**Seoul National University**

**College of Engineering**

**Department of Naval Architecture and Ocean Engineering**

**1, Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea**

Spring 2023

컴퓨터 이용 선박 설계

**(Computer-Aided Ship Design)**

**PA #1**

|  |  |
| --- | --- |
| Instructor name | 김태완 |
| Student name | 장민준 |
| Department | 조선해양공학과 |
| Student ID | 2019-10738 |
| Submission date | 2023-03-28 |
| Grade |  |

목차

1. 서론
   1. Affine transformation
2. 본론
   1. 파일 입력
   2. P1
   3. P2
   4. P3
3. 결론
4. Reference
5. 서론
   1. Affine transformation

Affine transformation이란, 평면 위의 도형을 회전, 확장 및 축소, 평행이동 시킬 수 있는 linear transformation이다.

임의의 array의 각 row마다 새로운 점으로 변환하려는 Affine transformation은 회전 변환과 평행이동을 합쳐서 만들 수 있다

임의의 점 를 로 변환하는 식을 작성해보면 다음과 같다

이러한 변환을 실행하는 array를 작성해보면 다음과 같다

임의의 점들을 나타내는 array를 하나 가정하겠다  
이 때, 3행의 상수들은 평행이동을 반영하기 위한 값이다.

위 array는 columm에 각 점들의 x,y 좌표를 저장하고 있다.

Pre의 각 columm vector들은 연산을 통해 Post의 column vector로 변환될 수 있다, 연산을 진행하면

위와 같이 새로운 array로 전환할 수 있다.  
변환을 실행하려는 A를 구하기 위해서는, Least square method를 통해 구할 수 있다

위와 같은 방식으로 Affine Matrix를 추정할 수 있다.[[1]](#footnote-1)

1. 본론
   1. 파일 입력 및 클래스 내부 함수

파일 입력을 받아오기 위해, C 언어 라이브러리 내부의 함수 fopen, fprintf, fclose 등을 사용해야 하기 때문에, 시스템 헤더 파일 cstudio를 불러온다. 이외에도 iostream과 linear algebra library인 Eigen을 불러온다.

#include <iostream>

#include <cstdio>

#include <Eigen/Dense>

세가지 텍스트 파일을 입력받을 것이고, 파일의 구성이 다 다르기 때문에, 클래스에 input1, input2, input3 3개의 함수를 포함시켰다. 그리고 데이터를 클래스에 저장하기 위해 private member와 생성자를 다음과 같이 추가했다

**class** Transformation {

**private**:

MatrixXd pre\_image\_points1;

MatrixXd post\_image\_points1;

MatrixXd pre\_image\_points2;

MatrixXd post\_image\_points2;

Vector<**float**, 3> circle\_elements;

**public**:

Transformation() {

pre\_image\_points1 = MatrixXd(3, 2);

post\_image\_points1 = MatrixXd(3, 2);

pre\_image\_points2 = MatrixXd(4, 2);

post\_image\_points2 = MatrixXd(4, 2);

circle\_elements = Vector<**float**, 3>(3);

}

다음은 input1 함수의 코드이다

**void** input1(FILE\* fp) {

**char** buf[256];

**float** num;

fgets(buf, 100, fp); // P1

fgets(buf, 100, fp); // "pre-image points"

pre\_image\_points1 = MatrixXd::Zero(3, 2);

**for** (**int** i = 0; i < 3; i++) {

**for**(**int** j = 0; j < 2; j++){

**if**(fscanf(fp, "%f", &num) == 1){

pre\_image\_points1(i,j) = num;}

}

}

fgets(buf, 100, fp);//행바꿈

fgets(buf, 100, fp);//행바꿈

fgets(buf, 100, fp);// "post-image points"

post\_image\_points1 = MatrixXd::Zero(3, 2);

**for** (**int** i = 0; i < 3; i++) {

**for**(**int** j = 0; j < 2; j++){

**if**(fscanf(fp, "%f", &num) == 1){

post\_image\_points1(i,j) = num;}

}

}

}

파일 입력을 받을 때, 텍스트 파일 내부에 관련없는 정보를 처리하기 위해 buf를 선언했고, 무시할 데이터들을 위와 같이 처리했다.

