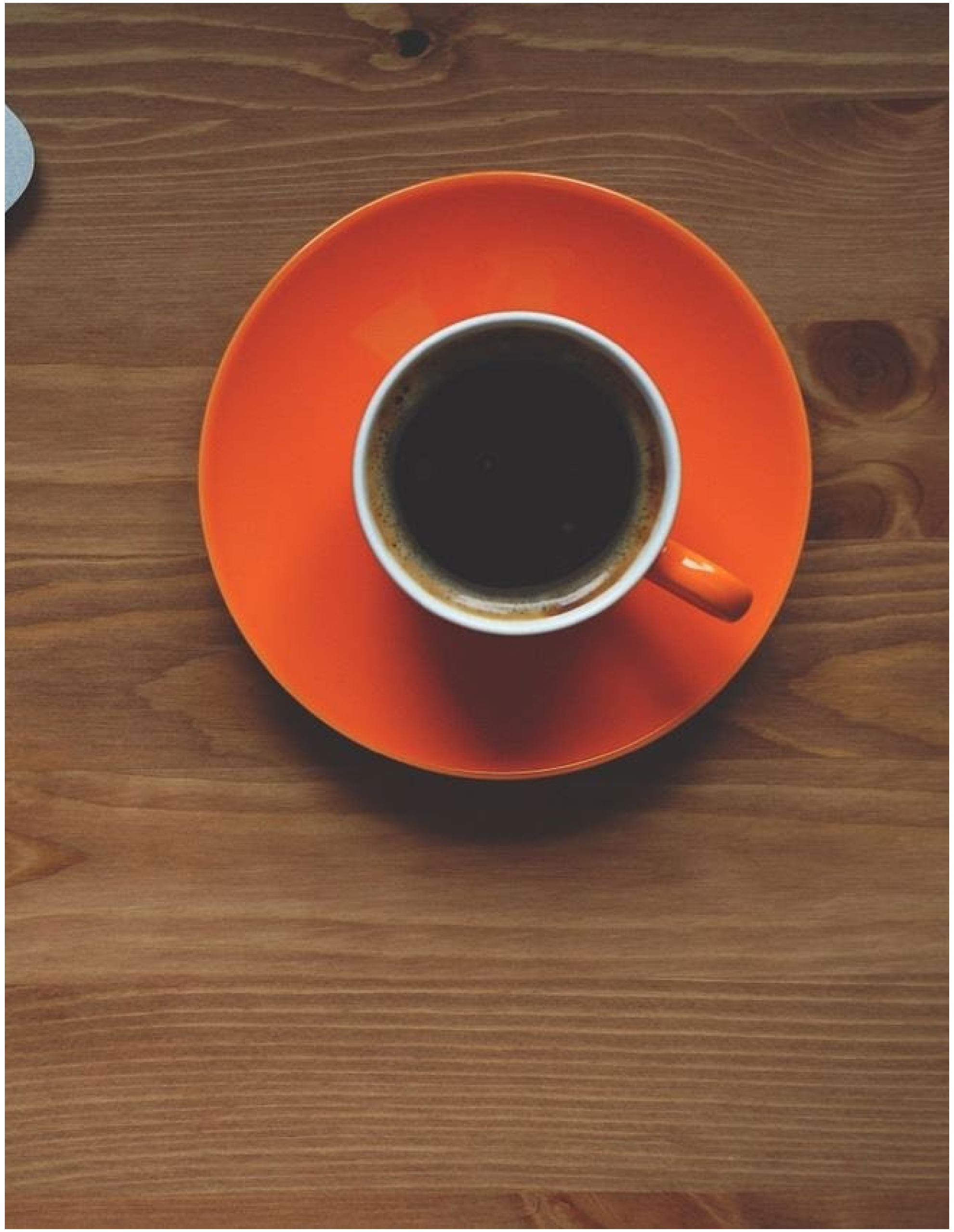
DoGest：一种无线交互系统

刘涵章、蔡畅、张天祺

CONTENTS

CONTENTS



/01

Project Analysis

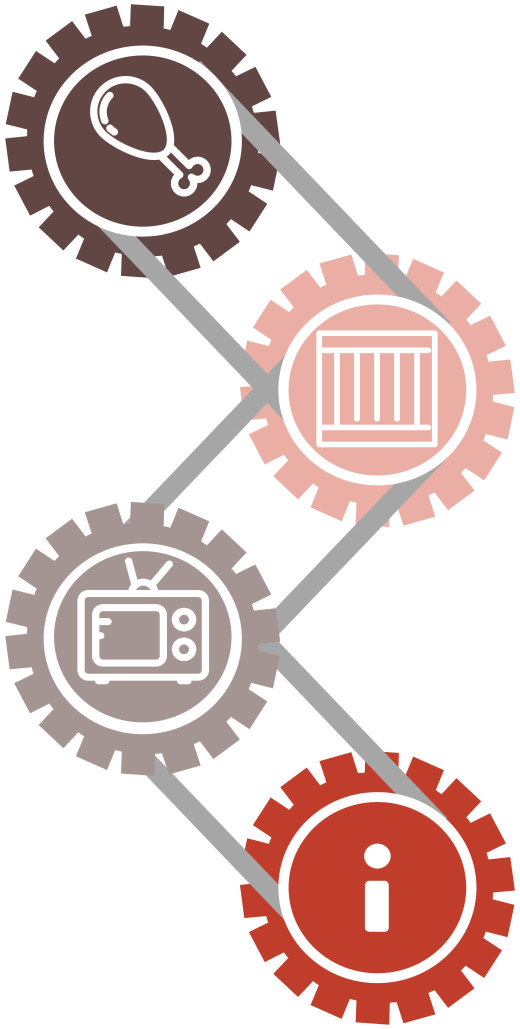
Product Analysis

Real pain

- When you are watching a fantasy show with fry chicken, it's **INCONVENIENT** to press the pause.

Key Market

- Fried chicken lovers
- TV series fans
- Phyzait
- Geek



Contribution

- A wireless interactive system with Sokoban game demo

Distinction

- No extra device required
- Geeky appearance
- Cost-effective

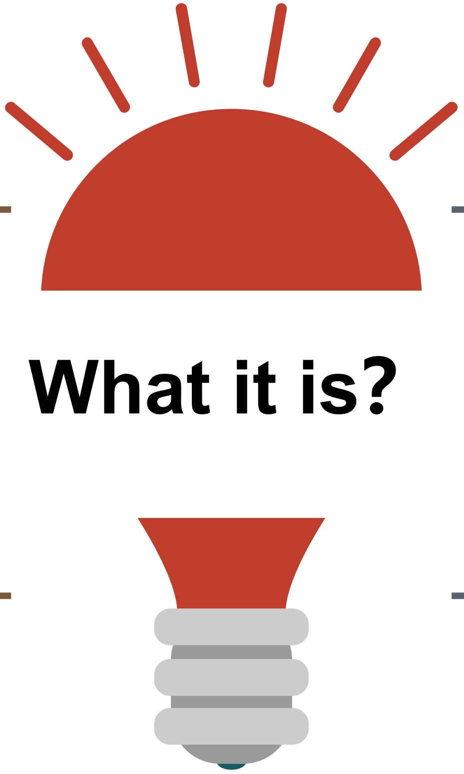
Product Description

TV Remoter

Fry chicken + TV Show = **double** happiness

Game Controller

Sokoban demo is realized using this system.



