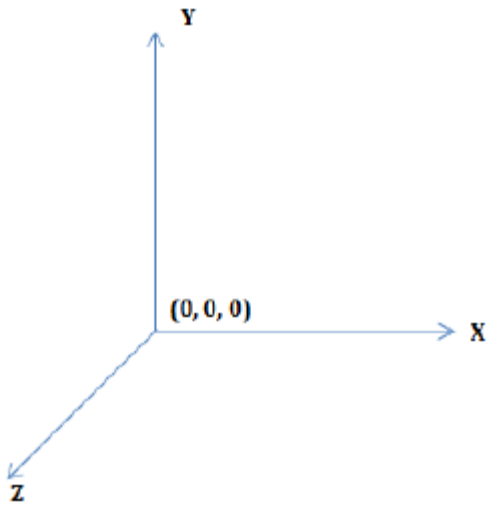
  - Points on finger tips are more stable than points on palm
  - Using multiple finger tips points
  - Use mean position of 5 finger tips of control hand

Can we simply choose any one of them? CAN NOT!

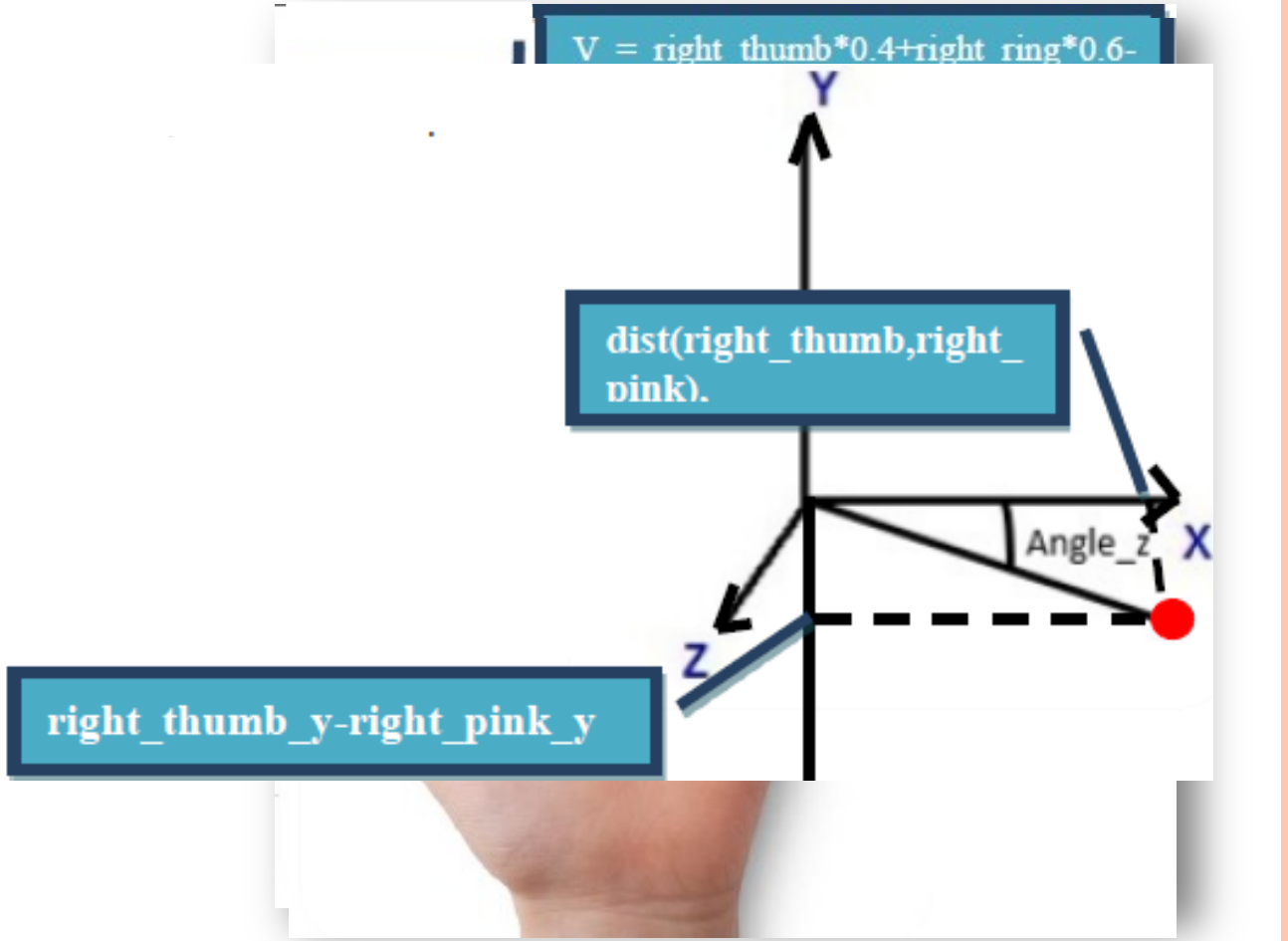


- Rotation angles and axis decision
  - Compute axis and angles at the same time is difficult, too many unknowns
  - Fix rotate axis and only compute angles, rotate around x, y, z axis respectively
  - Compute x, y, z component respectively

Rotate coordinate system



- Rotation angles and axis decision
  - X-Angles
  - Y-Angles
  - Z-Angles



- Left & right hand indistinguishable
  - Inherent limitation
  - Consider the first hand came in to camera range as left hand
  - Our control need to distinguish left and right

- Suggested solution
  - Fix hand entering order (worse user experience)
  - Decide whether SDK judgment on right and left is right
  - If judgment is wrong, swap it

- Lost tracking problem
  - Inherent limitation
  - When lost tracking happened, the lost tracked position would return another tracked position information
  - No warning, cause inconvenience

- Suggested solution
  - Fix point position when lost tracking happened
  - Still not perfect

Conclusion

Conclusion

- *What we got until now*
  - *A simple puppet control program*
  - Basic idea about perceptual computing
  - Knowledge in computer graphics
  - Algorithm of mapping realistic behaviors to virtual behaviors
  - Methods to overcome inherent limitations, code optimizations, algorithm design
  - Better skill, better patience

Conclusion

- *Future plan*
  - Introduce realistic physics engine, make the string lion puppet's behavior more realistic.
  - Algorithm optimization on solving lost-tracking problem; try to reduce and minimize system inherent error.
  - Pay more effort on lion puppet model rendering; make lion puppet's appearance has high similarity as real string lion puppet
  - Optimize the structure of code and data structure, improve code's efficiency, and reduce resource costs.
  - If everything goes well, we will further develop it and utilize the lion puppet to do more things, not merely a puppet movement simulation

Demostration