

2014학년도

경기과학고등학교 기초R&E 결과보고서

## 3차원 택시 거리계에서의 이차곡면

### Quadratic Surfaces on 3-Dimensional Taxicab Geometry

2014. 11. 28

연구참여자 : 박승원(psw1153@naver.com)

이상현(evenharder@naver.com)

이호진(hojini1117@naver.com)

지도교사: 이창석

과학영재학교 경기과학고등학교

# 3차원 택시 거리계에서의 이차곡면

## Quadratic Surfaces on 3-Dimensional Taxicab Geometry

### 초 록

보통 우리는 거리라는 말을 들으면 주로 Euclidean Geometry에서의 거리를 떠올린다. 그러나 다른 기본적인 거리로 Taxicab Geometry가 존재한다. Taxicab Geometry는 2차원에서는 충분히 연구되었지만, 아직 그 이상의 차원에서는 그 내용이 미흡하다.

이 보고서에서는 Taxicab Geometry를 3차원 이상의 Cartesian Coordinate에 적용시켜 Sphere, Ellipsoid, Hyperboloid, Paraboloid 와 같은 이차곡면들의 모습을 Geogebra 5, GSP 5, Cabri 3d, Mathematica 9, Dev-C++와 같은 다양한 소프트웨어를 이용하여 구현하고, 구현된 도형이 어떤 성질을 만족하는지 탐구하였다.

# I. Introduction

## 1.1 평면상의 유클리드 거리함수

평면 위의 두 점 사이의 거리를 나타내는데 가장 많이 쓰이는 표현은 두 점을 이은 선분의 길이로 정의되는 유클리드 거리함수이다. 이 거리 개념은 직관적으로 이해가 쉽기 때문에 수학에서 가장 많이 활용되고 있다.

직교좌표계에서 두 점  $P(x_1, y_1), Q(x_2, y_2)$ 를 잡으면, 이 두 점 사이의 유클리드 거리함수는 다음과 같이 정의된다.

정의 1.

$$d(P, Q) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

## 1.2 평면상의 택시거리함수

유클리드 거리함수는 수학적인 접근이 쉬운 반면에, 현실상의 한 지점에서 다른 한 지점으로 이동할 때 유용한 방법은 아니라고 할 수 있다. 이에 현대 도시에서 택시를 타고 이동하는 경로 개념을 도입한 실용적 거리함수인 택시거리가 등장하였다. 택시거리는 대부분이 직각으로 이루어져 있는 현대 도시의 도로에서의 움직임과 연관이 깊어 유클리드 거리함수에 비해 실제적이다. 위의 두 점 와 사이의 택시거리함수는 다음과 같이 정의된다.

정의 2.

$$d_T(P, Q) = |x_1 - x_2| + |y_1 - y_2|$$

## 1.3 2차원 택시기하에서의 선행연구

선행연구[1]에서는 2차원 평면 위의 택시거리계에 의한 원, 타원, 쌍곡선, 포물선의 다양한 개형을 관찰하고 그를 분류하였다. 이 개형들은 크게 Figure 1,2,3과 같이 나타내어진다.

