

Advanced AI and Machine Learning Assignment Write Up

by Zach Port-Lincoln

FILE	3805_ZACH_PORT-LINCOLN_ADVANCED_AI_AND_MACHINE_LEARNING_ASSIGNMENT_WRITE_UP_177048_2111612312.DOCX (147.61K)		
TIME SUBMITTED	22-MAR-2019 04:14 PM (UTC+0000)	WORD COUNT	2785
SUBMISSION ID	103029568	CHARACTER COUNT	14519

Advanced AI and Machine Learning

What is a convolution matrix?

A convolution matrix is a set of numbers which, multiplied by a fixed set of numbers, a filter, one is called Gaussian filter, to turn the first set of numbers into something else. These techniques have been used for multiple years in things like image processing in order to blur, sharpen and enhance and emboss.

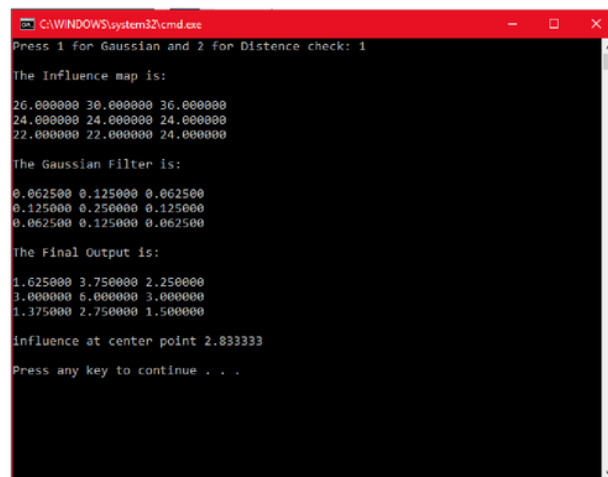
How does the filter and the influence map work together?

In order for the influence map and the Gaussian filter to work together the influence map will need to be bigger than the filter that will be used to multiply against it to get the final results. The first number in the influence portion you are using is multiplied against the first part of the Gaussian filter and added the next number until you have done all the numbers in the grid. This will then produce the final number that you can then plot on a new graph or store as data for later use.

In order to step along the image, you will need to move it by a number of pixels, this is called a stride and the stride is usually kept as 1 but it can be increased to be whatever you want. If the stride is increased you may need to increase the size of the image so that the stride won't exceed the image. This size increase is called padding.

The most common type of convolution matrix is a 2D convolution, usually called conv2D when referenced in papers and factual documents. This common one uses the technique I described above where the filter scans across a larger image to get the data values. [1]

```
1 #include <stdio.h>
2
3 void main()
4 {
5
6     //float infMap[9] = { 32.0,32.0,34.0,26.0,26.0,26.0,18.0,18.0,24.0 };
7     float infMap[9] = { 26.0,30.0,36.0,24.0,24.0,24.0,22.0,22.0,24.0 };
8
9     float gaussianFilter[9] = { 0.0625,0.125,0.0625,0.125,0.25,0.125,0.0625,0.125,0.0625 };
10
11     float finalInfMap[9];
12
13     float infCenterPoint = 0;
14
15     for (int i = 0; i < 9; i++)
16     {
17         finalInfMap[i] = infMap[i] * gaussianFilter[i];
18     }
19
20     for (int i = 0; i < 9; i++)
21     {
22         if (i % 3 == 0 && i != 0)
23             printf("\n");
24         printf("%f ", finalInfMap[i]);
25     }
26
27
28     for (int i = 0; i < 9; i++)
29     {
30         infCenterPoint += finalInfMap[i] + finalInfMap[i+1];
31     }
32
33     infCenterPoint = infCenterPoint / 9;
34
35     printf("\n \nInfluence at centre point %2f\n\n", infCenterPoint);
36
37
38 }
```



```
C:\WINDOWS\system32\cmd.exe
Press 1 for Gaussian and 2 for Distance check: 1

The Influence map is:
26.000000 30.000000 36.000000
24.000000 24.000000 24.000000
22.000000 22.000000 24.000000

The Gaussian Filter is:
0.062500 0.125000 0.062500
0.125000 0.250000 0.125000
0.062500 0.125000 0.062500

The Final Output is:
1.625000 3.750000 2.250000
3.000000 6.000000 3.000000
1.375000 2.750000 1.500000

Influence at center point 2.833333

Press any key to continue . . .
```

In order to get the value for the new pixel in question the influence map in position 0 is multiplied by the predefined Gaussian filter in position 0, its then does the same thing for all pixels in the 3 x 3 grid. The next step is to add up all 9 numbers (3 x 3 grid is 9 individual numbers) and divide it by 9 (again, the amount of numbers in the filter) this will get you the intensity value for the new pixel.