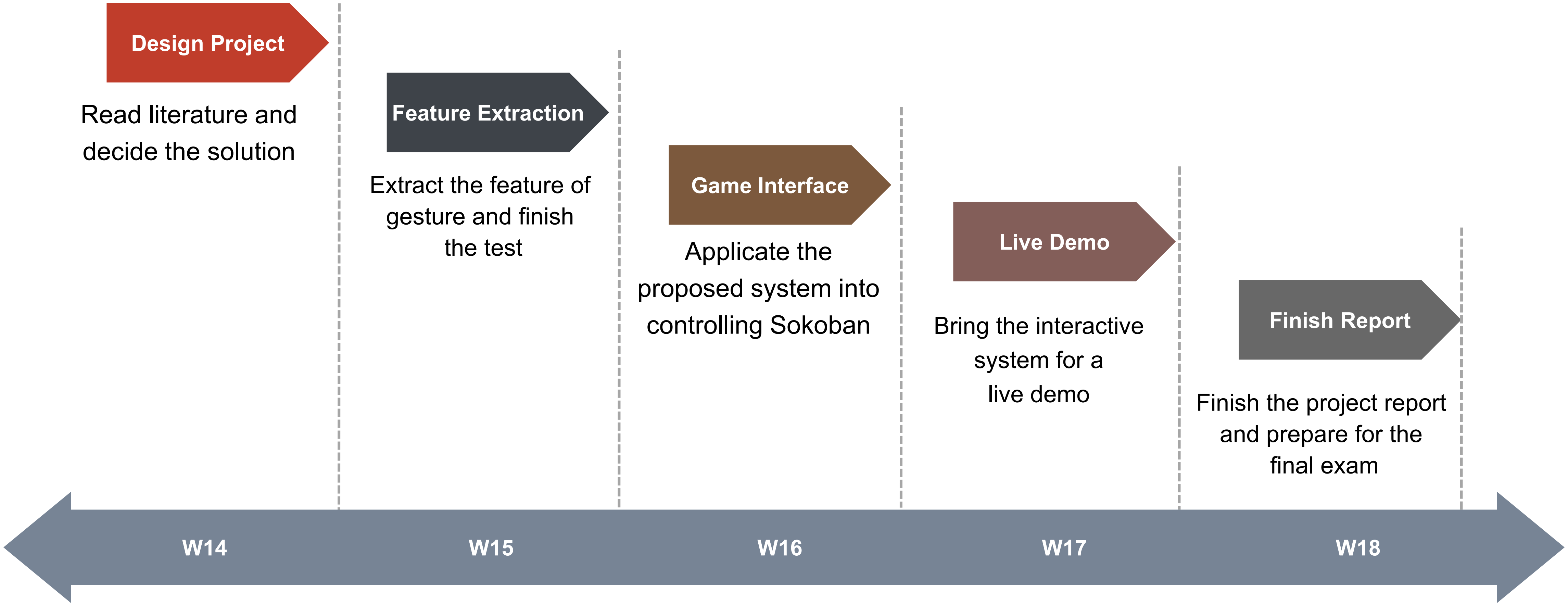
PPT Clicker

Never forget to take the clicker any more.