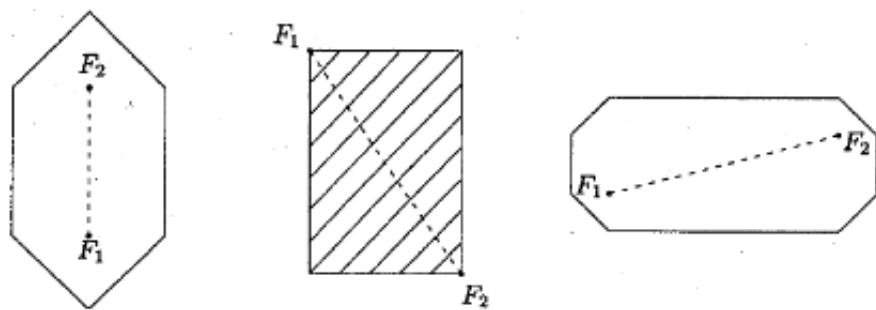


Figure 1. 택시 타원

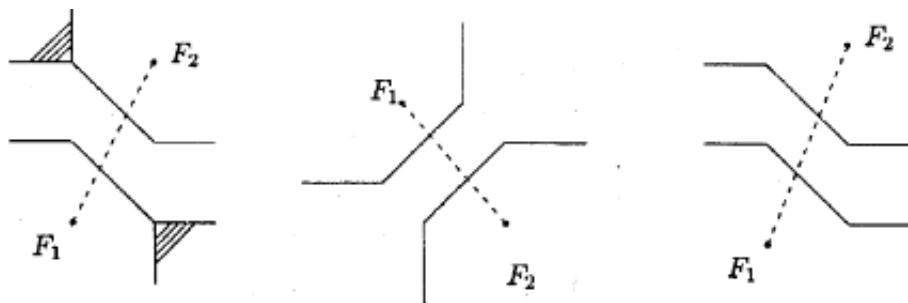


Figure 2. 택시 쌍곡선

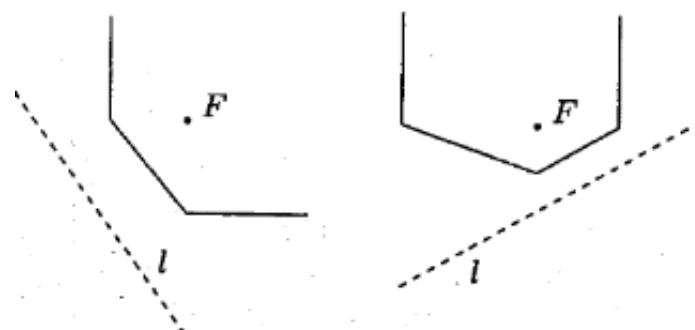


Figure 3. 택시 포물선

## II. Definintion

### 2.1 n차원 택시 거리계의 정의

두 점  $A(p_1, p_2, \dots, p_n), B(q_1, q_2, \dots, q_n)$ 에 대하여 두 점의 택시 거리는 다음과 같다.

정의 3.

$$d_T(A, B) = \sum_{i=1}^n |p_i - q_i|$$

### 2.2 n차원 택시 거리계의 존재성

거리함수의 3가지 조건은 다음과 같다.

$$\textcircled{1} \quad d(A, B) = 0 \Leftrightarrow A = B$$

$$\textcircled{2} \quad d(A, B) = d(B, A)$$

$$\textcircled{3} \quad d(A, C) \leq d(A, B) + d(B, C) \text{ (Let } C(r_1, r_2, \dots, r_n))$$

이 조건을 n차원 택시 거리계가 만족시킨다는 것을 증명하였다.

*Proof.*    \textcircled{1}  $d(A, B) = 0 \Leftrightarrow A = B$

$$\begin{aligned} & \forall_{1 \leq i \leq n, i \in \mathbb{N}} |p_i - q_i| \geq 0, \\ & \sum_{i=1}^n |p_i - q_i| = 0. \\ \therefore & \forall_{1 \leq i \leq n, i \in \mathbb{N}} |p_i - q_i| = 0. \therefore A = B. \end{aligned}$$

$$\textcircled{2} \quad d(A, B) = d(B, A)$$

$$\begin{aligned} & \forall_{1 \leq i \leq n, i \in \mathbb{N}} |p_i - q_i| = |q_i - p_i|. \\ & \sum_{i=1}^n |p_i - q_i| = \sum_{i=1}^n |q_i - p_i|. \\ \therefore & d(A, B) = d(B, A) \end{aligned}$$

$$\textcircled{3} \quad d(A, C) \leq d(A, B) + d(B, C) \text{ (Let } C(r_1, r_2, \dots, r_n))$$

$$\begin{aligned} \forall_{1 \leq i \leq n, i \in \mathbb{N}} |p_i - q_i| &\geq |p_i - r_i| + |r_i - q_i|. \\ \sum_{i=1}^n |p_i - q_i| &\geq \sum_{i=1}^n |p_i - r_i| + \sum_{i=1}^n |r_i - q_i|. \\ \therefore d(A, C) &\geq d(A, B) + d(B, C) \end{aligned}$$

□

## 2.3 점과 평면 사이의 거리

임의의 평면  $ax + by + cz + d = 0$ 와 점  $(p, q, r)$ 을 생각하자. 평면 위의 임의의 점  $(k, l, m)$ 에 대해  $ak + bl + cm + d = 0$ .