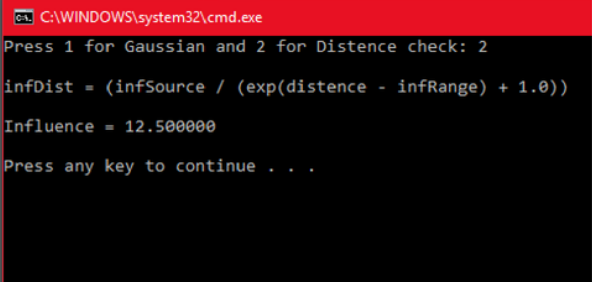
Space and time complexity (Big O Notation) of the Gaussian filter against the influence map

In regards to the complexity of this example, in regards to the big O notation, as the influence map increases in size then the time it will take for the program to run will also increase. The size of the number won't affect it as that only makes the numbers in the maths bigger, not the amount of maths needed. The longevity is down to the amount of numbers in the map. [2]

I feel there are few limitations in this algorithm, one being the amount of numbers in the influence map. I feel if this number is insanely large then it will take the program longer to acquire the maths and therefore longer to render or output the final numbers.

Distance check equation

```
49 typedef struct enemyInfluence {
50     float x;
51     float y;
52 };
53
54 enemyInfluence bad1;
55 enemyInfluence bad2;
56
57 bad1.x = 5.0f;
58 bad1.y = 1.0f;
59
60 bad2.x = 1.0f;
61 bad2.y = 1.0f;
62
63 float AIx = 1.0f;
64 float AIy = 1.0f;
65
66 float infDist = 0;
67
68 float infSource = bad1.x * bad1.y;
69 float infRange = 2.0f;
70 float distanceX = (bad1.x - AIx);
71 float distanceY = (bad1.y - AIy);
72 float distance = (bad1.x - AIx) + (bad1.y - AIy);
73 // printf("DistanceX = %d DistanceY = %d Total distance is: %d\n", distanceX, distanceY, distance);
74
75 //e is the exp function
76 //float e = exp(distance - infRange);
77
78 infDist = (infSource / (exp(distance - infRange) + 1.0));
79 printf("Influence = %f\n\n", infDist);
80 //printf("\ninfSource = %f exp: %f", infSource, e);
81
82
83 }
84
85
86 }
```



```
C:\WINDOWS\system32\cmd.exe
Press 1 for Gaussian and 2 for Distance check: 2

infDist = (infSource / (exp(distance - infRange) + 1.0))

Influence = 12.500000

Press any key to continue . . .
```

What the above picture code does is creates an X and a Y position for two enemy's and the X and Y position for the AI. It then gets its location source as one big number instead of an X and a Y in order to work out the position and distance etc. It will then calculate the distance from the source and the AI. I used this method in order to not need to do a square root calculation every time to avoid it being too computationally expensive. The other method would be to include the square root to get a smaller number but this method, as said

before, wouldn't be worth it due to its computational price. Finally, it will then produce the influence based upon the distance from the AI and the enemy source. To get this answer it will divide influence source by the exponential of the distance minus the influence range with 1 added to that value. I have printed out the answer so I can visually see if this is a value I want to be getting.

Space and time complexity (Big O Notation) of the distance checks between the AI and source

There are a few things that can increase the complexity of the formula and therefore increase the time to finish. One of these things being the number of sources it has to check its distance against. The more distances it has to check, the longer it will take to output the final influence number. Another thing that can increase the time till completion will be the number of steps to get the source, this will get bigger if the distance it has to travel is large and the size of the step it takes are small. Large distance and small steps will take longer to complete than large steps and a small distance. [2]

Implementation

Equation 1 (distance check) would be unlikely to be used in an influence map due to the distance being checked with every pixel again one or more sources, this is extremely computationally expensive and will require a lot of CPU time in order to run this every frame. I have thought of a few ways to improve the efficiency of this approach; one way would be to use the square magnitude instead of the square root to speed up the calculations. Another approach would be to do a distance check between a larger area of pixels and the sources instead of the individual pixels.

