Intelligent Machine Programming

Lecturer: Zhang Bin
Email: zhangbin@kanagawa-u.ac.jp

Objectives:
This course aims to provide students with the foundation of programming languages and basic knowledge of intelligent machines. Through practical exercises, students will acquire skills to implement information processing methods for intelligent machines using programming. The goal is to leverage artificial intelligence technology for the realization of intelligent machines.

Course Content:
The course covers the basics of programming languages, followed by fundamental knowledge of intelligent machines. Students will learn how to implement information processing methods for intelligent machines through programming.

Teaching Approach:
The course is divided into 'lectures,' where students learn the basics of program creation, and 'exercises,' where they apply their knowledge to create programs. The first 10 minutes of each lecture will be dedicated to reviewing the previous session and completing programming assignments.

Class Format:
All sessions will be conducted in person.

Class Management:
Students are required to take detailed notes during the lectures. The first 10 minutes of each session will be used for reviewing the previous material and completing assignments.

Course Schedule:

1. Orientation

Understand the overall course structure.

2. IF-ELSE for Flow Control

Learn the usage of IF-ELSE statements through lectures and exercises.

3. FOR Loops for Iterative Processing

Learn the usage of FOR loops through lectures and exercises.

4. WHILE Loops for Iterative Processing

Learn the usage of WHILE loops through lectures and exercises.

5. Algorithm Design

Learn methods for algorithm design.

6. Algorithm Design and Programming Exercises

Continuing from the fifth session to learn algorithm design methods and perform programming exercises.

7. Project based learning, and Q&A

Confirm the understanding of the contents before.

8. Basics of POINTER

Learn the usage of POINTER through lectures and exercises.

9. Basics of CLASS

Learn the usage of CLASS through lectures and exercises.

10. Foundations of Intelligent Machines – Robot Vision Part 1

Learn basic knowledge about robot vision, especially methods related to image processing: image input/output.

11. Foundations of Intelligent Machines – Robot Vision Part 2

Implement programming solutions for robot vision tasks (image processing exercise).

12. Foundations of Intelligent Machines – Robot Vision Part 3

Implement programming solutions for robot vision tasks (face recognition).

13. Foundations of Intelligent Machines – Robot Vision Part 4

Continue learning foundational knowledge about robot vision, with a focus on methods related to video processing.

14. Foundations of Intelligent Machines – Robot Vision Part 5

Implement programming solutions for robot vision tasks (object tracking).

15. Foundations of Intelligent Machines – Robot Vision Part 6

Recent research works in robot vision.

16. Project based learning, and Q&A

Confirm the understanding of the contents covered the full sessions.

1. Programming language・Features

C

C++

Python

Matlab

Java

…

1.1 Relationships and differences between C++ and C

1.2 Function

```
data_type    name  ()
{
        content；//finished by “ ; ”
}
```

Example：

```
#include<iostream>
using namespace std ;
void main ()
{
        cout<< “hello!”; // “string”, output in its original shape, including space and special characters
        cout<< endl; //change line
}
```

1.3 Data type

int	integer	32bit
double	decimal	64bit
float	decimal	32bit
char	alphabet	8bit

1.4 Variable

```
#include<iostream>
using namespace std ;
void main ()
{
  int a;
  a=1; //if 1.4 ?
  cout<<a;
}
```

1.5 Four arithmetic operations, remainder (remainder of division), logic operations

+ - * /

%

and or not && || !

1.6 Input and output cin cout

How to input or output continuously ?

Exercise：

(1) Output the following shape.

```
*
**
***
 **
  *
```

(2) Write a program to calculate 3+5*4-8%2.

(3) Write a program to calculate a+b-c. However, input a=1.4,b=2.5,c=3 by a keyboard.