이후, 받아야 할 데이터를 반복문을 통해 post\_image\_points의 각 원소에 저장을 받았다.

input2 함수의 구성은 input1과 동일하므로 생략하겠다.

Input3 함수는 배열 대신 벡터를 받아 동일하게 처리하였다.

그리고, 결과창에 점들과 array를 출력할 함수를 구현했다.

**void** print\_pre\_post1() {

cout << "Pre-image points:\n" << pre\_image\_points1 << endl;

cout << "Post-image points:\n" << post\_image\_points1 << endl;

}

**void** print\_pre\_post2() {

cout << "Pre-image points:\n" << pre\_image\_points2 << endl;

cout << "Post-image points:\n" << post\_image\_points2 << endl;

}

**void** print(**const** MatrixXd &array){

**for**(**int** i = 0; i < array.rows(); i++){

**for**(**int** j = 0; j < array.cols(); j++){

cout << array(i, j) << " ";

}

cout << endl;

}

}

또한, private member들은 클래스 외부에서 호출이 불가능해서, 이들을 불러올 함수들을 구현했다.

MatrixXd get\_private\_pre1(){

**return** pre\_image\_points1;

}

MatrixXd get\_private\_post1(){

**return** post\_image\_points1;

}

MatrixXd get\_private\_pre2(){

**return** pre\_image\_points2;

}

MatrixXd get\_private\_post2(){

**return** post\_image\_points2;

}

Vector<**float**, 3> get\_private\_circle\_elelments(){

**return** circle\_elements;

}

데이터를 전치시키고, 열을 추가해주는 함수를 구현했다.

MatrixXd Transverse\_pre1\_3X3() **const** {

MatrixXd pre = MatrixXd::Ones(3, 3);

**for**(**int** i = 0; i < 3; i++) {

**for**(**int** j = 0; j < 2; j++) {

pre(j, i) = pre\_image\_points1(i, j);

}

}

**return** pre;

}

MatrixXd Transverse\_post1\_3X3() **const** {

MatrixXd post = MatrixXd::Ones(3, 3);

**for**(**int** i = 0; i < 3; i++) {

**for**(**int** j = 0; j < 2; j++) {

post(j, i) = post\_image\_points1(i, j);

}

}

**return** post;

}

MatrixXd Transverse\_pre2\_3X4() **const** {

MatrixXd pre2 = MatrixXd::Ones(3, 4);

**for**(**int** i = 0; i < 4; i++) {

**for**(**int** j = 0; j < 2; j++) {

pre2(j, i) = pre\_image\_points2(i, j);

}

}

**return** pre2;

}

MatrixXd Transverse\_post2\_3X4() **const** {

MatrixXd post2 = MatrixXd::Ones(3, 4);

**for**(**int** i = 0; i < 4; i++) {

**for**(**int** j = 0; j < 2; j++) {

post2(j, i) = post\_image\_points2(i, j);

}

}

**return** post2;

}

Output 함수들은 각 문제에서 설명하겠다.

* 1. P1

Transformation IP1;

Input\_P1을 입력받기 위해 객체 IP1을 선언했다

FILE\* fp1 = fopen("/Users/minjun/Desktop/2023-1/컴선설/Programming Assignment#1/Input\_P1.txt", "r");

IP1.input1(fp1);

이후, IP1에 데이터를 전달하기 위해 파일 객체 fp1를 IP1의 input1함수에 입력한다.

IP1 객체에 데이터가 전달이 되었고, 이후 데이터를 확인하기 위해 다음 함수를 사용한다

IP1.print\_pre\_post1();

확인 후 파일을 닫아주고

fclose(fp1);

연산을 위해 다음 코드를 입력해 데이터들을 전치시키고, 1로 이루어진 행을 추가시킨다

MatrixXd pre1 = IP1.Transverse\_pre1\_3X3();

MatrixXd post1 = IP1.Transverse\_post1\_3X3();

cout << "pre1 \n";

IP1.print(pre1);

cout << "post1 \n";

IP1.print(post1);

데이터를 전치시켰고, 이후 Affine 행렬을 계산하기 위해 다음과 같이 코드를 입력한다.