And More

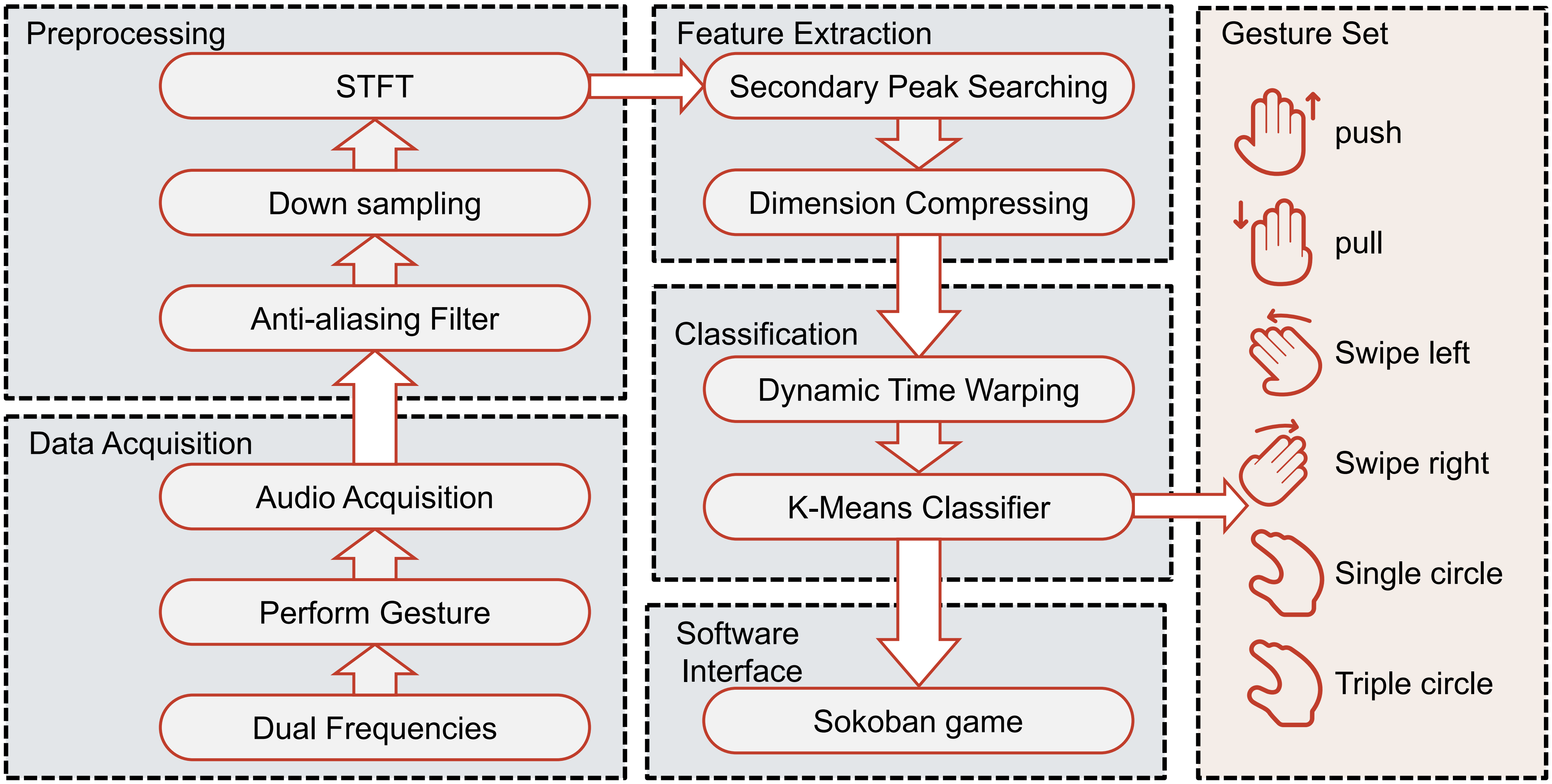
Waiting you to explore…

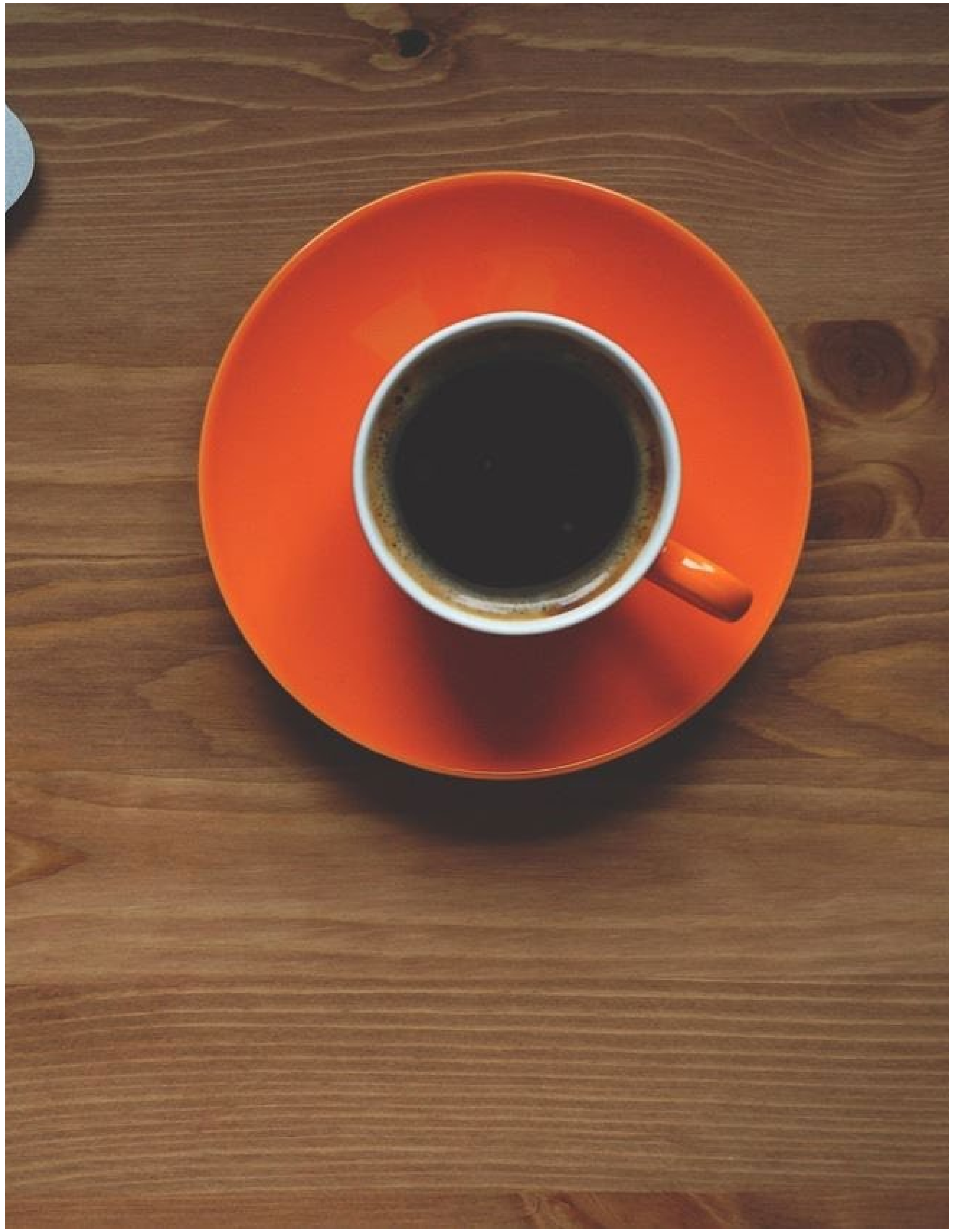
Time line



Design Project
Read literature and decide the solution
Feature Extraction
Extract the feature of gesture and finish the test
Game Interface
Applicate the proposed system into controlling Sokoban
Live Demo
Bring the interactive system for a live demo
Finish Report
Finish the project report and prepare for the final exam
W14
W15
W16
W17
W18