여기에서  $distance = |k-p| + |l-q| + |m-r|$ 의 최소인  $k, l, m$ 을  $p, q, r$ 의 식으로 구하면 된다. 그에 의한 점과 평면 사이의 거리는 다음과 같다.

**정의 4.**

$$distance = \frac{|ap + bq + cr + d|}{\max(|a|, |b|, |c|)}$$

*Proof.*  $ak + bl + cm + d = 0$  이다.  $ap + bq + cr + d = K$ 라 하자.

$a(p-k) + b(q-l) + c(r-m) = K$ 로 일정하고,

$Distance = \min(|p-k| + |q-l| + |r-m|)$ 이다.

WLOG  $|a| \geq |b| \geq |c|$ .

$p - k = \alpha, q - l = \beta, r - m = \gamma$  라 하자.

$$a\alpha + b\beta + c\gamma = K, \alpha + \frac{b}{a}\beta + \frac{c}{a}\gamma = \frac{K}{a}.$$

여기에서 삼각부등식  $|x+y| \leq |x| + |y|, |\frac{b}{a}| \leq 1, |\frac{c}{a}| \leq 1$  임을 이용하면

$$|\alpha| + |\beta| + |\gamma| \geq |\alpha| + |\frac{b}{a}\beta| + |\frac{c}{a}\gamma| \geq |\alpha + \frac{b}{a}\beta + \frac{c}{a}\gamma| = \frac{K}{a}.$$

□

## 2.4 택시 이차곡면의 정의

3차원 택시 거리계에서의 이차곡면은 유클리드 거리계에서와 동일하게, 한 점의 두 초점 또는 준선(준평면)으로부터의 거리 사이의 관계로 정의된다.

**정의 5.**

택시 구(Taxicab Sphere)

3차원 택시 거리계에서 공간상의 한 점으로부터의 거리가 일정한 점들의 집합

택시 타원체(Taxicab Ellipsoid)

3차원 택시 거리계에서 공간상의 두 초점으로부터의 거리의 합이 일정한 점들의 집합

택시 쌍곡면(Taxicab Hyperboloid)

3차원 택시 거리계에서 공간상의 두 초점으로부터의 거리의 차가 일정한 점들의 집합

택시 포물면(Taxicab Paraboloid)

3차원 택시 거리계에서 공간상의 한 초점과 평면으로부터의 거리가 같은 점들의 집합

### III. Quadratic Surfaces on 3d Taxicab Geometry

이 장에서는 3차원 택시 거리계에서의 이차곡면에 대해 다룰 것이다.

다룰 이차곡면은 구(Sphere), 타원체(Ellipsoid), 쌍곡면(Hyperboloid), 포물면(Paraboloid)이고, 각각의 이차곡면을 수학 소프트웨어 Geogebra 5, GSP 5, Cabri 3d, Mathematica 9과 통합 개발 환경 소프트웨어(IDE - Integrated Development Environments) Dev C++을 사용하여 이 곡면들을 구현해 볼 것이다.

#### 3.1 Taxicab Sphere

3차원 택시 거리계에서 중심이  $(0, 0, 0)$ 이고, 그로부터 거리가  $r$ 인 점들의 집합은 Figure 1과 같은 정팔면체이다. 이를 도형의 방정식으로 나타내면 다음과 같이 8개의 평면의 부분집합으로 나타내어진다.