MatrixXd Affine\_IP1 = post1\*pre1.inverse();

cout << "affine 변환 행렬 IP1 : \n";

MatrixXd Affine\_pre1 =(Affine\_IP1\*pre1).transpose();

IP1.print(Affine\_IP1);

Affine\_IP1의 값은 다음과 같다

이후 최종으로, 출력함수를 통해 파일을 만든다.

IP1.print\_Output1(Affined\_pre1);

출력함수는 다음과 같이 구성했다, 출력함수가 받는 변수는 Affine 행렬이다

void print\_Output1(MatrixXd& affine\_pre1) {

FILE \*fp;

fp = fopen("/Users/minjun/Desktop/2023-1/컴선설/Programming Assignment#1/OutPut1.ps", "w"); // 파일을 쓰기 모드로 열기

if (fp == NULL) {

printf("파일 열기 실패\n");

return;

}

// PostScript 코드 작성

fprintf(fp, "%%!PS\n"); // PostScript 파일 시작

fprintf(fp, "2 setlinewidth\n"); // 선 굵기 설정

fprintf(fp, "1 setlinecap\n"); // 끝 점 모양 설정

fprintf(fp, "1 setlinejoin\n"); // 교차점 처리 모양 설정

fprintf(fp, "1 0 0 setrgbcolor\n");

//pre\_image\_points

fprintf(fp, "%f %f moveto \n", 50\*pre\_image\_points1(0,0), 50\*pre\_image\_points1(0,1));

for(int i = 1; i < 3; i++){

float x\_ps = 50\*pre\_image\_points1(i,0);

float y\_ps = 50\*pre\_image\_points1(i,1);

fprintf(fp, "%f %f lineto \n", x\_ps, y\_ps);

}

fprintf(fp, "closepath\n");

fprintf(fp, "stroke \n");

//post\_image\_points

fprintf(fp, "newpath\n");

fprintf(fp, "2 setlinewidth\n"); // 선 굵기 설정

fprintf(fp, "1 setlinecap\n"); // 끝 점 모양 설정

fprintf(fp, "1 setlinejoin\n"); // 교차점 처리 모양 설정

fprintf(fp, "0 1 0 setrgbcolor\n");// post point 그리기

fprintf(fp, "%f %f moveto \n", 50\*post\_image\_points1(0,0), 50\*post\_image\_points1(0,1));

for(int i = 1; i < 3; i++){

float x\_ps = 50\*post\_image\_points1(i,0);

float y\_ps = 50\*post\_image\_points1(i,1);

fprintf(fp, "%f %f lineto \n", x\_ps, y\_ps);

}

fprintf(fp, "closepath\n");

fprintf(fp, "stroke \n");

//affine transformation

fprintf(fp, "newpath\n");

fprintf(fp, "2 setlinewidth\n"); // 선 굵기 설정

fprintf(fp, "1 setlinecap\n"); // 끝 점 모양 설정

fprintf(fp, "1 setlinejoin\n"); // 교차점 처리 모양 설정

fprintf(fp, "0 0 1 setrgbcolor\n");// post point 그리기

fprintf(fp, "%f %f moveto \n", 50\*affine\_pre1(0,0)+10, 50\*affine\_pre1(0,1)+10); // 동일하게 겹칠 것을 고려해서 (10,10) 평행이동

for(int i = 1; i < 3; i++){

float x\_ps = 50\*affine\_pre1(i,0)+10;

float y\_ps = 50\*affine\_pre1(i,1)+10;

fprintf(fp, "%f %f lineto \n", x\_ps, y\_ps);

}

fprintf(fp, "closepath\n");

fprintf(fp, "stroke \n");

fprintf(fp, "showpage\n"); // 페이지 출력

fclose(fp); // 파일 닫기

}

위와 같이 객체에서 데이터를 받아와서 Pre-Points, Post-Points, 그리고 Affine Matrix로 변환된 삼각형을 Postscript 파일로 작성한다. 여기서는 Affine Matrix로 변환된 삼각형을 확인하기 위해 (10,10)만큼 평행이동을 시켰다.

차트이(가) 표시된 사진

자동 생성된 설명

* 1. P2

전체적으로 구성은 P1과 거의 동일해, 차이점만 설명하겠다.

P2의 점들은 서론에서 언급한 Least square method를 통해 Affine matrix를 근사해야 한다.