System Architecture





/02

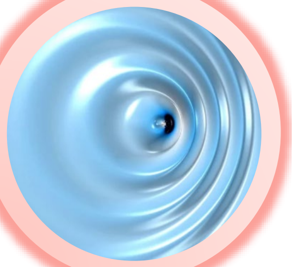
Technology Analysis

Technology

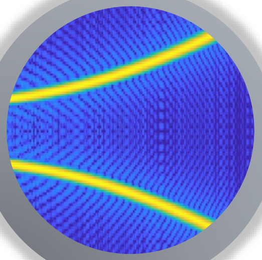
01

Doppler Effect

The basic physical law how to sense the movement



02



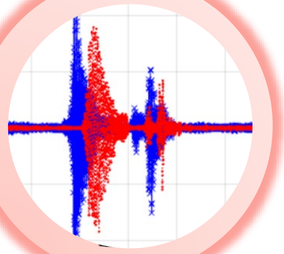
STFT

The signal process of feature extraction

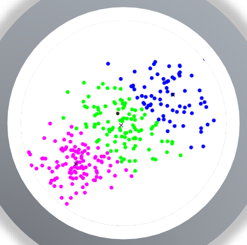
03

Dynamic Time Warping

Enhance the system robust



04



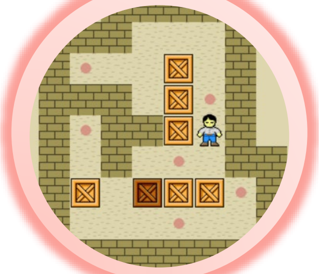
K-Means Classifier

To classify the different gesture

05

Sokoban Demo

A classic game for live demo to show the application



Doppler Effect

Source frequency f_0, received frequency f

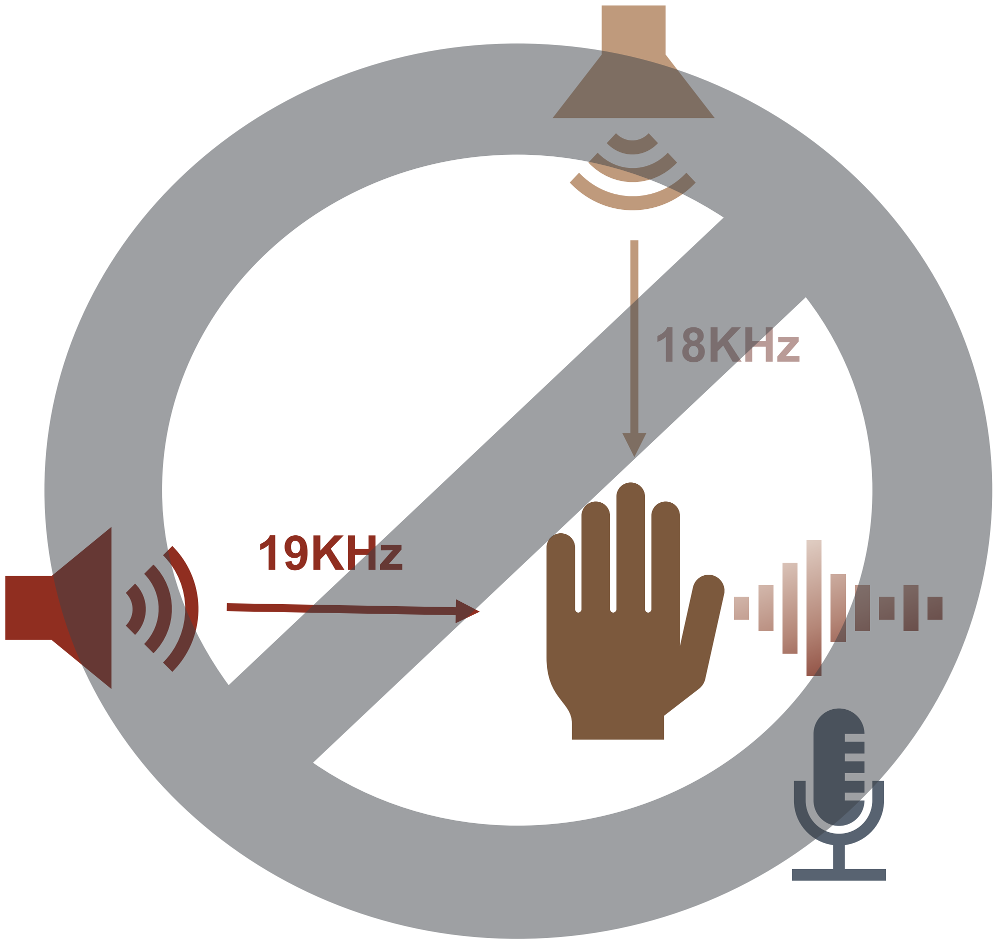
$$f = \frac{c \pm v_r}{c \pm v_t} f_0$$

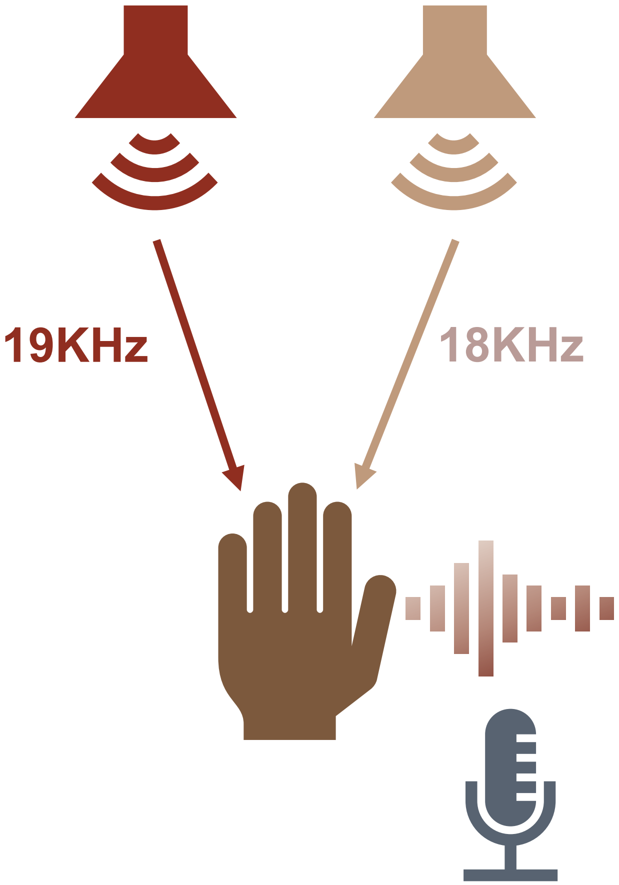
When$c \gg v$

$$\Delta f = f - f_0 = \frac{c+v}{c-v} f_0 - f_0 \approx \frac{2v}{c} f_0$$

Source frequency： 18KHz, 19KHz

What’s new





Doppler Effect

Source frequency f_0, received frequency$f$

$$f = \frac{c \pm v_r}{c \pm v_t} f_0$$

When$c \gg v$

$$\Delta f = f - f_0 = \frac{c + v}{c - v} f_0 - f_0 \approx \frac{2v}{c} f_0$$

The FFT number N_{FFT}, spectral resolution Δn, sample rate F_s

$$\Delta n = \frac{F_s}{N_{FFT}}$$

To get an accuracy result, we set

$$\Delta n \sim \Delta f / 100$$

Source frequency：18KHz, 19KHz

Design specifications：Spectral resolution 2Hz

Down Sampling

The Baseband sampling rate $f_{SB} = 48$KHz, down sample rate k

$$\frac{2f_H}{k} \leq f_s \leq \frac{2f_L}{k-1}$$

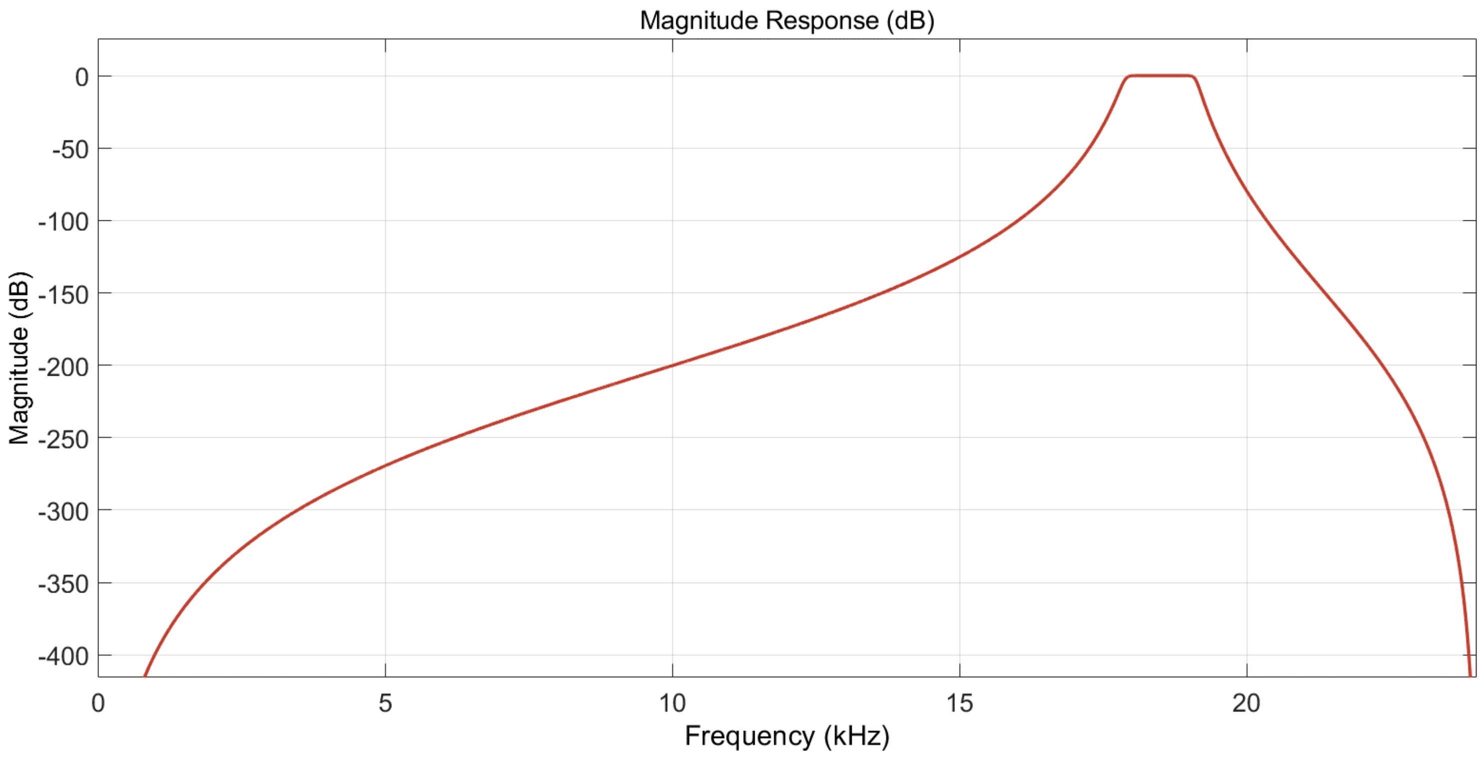
k	$f_s = f_{SB}/k$	(f_L, f_H)	N_{FFT}	Δn
1	48K	(0, 24K)	4096	11.7
5	9.6K	(14.4K, 19.2K)	4096	2.3
6	**8K**	**(16K, 20K)**	**4096**	**2**
7	6.9K	(17.2K, 20.7K)	4096	1.7
8	6K	(18K, 21K)	4096	1.5

Anti-aliasing Filter Design

Butterworth IIR filter, order:18

$F_{pass1} = 18\text{KHz}$, $F_{pass2} = 19\text{KHz}$, $F_{stop1} = 16\text{KHz}$, $F_{stop2} = 20\text{KHz}$

$A_{pass} = 0\text{dB}$, $A_{stop} = -80\text{dB}$



STFT

The define of short time Fourier transform is

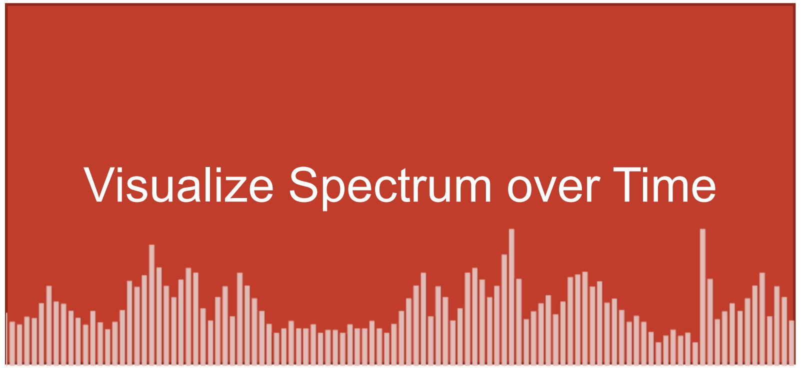
$$STFT(e^{j\omega}, n) = \sum_{m=-\infty}^{\infty} x[n-m]w[m]e^{-j\omega m}$$

We choose the Hamming window.

$$w_{Hamming}[n] = 0.54 + 0.46\cos\left(\frac{2\pi n}{N_w}\right) \qquad -N_w/2 < n \leq N_w/2$$

We set the window interception length $\delta_t = 85$ms, overleap = 75%

$$N_w = \delta_t \cdot f_s = 680$$



Secondary Peak Searching

Algorithm Secondary Peak Searching

1: **Input:** spectrum S of FFT length 4096

2: **Output:** Variable d indicating the direction with respect to the speaker. Value $+1$ for forward, -1 for backward and 0 for stationary.

3: $N \leftarrow$ the FFT point w.r.t. the frequency of the speaker.

$\Delta n \leftarrow$ spectrum resolution.