2. if function

2.1

```
  if(condition)
    {
       content;
  }
```

2.2

```
    if(condition)
   {
      content 1;
   }
   else   // optional
   {
      content 2;
   }
```

2.3

```
    if(condition 1)
   {
      content 1;
   }
   else if(condition 2)// any number of conditions
   {
      content 2;
   }
   else   // optional
   {
      content 3;
   }
```

Excercise：

(1) Input a score and show its level. (Excellent, good, medium, passed, failed)

3. Loop function：for function

```
for(define variables；condition；added command)
{
   content；//repeat
}
```

Notice ：Which content is optional?

Exercise:

(1) Write a program to calculate 1+2+3+…+10.

(2) Write a program to calculate 1*2*3*…*10.

(3) Write a program to output even number within 100.

(4) Write a program to output odd number within 100.

(5) Write a program to output common multiple of 3 and 5 within 100.

(6) Write a program to output 3 or 7 multiples within 100.

4.　Loop function：while function

```
while(condition)
{
   content；//repeat
}
```

Notice：　How to use “break” and “continue” commands?

Exercise (using while function):

(1) Write a program to calculate 1+2+3+…+10.

(2) Write a program to calculate 1*2*3*…*10.

(3) Write a program to output even number within 100.

(4) Write a program to output odd number within 100.

(5) Write a program to output common multiple of 3 and 5 within 100.

(6) Write a program to output 3 or 7 multiples within 100.

5. Algorithm design

Array　int a[5]

Exercise:

(1) Output the array {4,1,5,3,2} in descending/ascending order.

6. Algorithm Design and Programming Exercises

How to use visual studio 2022.

Exercise:

(1) cout<<4/2+(1+3)*5-15%4;

(2) cout<<((a=2)&& (b=-1));

(3)

```cpp
#include <iostream>
using namespace std;

int main()
{
    int f, f1 = 0, f2 = 1;
    for (int i = 3; i <= 6; i++)
    {
        f = f1 + f2;
        f1 = f2; f2 = f;
    }
    cout << f << endl;
    return 0;
}
```

(4)

```cpp
#include <iostream>
using namespace std;

int main() {
    int i, s = 0;
    for (i = 1; s < 20; i += 2)s += i * i;
    cout << i << endl;
    return 0;
}
```

(5) $\sum_{i=1}^{20} i$

(6) Sum of even numbers within 100.

(7) Sum of common multiples of 3 and 5 within 50.

(8) Product of odd numbers within 20.

7. Project based learning, and Q&A

8. Pointer

```
int * i;
int a;
i=&a;
*i= 5;
```

Exercise:

(1) Write a program to output the array {1,7,2,5,9} in order.

```
#include <iostream>
using namespace std;

int main() {

    int a[5] = {1,7,2,5,9};
    int* p;
    p = a;
    for(int i =0;i<5;i++)
    {
        for (int j = i+1; j < 5; j++)
        {
            if (*(p + i) > *(p + j))
            {
                int  tmp = *(p + j);
                *(p + j) = *(p + i);
                *(p + i) = tmp ;

            }
        }
        std::cout << *(p + i)<<" ";
    }

    return 1;
}
```

9.Class

```
class name
{
   public:    //accessible from outside
           variable；
           void f()；
           name();//constructor
           ~name(); //destructor
   private:      // inaccessible from outside, default, accessible only from the same class
           variable；
           function；
   protected:      // inaccessible from outside, accessible only from the same class or a inheritance class
           variable；
           function；
}
void name:: f()
{
     content；
}
```

Example 1：

```
#include <iostream>
using namespace std;
class Box {
public:
double length;
double width;
double height;
double get(void);
void set(double len, double bre, double hei );
};
double Box::get(void)
{ return length * width * height; }
void Box::set(double len, double wid, double hei)
 { length = len; width = wid; height = hei; }
int main( )
 {
  Box Box1;
  Box Box2;
  double volume = 0.0;
  Box1.height = 5.0;
  Box1.length = 6.0;
  Box1.width = 8.0;
  volume = Box1.height * Box1.length * Box1.width;
  cout << "Box1：" << volume <<endl;
  Box2.set(10.0, 8.0, 7.0);
  volume = Box2.get();
  cout << "Box2：" << volume <<endl;
  return 0;
}
```