Problem

One problem when working with influence maps are the edge boundaries. There are two ways to combat this issue, one being adapt the matrix, this means it will only include in the matrix, 1 less than the edge, if the grid was 10 wide, then it will only include 9 in the calculation. Another way would be to modify the map, what this would entail is adding a border around the outer edges and clamping the values so their numbers are never changed when doing the convolution calculation. [1]

References

- [1] "SAAMA," [Online]. Available: <https://www.saama.com/blog/different-kinds-convolutional-filters/>.
- [2] A. Mohr, "Quantum Computing in Complexity Theory and Theory of Computation".
- [3] F. J. Millington I, *ARTIFICIAL INTELLIGENCE FOR GAMES Second Edition*. .

Bibliography

- [1] P. Avery and S. Louis, *Coevolving Influence Maps for Spatial Team Tactics in a RTS Game*. 2010.
- [2] A. Krontiris, K. E. Bekris, and M. Kapadia, "ACUMEN: Activity-Centric Crowd Authoring Using Influence Maps."

Abstract –

Code for the first influence map tests

```
#include <stdio.h>
#include <math.h>

const int WIDTH = 800;
const int HIGHT = 600;

typedef struct enemyInfluence {
    float x;
    float y;
};

void main(int argc, char **argv)
{

    int choice = 0;

    printf("Press 1 for Gaussian and 2 for Distance check: ");
    scanf("%d", &choice);

    if (choice == 1)
    {
        //float infMap[9] = { 32.0,32.0,34.0,26.0,26.0,26.0,18.0,18.0,24.0 };
        float infMap[9] = { 26.0,30.0,36.0,24.0,24.0,24.0,22.0,22.0,24.0 };

        float gaussianFilter[9] = {
            0.0625,0.125,0.0625,0.125,0.25,0.125,0.0625,0.125,0.0625 };

        float finalInfMap[9];

        float infCenterPoint = 0;

        printf("\nThe Influence map is:\n\n");

        for (int i = 0; i < 9; i++)
        {
            if (i % 3 == 0 && i != 0)
                printf("\n");
            printf("%2f ", infMap[i]);

        }

        printf("\n\n");

        printf("The Gaussian Filter is:\n\n");

        for (int i = 0; i < 9; i++)
        {
            if (i % 3 == 0 && i != 0)
                printf("\n");
            printf("%2f ", gaussianFilter[i]);

        }

    }

}
```

```

    }

    printf("\n\n");

    for (int i = 0; i < 9; i++)
    {
        finalInfMap[i] = infMap[i] * gaussianFilter[i];
    }

    printf("The Final Output is:\n\n");

    for (int i = 0; i < 9; i++)
    {
        if (i % 3 == 0 && i != 0)
            printf("\n");
        printf("%2f ", finalInfMap[i]);

    }

    for (int i = 0; i < 9; i++)
    {
        infCenterPoint += finalInfMap[i] + finalInfMap[i+1];
    }

    infCenterPoint = infCenterPoint / 9;

    printf("\n \ninfluence at center point %2f\n\n", infCenterPoint);
}

if (choice == 2)
{

    enemyInfluence bad1;
    enemyInfluence bad2;

    bad1.x = 5.0f;
    bad1.y = 5.0f;

    bad2.x = 1.0f;
    bad2.y = 1.0f;

    float AIX = 4.0f;
    float AIY = 4.0f;

    float infDist = 0;

    float infSource = bad1.x * bad1.y;
    float infRange = 2.0f;
    float distanceX = (bad1.x - AIX);
    float distanceY = (bad1.y - AIY);
    float distance = (bad1.x - AIX) + (bad1.y - AIY); //Works out the
distance between the 2 points

    //    printf("DistanceX = %d DistanceY = %d Total distance is: %d\n\n",
distanceX, distanceY, distance); // used for testing

    infDist = (infSource / (exp(distance - infRange) + 1.0)); // calculates
the influence distance from the source