점 $A(x, y, z)$ 의 위치	방정식
$x \geq 0, y \geq 0, z \geq 0$	$x + y + z = r$
$x < 0, y \geq 0, z \geq 0$	$-x + y + z = r$
$x \geq 0, y < 0, z \geq 0$	$x - y + z = r$
$x < 0, y < 0, z \geq 0$	$-x - y + z = r$
$x \geq 0, y \geq 0, z < 0$	$x + y - z = r$
$x < 0, y \geq 0, z < 0$	$-x + y - z = r$
$x \geq 0, y < 0, z < 0$	$x - y - z = r$
$x < 0, y < 0, z < 0$	$-x - y - z = r$

이와 같이 택시 구는 다음과 같은 방정식의 형태로 나타낼 수 있다.

$$|x| + |y| + |z| = r$$

#### 3.2 Taxicab Ellipsoid - Cabri 3d

택시 타원체의 두 초점을  $(x_1, y_1, z_1), (x_2, y_2, z_2)$ 이라 하고, 이 둘로부터 거리의 합이  $r$ 인 점들의 집합을 구해 보았다. (단,  $r \geq a + b + c$ )

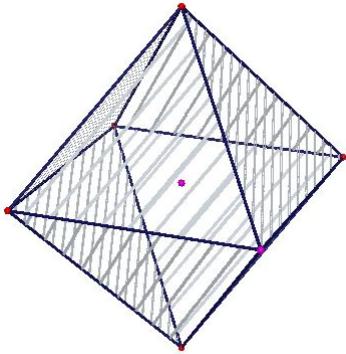


Figure 1. 택시 구

구하기에 앞서 Section 1.3의 Figure 1를 참조하여, 택시 타원체의 개형을 Figure 2와 같은 ’늘린 맞붙인 사각지붕’으로 추측하였다. (입체도형의 이름과 Figure 2 는 Poly Pro로부터 가져옴)

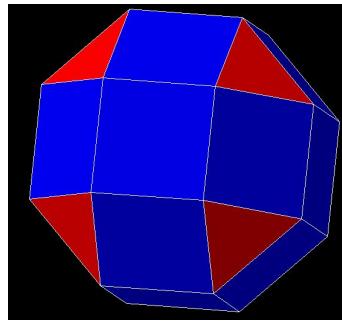


Figure 2. 추정되는 택시 타원의 개형

택시 타원체를 그려내기 위해, 우리는 유clidean 거리계에서 타원을 그릴 때 사용하는 방법을 확장시키기로 하였다. 유clidean 거리계에서 Geogebra 5을 이용하여 타원을 그리는 방법은 다음과 같다.

우리가 2차원 평면에서 두 초점으로부터의 거리의 합이  $a$ 인 점들의 집합을 구할 때에는, 반지름이  $r, a - r$ 이고 각 초점을 중심으로 하는 원을 그려 그 교점을 작도하고, 슬라이더  $r$ 에 변화를 주고 교점의 자취를 구하면 Figure 3와 같은 타원을 구할 수 있다.

마찬가지의 방법을 Cabri 3d를 이용하여 택시 타원체를 구할 때에 사용할 수 있다.

3차원 공간에서 두 초점으로부터의 거리의 합이  $a$ 인 점들의 집합을 구할 때에는, 반지름이  $r, a - r$ 이고 각 초점을 중심으로 하는 택시 구를 그려 그 교점(또는 교선)을 작도하고,

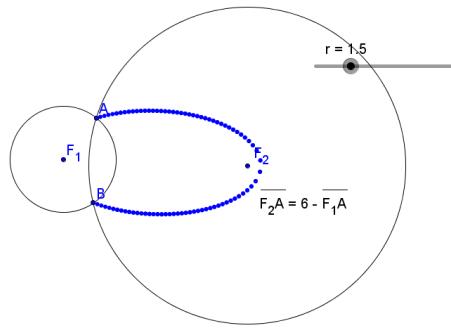


Figure 3. Geogebra 5를 이용하여 유클리드 거리계에서의 타원을 구하는 방법

슬라이더  $r$ 에 변화를 주며 교점(또는 교선)의 자취를 구하면 Figure 4와 같은 택시 타원체의 골격의 일부를 구할 수 있다.

단, Cabri 3d에는 슬라이더 기능이 없기 때문에 택시 타원 위의 한 점을 지정하고 그를 움직이면 된다.