Example 2：

```
#include <iostream>
using namespace std;
class Line {
  public:
  void setLength( double len );
  double getLength( void );
  Line(double len); // constructor
  private:
  double length;
 };
Line::Line( double len)
 {
cout << "Object is being created, length = " << len << endl;
length = len;
}
void Line::setLength( double len )
{ length = len; }
double Line::getLength( void )
{ return length; }
int main( )
{
   Line line(10.0);
   cout << "Length of line : " << line.getLength() <<endl;
  line.setLength(6.0);
  cout << "Length of line : " << line.getLength() <<endl;
  return 0;
}
/*
Object is being created, length = 10
Length of line : 10
Length of line : 6

*/
```

Example 3 :

```
#include <iostream>
using namespace std;
class Line {
  public:
  void setLength( double len );
  double getLength( void );
  Line();//constructor
  ~Line(); // destructor
  private:
  double length;
};
Line::Line(void)
{ cout << "Object is being created" << endl; } Line::~Line(void)
{ cout << "Object is being deleted" << endl; } void Line::setLength( double len )
{ length = len; }
double Line::getLength( void )
{ return length; }
int main( )
{
  Line line;
  line.setLength(6.0);
  cout << "Length of line : " << line.getLength() <<endl;
  return 0;
}
/*
Object is being created
Length of line : 6
Object is being deleted
*/
```

10.Robot vision～image processing～

10.1 What is image?

10.2 What is color? What is the difference between color image and gray image?

10.3 What is the size of an image ? The concept of resolution.

10.4 What is image processing ?

10.5 image processing library opencv　（opencv 4.5.3 ver.）

```
#pragma comment(lib, " opencv_world453.lib " )
```

//add `opencv_world453.dll` to system path

11 Robot vision～image processing exercise～

11.1 Read an image and show it on screen.

```
#include <opencv2/opencv.hpp>
#pragma comment(lib, " opencv_world453.lib " )
int main(void)
{
	cv::Mat	image;
	image = cv::imread("C:/opencv/sources/samples/data/lena.jpg");
	if (image.empty() == true) {
		return 0;
	}
	cv::imshow("画像", image);
	cv::waitKey();
	return 0;
}
```

11.2 Save an image.

```
#include <opencv2/opencv.hpp>
#pragma comment(lib, " opencv_world453.lib " )
int main(void)
{
	cv::Mat	srcImage, dstImage;
	srcImage = cv::imread("C:/opencv/sources/samples/data/lena.jpg");

	if (srcImage.empty() == true) {
		return 0;
	}
	cv::flip(srcImage, dstImage, 0);
	cv::imshow("original", srcImage);
	cv::imshow("processed", dstImage);
	cv::imwrite("processed.jpg", dstImage);
	cv::imwrite("processed.png", dstImage);
	cv::waitKey();
	return 0;
}
```

12. Robot vision〜face detection〜

```
//Haar-like feature based face detection

#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>
#include <vector>
#pragma comment(lib, " opencv_world453.lib " )

using namespace cv;
using namespace std;

int main()
{
	CascadeClassifier cascade;
	cascade.load("--filepath--/haarcascade_frontalface_alt.xml");
	Mat img = imread("--filepath--/lena.jpg", IMREAD_UNCHANGED);
	vector<Rect> faces;
	cascade.detectMultiScale(img, faces, 1.1, 3, 0, Size(20, 20));

	for(int i = 0; i<faces.size(); i++)
	{
		rectangle(img, Point(faces[i].x, faces[i].y), Point(faces[i].x + faces[i].width, faces[i].y + faces[i].height), Scalar(0, 0, 255), 3, CV_AA);
	}

	imshow("detect face", img);
	waitKey(0);
}
```