4: $L \leftarrow \{k \mid S[k] \geq \beta, N - \frac{100}{\Delta n} \leq k < N\}$

$l \leftarrow \max_{k \in L} S[k]$

$R \leftarrow \{k \mid S[k] \geq \beta, N < k \leq N + \frac{100}{\Delta n}\}$

$r \leftarrow \max_{k \in R} S[k]$

5: **if** $l > r$ **then**

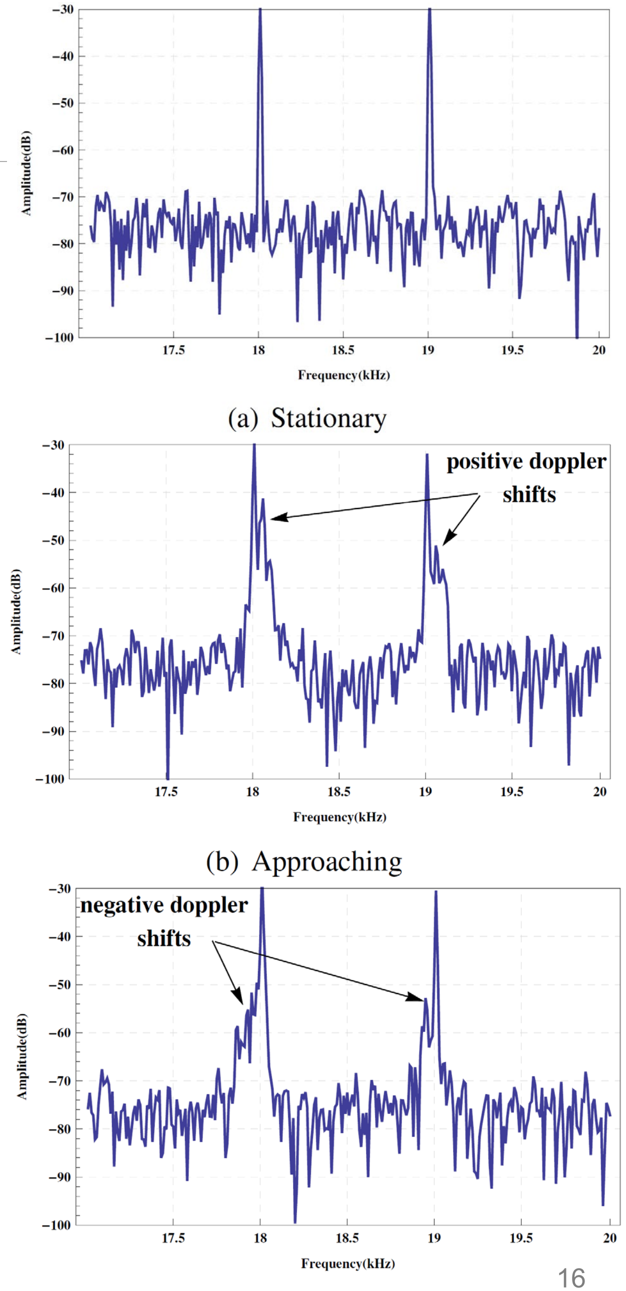
$d \leftarrow -1$

6: **if** $l < r$ **then**

$d \leftarrow 1$

7: **else**

$d \leftarrow 0$

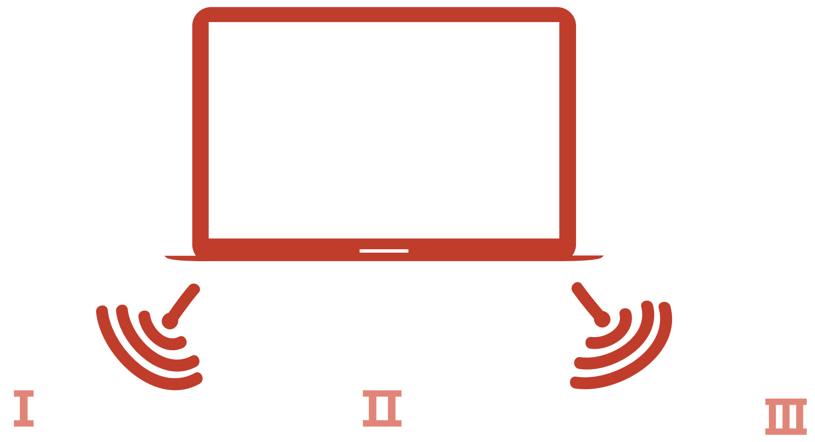


(a) Stationary

(b) Approaching

(c) Departing

Feature Extraction



Gesture	Feature	Label
push	[$(1,1)^+$]	1
pull	[$(-1,-1)^+$]	2
swipe left	[$(1,1)^+$,$(1,0)^+$,$(1,-1)^+$,$(0,-1)^+$,$(-1,-1)^+$]	3
swipe right	[$(1,1)^+$,$(0,1)^+$,$(-1,1)^+$,$(-1,0)^+$,$(-1,-1)^+$]	4
…	…	…

k-NN Classifier

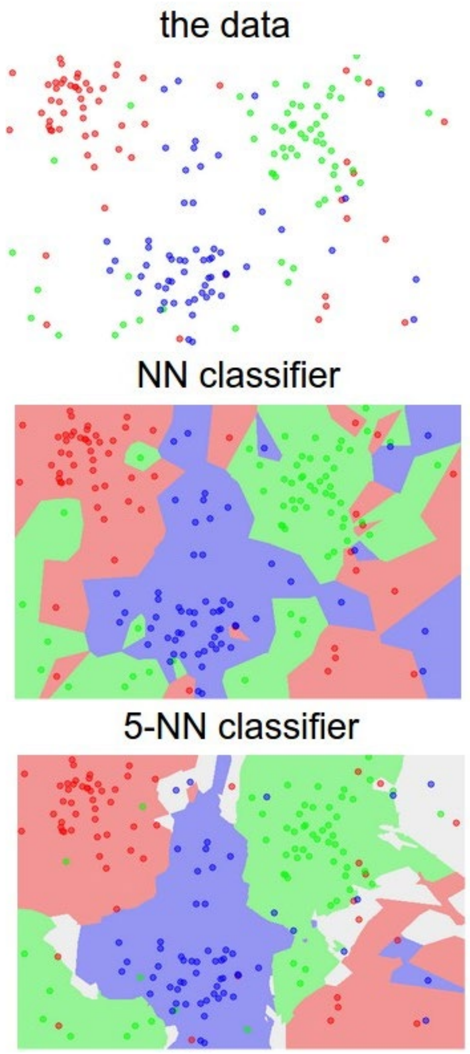
k-NN is a simple classification algorithm containing two steps:

- Compute the distances between test example and all train examples.
- Given these distances, for each test example find the k nearest examples and have them vote for the label

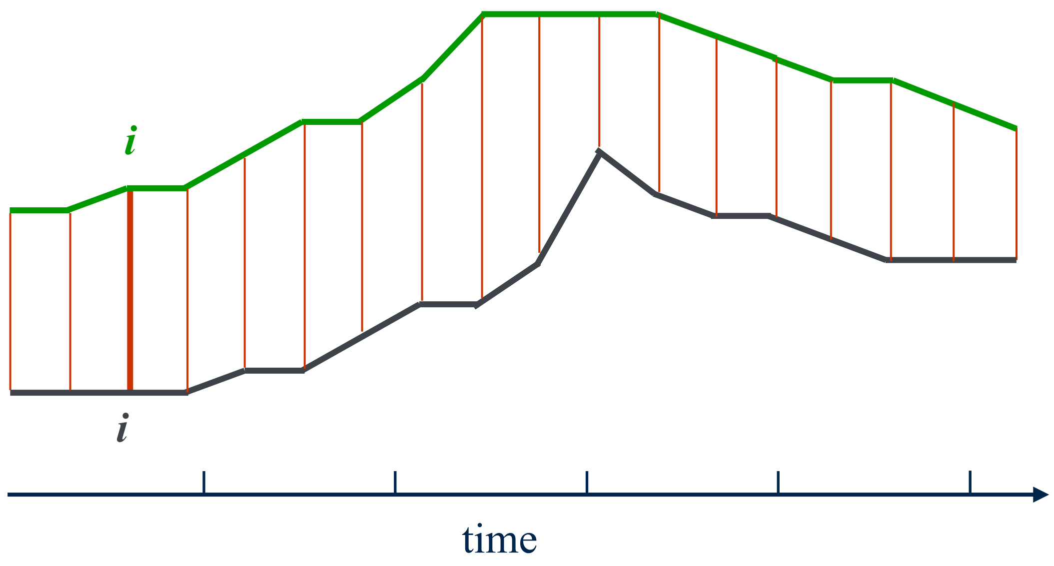
Question:

How to calculate the “distance”?