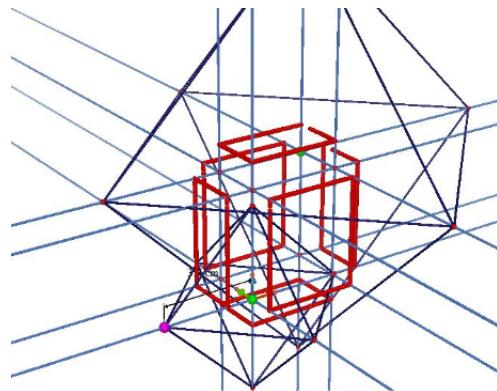


Figure 4. Cabri 3d를 이용하여 택시 타원체의 골격을 구하는 방법

우리는 Figure 4로부터 부차적인 요소들을 제하고 Cabri 3d를 이용하여 Figure 5와 같은 택시 타원체의 골격을 그려 보았다.

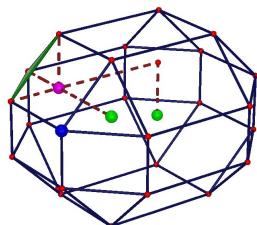


Figure 5. Cabri 3d를 이용하여 그린 택시 타원의 골격

### 3.3 Taxicab Ellipsoid - Mathematica 9

Mathematica 9를 이용하여 Figure 6와 같은 택시 타원체의 형태를 구해 보았다. 입력한 수식은 다음과 같다.

```
ContourPlot3D[Abs[(Abs[x]+Abs[3-x])+(Abs[y]+Abs[4-y])+(Abs[z]+Abs[5-z])],x,-10,10,y,-10,10,z,-10,10,Contours→20]
```

설명 : 두 초점  $(0, 0, 0), (3, 4, 5)$ 으로부터의 거리의 합이 20인 점들의 집합.

```
ContourPlot3D[
  Abs[(Abs[x] + Abs[3 - x]) + (Abs[y] + Abs[4 - y]) + (Abs[z] + Abs[5 - z])],
  {x, -10, 10}, {y, -10, 10}, {z, -10, 10}, Contours → {20}]
```

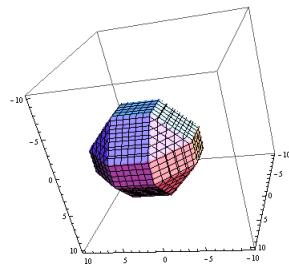


Figure 6. Mathematica 9 를 이용하여 그린 택시 타원의 골격

### 3.4 Taxicab Ellipsoid - Dev-C++

우리는 택시 타원체의 정의에 입각하여, 이를 평면  $z = i (-k \leq i \leq k)$ 로 자른 단면을 차례차례 잘라내 이를 보여주는 프로그램을 제작하였다. 실행 코드는 다음과 같다.

```
#include<stdio.h>
#include<windows.h>
#include<time.h>
#include<stdlib.h>

#define col GetStdHandle(STD_OUTPUT_HANDLE)
#define scta SetConsoleTextAttribute

int x, y, z;
int r;
```

```

int showsize;

int checkVertex(int xx,int yy);

void intro()
{
    scta(col,15);

    puts("The coordinate of one focus is (0,0,0).");

    printf("Enter the coordinate of other focus (x,y,z) : ");

    scanf("%d%d%d",&x,&y,&z);

    printf("Enter the r-value : ");

    scanf("%d",&r);

    printf("Enter the showsize (-n to n) : ");

    scanf("%d",&showsize);

}

void f()
{
    int i,j,k;

    int start;

    for(i=-showsize;i<=showsize;i++) // z

    {
        start=0;

        scta(col,15);

        printf("z = %d\n",i);

        for(j=-showsize;j<=showsize;j++) // y

        {
            for(k=-showsize;k<=showsize;k++) // x

            {
                int back=0;

                if(checkVertex(k,j)) back=128; // GRAY

                if(abs(i)+abs(j)+abs(k)+abs(x-k)+abs(y-j)+abs(z-i)!=r)