```

```

        printf("\ninfDist = (infSource / (exp(distance - infRange) +
1.0))\n\n");
        printf("Influence = %f \n\n", infDist);
        //printf("\ninfSource = %f exp: %f",infSource, e); // Testing the value

    }

}

```

Code for the Graphical interface influence maps

```

/*****
*****
* Influence map example using convolution with a distance check to spread out
influence.
#define GRIDSIZE is used to set map size - here its 50 by 50.
Press space bar to trigger a pass of the filter over the map.
*****
*****/

///// Includes
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "glut.h"
#define GRIDSIZE 50

///// Global Variables
typedef struct rgbtag {
    float r;
    float g;
    float b;
}rgb;

//This struct stores the attributes for each item that needs an influence
typedef struct enemyInfluence {
    int x;
    int y;
    float radius;
    float inf;
};

//X and Y struct for storing the current position
struct vec2 {
    int x, y;
};

//creating the struct names for the "enemy and friendly"
enemyInfluence red;
enemyInfluence blue;

// used later to store current grid position
vec2 position;

//Struct for the AI bot that moves around
struct AI {
    int x;
    int y;
    float radius;
    float inf;
};

```

```

};

//Creation of ai bot
AI ai;

//struct to hold all the tests in one place
struct InfTest
{
    float up, down, left, right;
};

//creation of the test structure
InfTest infTest;

float influence = 0;

rgb grid[GRIDSIZE*GRIDSIZE];
rgb gridtemp[GRIDSIZE*GRIDSIZE];
//function prototypes
void init(void);
void display(void);
void drawBox(float w, float h, float l);
void reshape(int w, int h);

void init()
// Initialize
//*****
// desc: initializes OpenGL
{
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);          // clears colour buffer
    glEnable(GL_DEPTH_TEST);                       // hidden surface
removal
    glEnable(GL_CULL_FACE);                         // do not
calculate inside of poly's
    glFrontFace(GL_CCW);                           // counter clock-wise
polygons are out

    for (int i = 0; i<GRIDSIZE; i++) {
        for (int j = 0; j<GRIDSIZE; j++) {

            grid[i + (j*GRIDSIZE)].r = 0;
            grid[i + (j*GRIDSIZE)].g = 0;
            grid[i + (j*GRIDSIZE)].b = 0;
        }
    }

    //The next groups of code declares all the variables with there own numbers
    position.x = 0;
    position.y = 0;
    red.x = 16.0f;
    red.y = 16.0f;
    red.radius = 1.0f;
    red.inf = 2.0f;

    blue.x = 26.0f;
    blue.y = 40.0f;
    blue.radius = 4.0f;
    blue.inf = 5.444f;

    ai.x = 4.0f;
    ai.y = 4.0f;
    ai.radius = 1.0;

```

```

        ai.inf = 1.0;
    }

    void drawBox(float w, float h, float l)
    //drawBox
    //*****
    //This function first translates a further x,y,z units then draws a box of
    //width w, height h, length l
    {
        glPushMatrix();          //save modelview
        w = w / 2.0; h = h / 2.0; l = l / 2.0; //adjust values so centre is in middle
    of box
                                                                    //draw faces
    of box
        glBegin(GL_POLYGON);
        glVertex3f(w, l, h);
        glVertex3f(w, l, -h);
        glVertex3f(-w, l, -h);
        glVertex3f(-w, l, h);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex3f(w, l, h);
        glVertex3f(-w, l, h);
        glVertex3f(-w, -l, h);
        glVertex3f(w, -l, h);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex3f(w, l, h);
        glVertex3f(w, -l, h);
        glVertex3f(w, -l, -h);
        glVertex3f(w, l, -h);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex3f(-w, l, h);
        glVertex3f(-w, l, -h);
        glVertex3f(-w, -l, -h);
        glVertex3f(-w, -l, h);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex3f(-w, -l, h);
        glVertex3f(-w, -l, -h);
        glVertex3f(w, -l, -h);
        glVertex3f(w, -l, h);
        glEnd();
        glBegin(GL_POLYGON);
        glVertex3f(w, -l, -h);
        glVertex3f(-w, -l, -h);
        glVertex3f(-w, l, -h);
        glVertex3f(w, l, -h);
        glEnd();
        glPopMatrix(); //restore previous modelview matrix so leaving things as you
found them
    }

    // display
    //*****
    // desc: handles drawing of scene
    void display()
    {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); // clear screen
        and depth buffer
    }