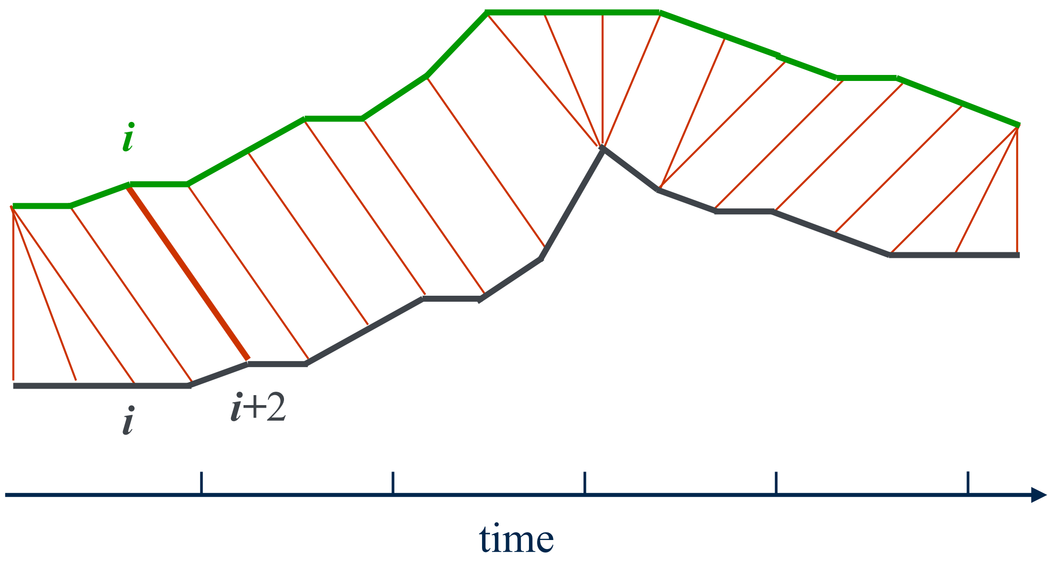
Is it reasonable to use L1 or L2 metrics?



Dynamic Time Warping

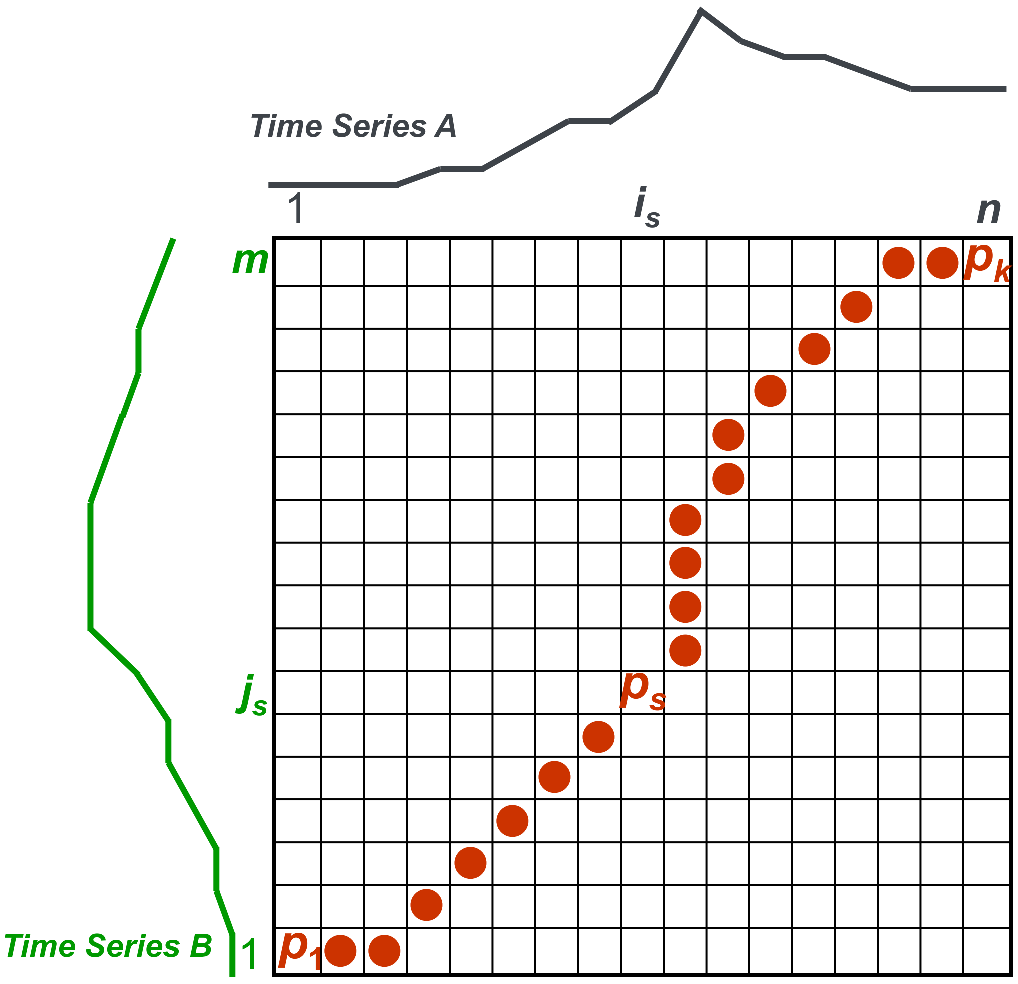


Euclidean



Dynamic Time Warping

Dynamic Time Warping



Time-normalized distance between A and B :

$$D(A, B) = \left[\frac{\sum_{s=1}^{k} d(p_s) \cdot w_s}{\sum_{s=1}^{k} w_s} \right]$$

$d(p_s)$: distance between i_s and j_s

$w_s > 0$: weighting coefficient.

Best alignment path between A and B :

$$P_0 = \arg\min_{P} (D(A, B)).$$

The Choice of the Weighting Coefficient

Time-normalized distance between ***A*** and ***B*** :

$$D(A\,,B\,) = \min_P \left[\frac{\sum_{s=1}^{k} d(p_s)\cdot w_s}{\sum_{s=1}^{k} w_s} \right].$$

complicates optimization

Seeking a weighting coefficient function which guarantees that:

$$C = \sum_{s=1}^{k} w_s$$

is independent of the warping function. Thus

$$D(A\,,B\,) = \frac{1}{C}\min_P \left[\sum_{s=1}^{k} d(p_s)\cdot w_s \right]$$

can be solved by use of dynamic programming.

Weighting Coefficient Definitions

- *Symmetric form*

  $w_s = (\boldsymbol{i}_s - \boldsymbol{i}_{s\text{-}1}) + (\boldsymbol{j}_s - \boldsymbol{j}_{s\text{-}1}),$