MatrixXd x = IP2.get\_private\_pre2();

cout << "x \n";

IP2.print(x);

MatrixXd y =IP2.get\_private\_post2();

cout << "y \n";

위 코드로 private member를 가져와 새롭게 array를 만들고, 이 데이터를 이용해 다음과 같이 연산해 Affine Matrix를 근사한다.

MatrixXd AffineIP2 = y.transpose()\*pre2.transpose()\*(pre2\*pre2.transpose()).inverse();

cout << "affine 변환 행렬 IP2 : \n";

IP2.print(AffineIP2);

MatrixXd Affined\_pre2 = (AffineIP2\*pre2).transpose();

IP2.print\_Output2(Affined\_pre2);

IP2.print(Affined\_pre2);  
이후에 Affined\_pre2를 통해 변환된 도형을 print\_Output2를 통해 출력하고, Affined\_Pre2를 출력한다.

차트이(가) 표시된 사진

자동 생성된 설명

* 1. P3

입력받은 벡터를 바탕으로 print\_Output3 함수를 구현한다

**void** print\_Output3() {

FILE \*fp;

fp = fopen("/Users/minjun/Desktop/2023-1/컴선설/Programming Assignment#1/OutPut3.ps", "w"); // 파일을 쓰기 모드로 열기

**if** (fp == **NULL**) {

printf("파일 열기 실패\n");

**return**;

}

// PostScript 코드 작성

fprintf(fp, "%%!PS\n"); // PostScript 파일 시작

fprintf(fp, "newpath\n");

fprintf(fp, "%f %f %f 0 360 arc\n",100\*circle\_elements(0),100\*circle\_elements(1),100\*circle\_elements(2));

fprintf(fp, "stroke\n");

fprintf(fp, "%f %f %f 0 360 arc\n",100\*circle\_elements(0),100\*circle\_elements(1),100\*circle\_elements(2));

fprintf(fp, "closepath\n");

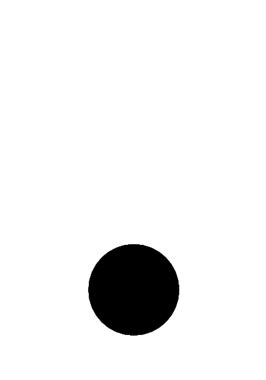
fprintf(fp, "0 0 0 setrgbcolor\n");

fprintf(fp, "fill\n");

fprintf(fp, "showpage\n"); // 페이지 출력

fclose(fp); // 파일 닫기

}



1. 결론

P1의 경우, 세 점을 array로 입력받아 이를 새로운 세 점으로 변환하는 affine matrix를 추정하는 것이다. 회전, 확장변환 및 평행이동이므로 2 -dimensional일 때, 변수가 총 6개이므로, independent한 식 6개가 있어야 determined될 수 있고, 실제로 fully determined되었다.

P2의 경우에는 점 8개로 6개의 변수와 8개의 식이 구성되어 over-determined된다. 이 경우에는 least square method를 통해 근사해야 한다. 이를 통해 근사해서 찾아 낸 Affine Matrix를 이용해 Postscript를 통해 구현하여 시각적으로 얼마나 근사한지 확인할 수 있었다.

전체적으로 연산 과정을 C++을 통해 구현할 수 있었고, visualization을 Postscript를 통해 구현할 수 있었다. 이는 향후 다양한 현상을 해석하는 계산모델을 적용하고, 시각화 할 수 있음을 시사한다.

1. Reference
2. Computer-Aid ship design 수업자료 (김태완)
3. [Affine Transformation -- from Wolfram MathWorld](https://mathworld.wolfram.com/AffineTransformation.html)
4. [REAKWON :: [C언어] 출력 형식(format) 총정리 (Feat. sprintf, fprintf) - 일정한 간격으로 문자열 출력 예제 까지 (tistory.com)](https://reakwon.tistory.com/169)

1. [LN\_3\_Least\_Squares\_2023\_03\_27.pdf: 컴퓨터이용선박설계 (001) (snu.ac.kr)](https://myetl.snu.ac.kr/courses/240353/files/2092759?module_item_id=646919) [↑](#footnote-ref-1)