13. Robot vision～video processing～

13.1 Read a video file and show it continuously on screen.

```
#include <opencv2/opencv.hpp>
#pragma comment(lib, " opencv_world453.lib " )
int main(void)
{
	cv::VideoCapture cap;
	cap.open("C:/opencv/sources/samples/data/Megamind.avi");
	if (cap.isOpened() == false) {
		return 0;
	}
	cv::Mat frame;

	for (;;) {
		cap >> frame;

		if (frame.empty() == true) {
			break;
		}
		cv::imshow("再生中", frame);
		cv::waitKey(10);
	}
	return 0;
}
```

13.2 Rewrite a video from another video in different extension.

```
#include <opencv2/opencv.hpp>
#pragma comment(lib, " opencv_world453.lib " )
int main(void)
{
	cv::VideoCapture cap;
	cap.open("C:/opencv/sources/samples/data/Megamind.avi");
	if (cap.isOpened() == false) {
		return 0;
	}
	int   width, height, fourcc;
	double fps;

	width  = (int)cap.get(cv::CAP_PROP_FRAME_WIDTH);
	height = (int)cap.get(cv::CAP_PROP_FRAME_HEIGHT);
	fps    = cap.get(cv::CAP_PROP_FPS);
	fourcc = cv::VideoWriter::fourcc('X', 'V', 'I', 'D');		// Xvid  / file extension .avi
	// other formarts
//	fourcc = cv::VideoWriter::fourcc('D', 'I', 'V', 'X');		// DivX  / .avi
//	fourcc = cv::VideoWriter::fourcc('H', '2', '6', '4');		// H.264 / .wmv
//	fourcc = cv::VideoWriter::fourcc('V', 'P', '9', '0');		// VP9    / .avi
//	fourcc = cv::VideoWriter::fourcc('W', 'M', 'V', '2');		// WMV8  / .wmv
//	fourcc = cv::VideoWriter::fourcc('W', 'M', 'V', '3');		// WMV9  / .wmv
//	fourcc = cv::VideoWriter::fourcc('m', 'p', '4', 'v');		// ISO MPEG-4 / .mp4
//	fourcc = cv::VideoWriter::fourcc('M', 'P', '4', '3');		// MS MPEG-4  / .avi
//	fourcc = cv::VideoWriter::fourcc('D', 'I', 'B', ' ');  // RGB without compression / .avi
	cv::VideoWriter writer;
	writer.open("video.avi", fourcc, fps, cv::Size(width, height));
	cv::Mat frame, dst;

	for (;;) {
		cap >> frame;

		if (frame.empty() == true) {
			break;
		}

		cv::imshow("changed", frame);
		cv::flip(frame, dst, 1);
		writer << dst;
		cv::waitKey(1);
	}

	return 0;
}
```

13.3 Get an image stream from a web camera.

```
#include <opencv2/opencv.hpp>
 #pragma comment (lib, " opencv_world453.lib " )
int main(void)
{
	cv::VideoCapture cap;
	cap.open(0);
	if (cap.isOpened() == false) {
		return 0;
	}
	int   width  = (int)cap.get(cv::CAP_PROP_FRAME_WIDTH);
	int   height = (int)cap.get(cv::CAP_PROP_FRAME_HEIGHT);

	std::cout << "image size " << width << "x" << height << std::endl;

	cv::Mat frame;

	for (;;) {
		cap >> frame;
		if (frame.empty() == true) {
			break;
		}
		cv::imshow("input", frame);
		int key = cv::waitKey(30);

		if (key > 0) {
			break;
		}
	}

	return 0;
}
```