```

```

glLoadIdentity();
// reset modelview matrix

gluLookAt(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0); //reset camera position
glColor3f(0, 0, 0); //set colour
glTranslatef(0.5, 0.5, 0.5); //move by x,y and z units
//tests grid by applying
greyscale values to each position
//
for (int i = 0; i<GRIDSIZE; i++) {
    for (int j = 0; j<GRIDSIZE; j++) {
        if (grid[i + (j*GRIDSIZE)].r>grid[i + (j*GRIDSIZE)].b)
        {
            glColor3f(grid[i + (j*GRIDSIZE)].r, 0, 0);
        }
        else
        {
            glColor3f(0, 0, grid[i + (j*GRIDSIZE)].b); //set colour
        }
    }

    //This sets the grid to be green wherever the "ai" is currently
    located
    if (i == ai.x && j == ai.y)
    {
        glColor3f(0, grid[i + (j*GRIDSIZE)].g, 0);
    }

    //sets the borders to be red to stop the ai from leaving
    if (i == 0 || j == 0 || i == GRIDSIZE - 1 || j == GRIDSIZE - 1)
    {
        glColor3f(1, 0, 0);
    }

    glPushMatrix(); //save current position/orientation, etc.
    glTranslatef(i, j, 0);

    drawBox(1, 1, 1); //draw
    glPopMatrix(); //restore previous "current" position/orientation
    etc.
}

glFlush(); //force drawing

glutSwapBuffers(); // bring backbuffer to foreground
//any errors then display error
codes
GLenum errCode;
const GLubyte *errString;
if ((errCode = glGetError()) != GL_NO_ERROR)
{
    errString = gluErrorString(errCode);
    fprintf(stderr, "OpenGL error : %s\n", errString);
}
}
void keyboardread(unsigned char key, int x, int y)
{

```

```

//Create a variable to hold the distance calculation
float distance = 0;

switch (key) {
case 'u':

    break;
case ' ':

    //used for testing to make sure the numbers looks how they were supposed
    too
    //printf("%d\n", ai.x + ((ai.y + 1) *GRIDSIZE));
    //printf("%d\n", ai.x + ((ai.y - 1) *GRIDSIZE));
    //printf("%d\n", ai.x - 1 + (ai.y *GRIDSIZE));
    //printf("%d\n", ai.x + 1 + (ai.y *GRIDSIZE));

    //Sets the test variables to be the red value of the grid
    infTest.up = grid[ai.x + ((ai.y + 1) *GRIDSIZE)].b - grid[ai.x + ((ai.y
+ 1) *GRIDSIZE)].r;
    infTest.down = grid[ai.x + ((ai.y - 1) *GRIDSIZE)].b - grid[ai.x +
((ai.y - 1) *GRIDSIZE)].r;
    infTest.left = grid[(ai.x - 1) + (ai.y *GRIDSIZE)].b - grid[(ai.x - 1) +
(ai.y *GRIDSIZE)].r;
    infTest.right = grid[(ai.x + 1) + (ai.y *GRIDSIZE)].b - grid[(ai.x + 1)
+ (ai.y *GRIDSIZE)].r;

    //again, used to test the numbers are correct
    //printf("%f, %f, %f, %f\n", infTest.up, infTest.down, infTest.left,
infTest.right);

    //This will test all the directions to see which value is lowest
    //Then will update the ai position to move in that direction
    if ((infTest.up >= infTest.down) && (infTest.up >= infTest.left) &&
(infTest.up >= infTest.right))
    {
        ai.y += 1;
        ai.x = ai.x;
    }

    if ((infTest.down >= infTest.up) && (infTest.down >= infTest.left) &&
(infTest.down >= infTest.right))
    {
        ai.y -= 1;
        ai.x = ai.x;
    }

    if ((infTest.left >= infTest.up) && (infTest.left >= infTest.down) &&
(infTest.left >= infTest.right))
    {
        ai.x -= 1;
        ai.y = ai.y;
    }

    if ((infTest.right >= infTest.up) && (infTest.right >= infTest.down) &&
(infTest.right >= infTest.left))
    {
        ai.x += 1;
        ai.y = ai.y;
    }

    for (int i = 0; i < GRIDSIZE; i++) {
        for (int j = 0; j < GRIDSIZE; j++) {