  then $C = \boldsymbol{n} + \boldsymbol{m}.$

- *Asymmetric form*

  $w_s = (\boldsymbol{i}_s - \boldsymbol{i}_{s\text{-}1}),$

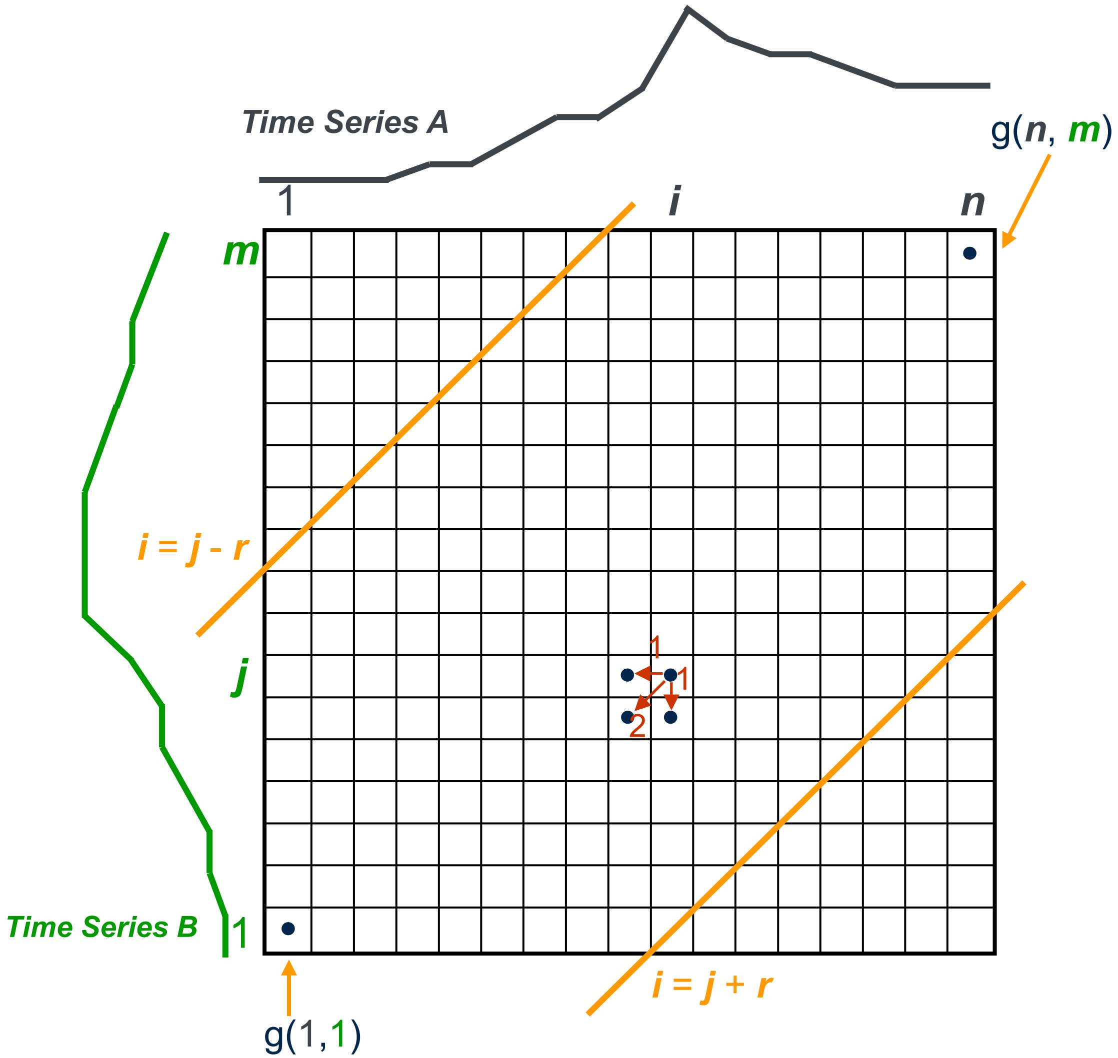
  then $C = \boldsymbol{n}.$

  Or equivalently,

  $w_s = (\boldsymbol{j}_s - \boldsymbol{j}_{s\text{-}1}),$

  then $C = \boldsymbol{m}.$

Symmetric DTW Algorithm



Initial condition: $g(1,1) = 2d(1,1)$.

DP-equation:

$$g(i, j) = \min \begin{pmatrix} g(i, j-1) + d(i, j) \\ g(i-1, j-1) + 2d(i, j) \\ g(i-1, j) + d(i, j) \end{pmatrix}.$$

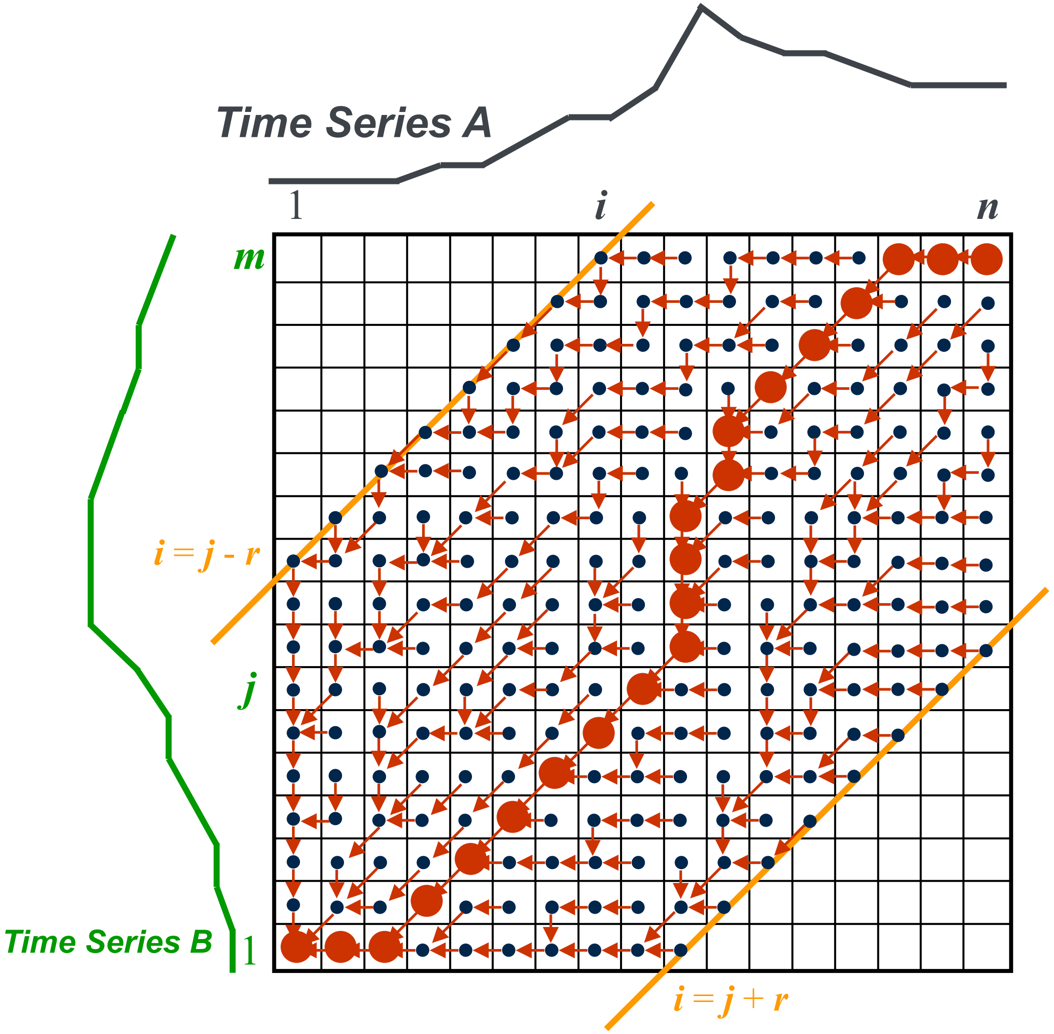
Warping window: $j - r \le i \le j + r$.

Time-normalized distance:

$$D(A, B) = g(n, m) / C$$

$$C = n + m.$$

DTW Algorithm at Work



Start with the calculation of g(1,1) = d(1,1).

Calculate the first row g(***i***, 1) = g(***i***–1, 1) + d(***i***, 1).

Calculate the first column g(1, ***j***) = g(1, ***j***) + d(1, ***j***).

Move to the second row g(***i***, 2) = min(g(***i***, 1), g(***i***–1, 1), g(***i*** – 1, 2)) + d(***i***, 2). Book keep for each cell the index of this neighboring cell, which contributes the minimum score (red arrows).