14. Robot vision～object tracking～

//Camshift based object tracking

```
#include <iostream>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui/highgui.hpp>
#pragma comment(lib,"opencv_world453.lib")
using namespace cv;
using namespace std;
Mat image;
bool backprojMode = false;
bool selectObject = false;
int trackObject = 0;
bool showHist = true;
Point origin;
Rect selection;
int LINE_AA = 16;

void onMouse(int event, int x, int y, int, void*) {
	if (selectObject) {
		selection.x = MIN(x, origin.x);
		selection.y = MIN(y, origin.y);
		selection.width = std::abs(x - origin.x);
		selection.height = std::abs(y - origin.y);
		selection &= Rect(0, 0, image.cols, image.rows);
	}
	switch (event) {
	case EVENT_LBUTTONDOWN:
		origin = Point(x, y);
		selection = Rect(x, y, 0, 0);
		selectObject = true;
		break;
	case EVENT_LBUTTONUP:
		selectObject = false;
		if (selection.width > 0 && selection.height > 0)
			trackObject = -1;
		break;
	}
}
int main(void)
{
	VideoCapture video;
	video.open("video.mp4");
	Rect trackWindow;
	int hsize = 16;
	float hranges[] = { 0,180 };
	const float* phranges = hranges;
	if (!video.isOpened()) {
		cout << "can't open your video" << endl;
	}
	namedWindow("Camera", 0);
	setMouseCallback("Camera", onMouse, 0);
	Mat frame, hsv, hue, mask, hist, histimg = Mat::zeros(200, 320, CV_8UC3), backproj;
```

```
bool paused = false;
while (1) {
        Mat frame;
        video >> frame;

        if (frame.empty()) {
                break;
        }
        frame.copyTo(image);
        if (!paused)
        {
                cvtColor(image, hsv, COLOR_BGR2HSV);
                if (trackObject) {

                        inRange(hsv, Scalar(0, 0, 0),
                                Scalar(180, 255, 255), mask);
                        //cout << mask << endl;
                        int ch[] = { 0, 0 };
                        hue.create(hsv.size(), hsv.depth());
                        mixChannels(&hsv, 1, &hue, 1, ch, 1);

                        if (trackObject < 0) {
                                Mat roi(hue, selection), maskroi(mask, selection);
                                calcHist(&roi, 1, 0, maskroi, hist, 1, &hsize, &phranges);
                                normalize(hist, hist, 0, 255, NORM_MINMAX);
                                trackWindow = selection;
                                trackObject = 1;
                        }
                        calcBackProject(&hue, 1, 0, hist, backproj, &phranges);
                        backproj &= mask;
                        RotatedRect trackBox = CamShift(backproj, trackWindow,
                                TermCriteria(TermCriteria::EPS | TermCriteria::COUNT, 10, 1));
                        if (trackWindow.area() <= 1) {
                                int cols = backproj.cols, rows = backproj.rows, r = (MIN(cols, rows) + 5) / 6;

                                trackWindow = Rect(trackWindow.x - r, trackWindow.y - r,
                                        trackWindow.x + r, trackWindow.y + r) &
                                        Rect(0, 0, cols, rows);
                        }
                        if (backprojMode)
                                cvtColor(backproj, image, COLOR_GRAY2BGR);
                        ellipse(image, trackBox, Scalar(0, 0, 255), 3, 16);
                }
        }

        else if (trackObject < 0)
                paused = false;

        if (selectObject && selection.width > 0 && selection.height > 0) {
                Mat roi(image, selection);
                bitwise_not(roi, roi);
        }
        imshow("Camera", image);
        char c = (char)waitKey(10);
        }
}
```

15. Recent research works in robot vision.

http://www.mech.kanagawa-u.ac.jp/lab/zhang_lab/index-e.html

16. Project based learning, and Q&A

References：

[1] http://cvwww.ee.ous.ac.jp/opencv_practice1/

[2] https://www.youtube.com/watch?v=tDRsH2UzZIo

[3] https://kanagawa-u.ent.box.com/s/58551pq2uiiiv5rd67ks3gje27cpmo8r

[4] https://kanagawa-u.box.com/s/gf5y63i39m0502h3sl4g0k69cxxjczle

END