```

```

        //Sets the current position
        position.x = i;
        position.y = j;

        //This group does the calculations for the red circle
        distance = sqrt(((red.x - position.x)*(red.x -
position.x)) + ((red.y - position.y)*(red.y - position.y)));
        influence = (red.inf / (exp(distance - red.radius) +
1.0)); //Influence calculation for the Red circle
        grid[i + j * GRIDSIZE].r = influence; //sets the red colour
value to be the influence

        //This group does the calculations for the blue circle
        distance = sqrt(((blue.x - position.x)*(blue.x -
position.x)) + ((blue.y - position.y)*(blue.y - position.y)));
        influence = (blue.inf / (exp(distance - blue.radius) +
1.0));
        grid[i + j * GRIDSIZE].b = influence;

        //This group does the calculations for the green AI
        distance = sqrt(((ai.x - position.x)*(ai.x - position.x))
+ ((ai.y - position.y)*(ai.y - position.y)));
        influence = (ai.inf / (exp(distance - ai.radius) + 1.0));
        grid[i + j * GRIDSIZE].g = influence;

        //This check is to make sure the AI doesn't ignore the
boundary's of the screen
        if (i == 0 || j == 0 || i == GRIDSIZE - 1 || j == GRIDSIZE
- 1)
        {
            grid[i + j * GRIDSIZE].r = 1.0;
        }
    }

    red.x = red.x + 1; // This moves the red influence

    glutPostRedisplay();
    break;
}

void reshape(int w, int h)
//reshape
//*****
//OpenGL reshape callback is called when the window is resized
{
    glViewport(0, 0, (GLsizei)w, (GLsizei)h); //respecify size of view port to
match new w and h
    glMatrixMode(GL_PROJECTION); //go into projection mode
    glLoadIdentity(); //reset projection matrix
    glOrtho(0.0, GRIDSIZE, 0.0, GRIDSIZE, 0.0, 100.0); //defines orthoganal
projection (parrallel

    // viewing volume

    // window limits of display

    // left =0.0, right =2.0,

```

```

        // bottom = 0.0, top = 2.0

        // near =0.0, far = 100.0

        //gluPerspective(60.0,(GLfloat)w/(GLfloat)h,1.0,200.0); //redefine
projection matrix to match changes in w and h
        glMatrixMode(GL_MODELVIEW); //switch back to modelview mode
        glLoadIdentity(); //reset modelview matrix
        gluLookAt(0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); //reset camera position
    }

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(1000, 200);//I changes these values to be more central
for me
    glutCreateWindow(argv[0]);
    init();
    glutDisplayFunc(display);
    glutReshapeFunc(reshape);
    glutKeyboardFunc(keyboardread);
    glutMainLoop();
    return 0;
}

```

Advanced AI and Machine Learning Assignment Write Up

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

52/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

GRADING FORM: SMU GENERIC_GJ

ZACH PORT-LINCOLN

52

RESEARCH-INFORMED LITERATURE

 5/10 Ref to Quantum computing - odd - if it has a good section on complexity in general then ok - but most ADS texts have excellent accounts of Complexity. Inf distance calculations programmed but flawed implementation.

5

KNOWLEDGE AND UNDERSTANDING

 12/25 Essential ideas covered and concepts discussed. Some issues with the discussion/implementation of the distance calc. A separate example and illustration would help enormously here.

12

ANALYSIS

 9/20 Calculations:- Convolution calculated via program - how do you know its correct? Apart from the division by 9 it is fine. Issues with choice of values for your linear distance calcs. Flawed. Algorithms discussed and analysed and big O notation and complexity mentioned but not really applied with understanding here.

9

PRACTICAL APPLICATION AND DEP.

 24/40 Doesn't load influence map. Influence calculated each time - but see comments regarding

23

correctness of the calculation in the Analysis section. AI doesn't contribute to/update the influence map. Some good work here but this is not reflected in the report.

SKILLS FOR PROFESSIONAL PRAC.



3/5 Table of contents would help. Report structured. Refs included and bibliography - not many refs though!.

3