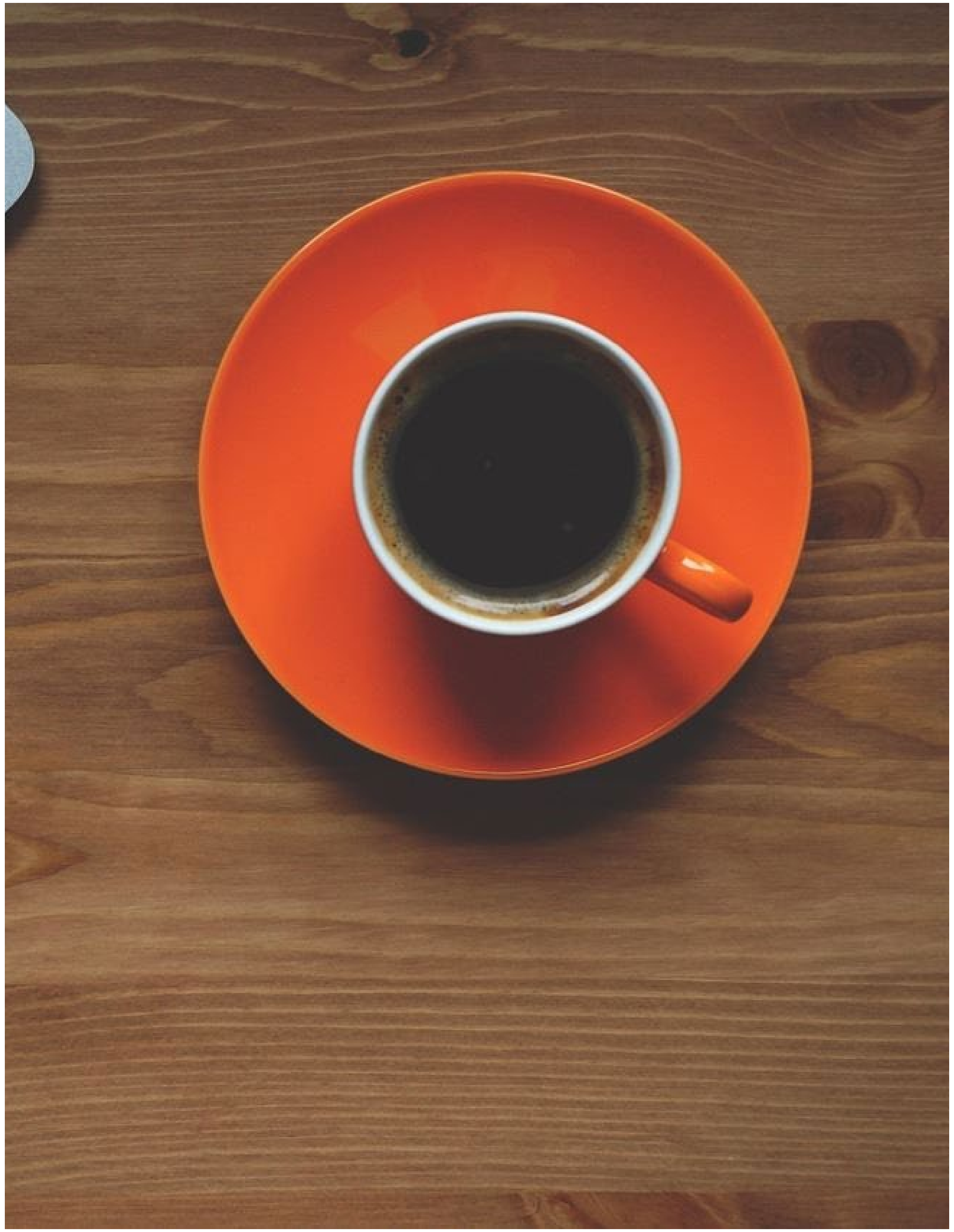
Carry on from left to right and from bottom to top with the rest of the grid g(***i***, ***j***) = min(g(***i***, ***j***–1), g(***i***–1, ***j***–1), g(***i*** – 1, ***j***)) + d(***i***, ***j***).

Trace back the best path through the grid starting from g(***n***, ***m***) and moving towards g(1,1) by following the red arrows.

Result

Classified Gestures

	push	pull	swipe left	swipe right	single loop	triple loop
push	0.99	0.00	0.00	0.01	0.01	0.01
pull	0.00	0.98	0.00	0.00	0.01	0.02
swipe left	0.00	0.00	0.94	0.03	0.04	0.02
swipe right	0.00	0.00	0.05	0.95	0.03	0.01
single loop	0.01	0.01	0.01	0.01	0.89	0.02
triple loop	0.00	0.01	0.00	0.00	0.02	0.92

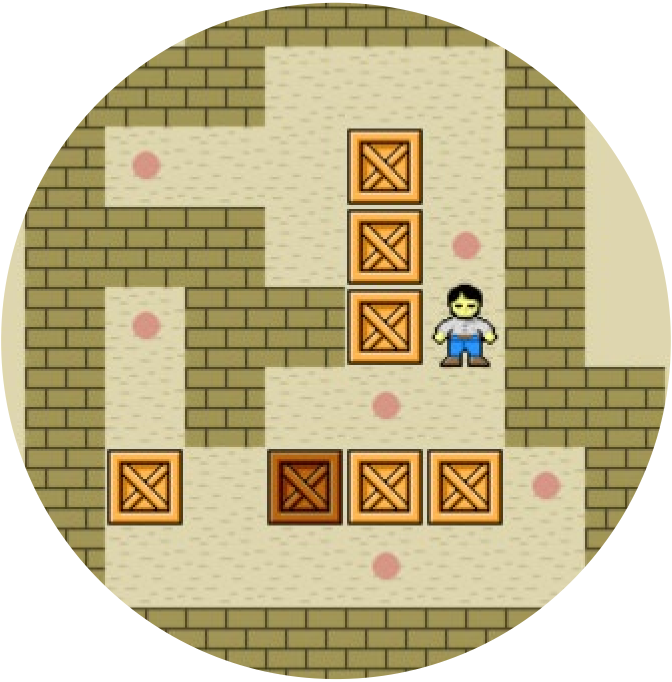


/03

Live Demo

Live Demo

Sokoban



Gestures:

- Left/Right/Forward/Backward
- Circle(s)

Recognition Time:

- 2s (When pressing keyboard ‘z’)

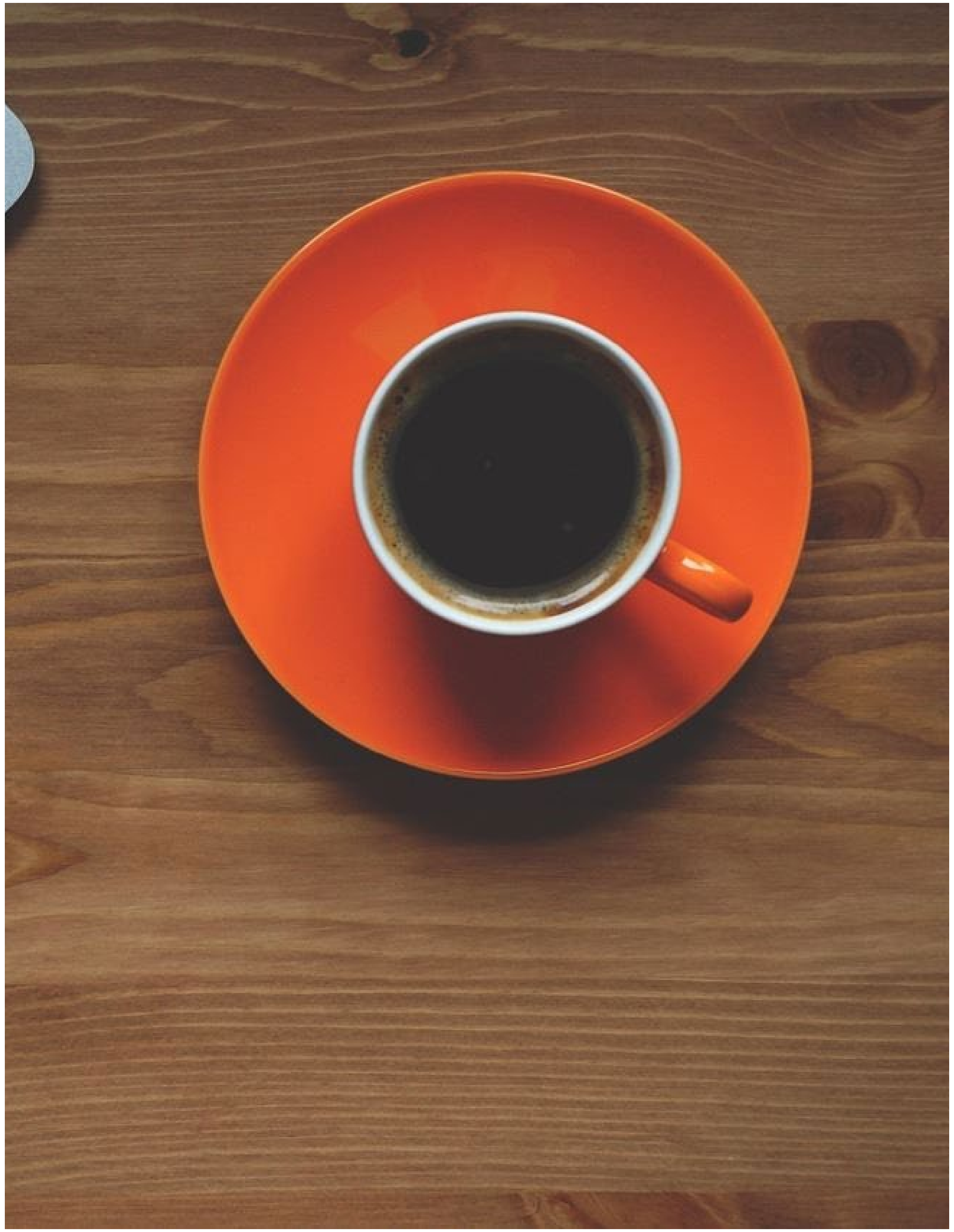
Fruit Ninja

Gestures:

- Left/Right

Recognition Time:

- All the time (2s delay)



/04

Q&A


THANKS