```

```

    {
        scta(col,back+8); // RED
        putchar('.');
    }
    else
    {
        start=1;
        scta(col,back+10); // WHITE
        putchar('*');
    }
    putchar(' ');
}
putchar('\n');
}

putchar('\n');
if(start==0 && i>0) break;
}

int checkVertex(int xx,int yy)
{
    int check=2;
    if(xx==0 || xx==x) check++;
    if(yy==0 || yy==y) check++;
    return check>>2;
}

int main()
{
    intro();
    unsigned int time_start,time_end;

```

```

time_start=GetTickCount () ;

scta (col, 15) ; // WHITE

f () ;

scta (col, 15) ; // WHITE

time_end=GetTickCount () ;

printf ("\nTime : %u\ (ms)\ \n", time_end-time_start) ;

}

```

이 프로그램으로 얻은 결과는 다음 그림들에서 볼 수 있듯이 Cabri 3d와 Mathematica 9를 이용하여 그린 결과와 동일했다.

여기에서 택시 타원체의 초점은  $(0, 0, 0), (3, 4, 5)$ 이며  $r = 12^\circ$ 이고,  $x, y, z = -10 \sim 10$ 의 격자점에서 나타낸 것이다.



### 3.5 Taxicab Hyperboloid - Cabri 3d

택시 쌍곡면도 타원체와 마찬가지로 평면 유클리드 거리계에서 쌍곡선을 구하는 방법을 응용하였다. 평면 유클리드 거리계에서 쌍곡선을 그리는 방법은 두 초점을 중심으로 하고 각각 반지름이  $r, a + r$ ( $a$ 는 상수)인 원을 그려 그의 초점의 자취를 구하는 것이었다. 단, 이 방법은 쌍곡선 중 하나의 곡선만 나타낸다.

마찬가지로 Cabri 3d에서 택시 쌍곡면의 골격을 구하기 위해 두 초점  $(x_1, y_1, z_1), (x_2, y_2, z_2)$ 를 중심으로 하고 각각의 반지름이  $r, a + r$ 인 택시 구를 그려 그의 교선의 자취를 그렸다. 결과는 Figure 7과 같다.

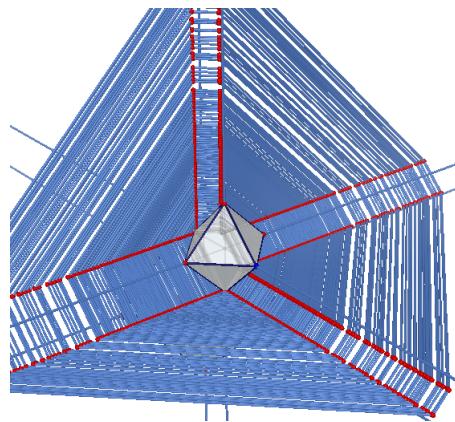


Figure 7. Cabri 3d를 이용하여 그린 택시 쌍곡면

### 3.6 Taxicab Hyperboloid - Mathematica 9

Mathematica 9를 통해 택시 쌍곡면을 그리기 위해 입력한 수식은 다음과 같다. 그 결과는 Figure 8과 같다.

```
ContourPlot3D[Abs[(Abs[x]-Abs[3-x])+(Abs[y]-Abs[4-y])+(Abs[z]+Abs[5-z])],x,-10,10,y,-10,10,z,-10,10,Contours→5]
```

설명 : 두 초점  $(0, 0, 0), (3, 4, 5)$ 으로부터의 거리의 차가 5인 점들의 집합.

택시 타원의 경우, Cabri 3d 프로그램을 통해 기하적으로 예측했던 개형과 대수적 모형이 잘 맞아떨어졌다. 이후 쌍곡면에 대한 다양한 연구를 진행하였는데, 거리의 차  $k$ 에 해당하

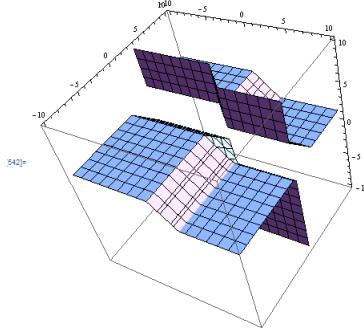


Figure 8. Mathematica 9를 이용하여 그린 택시 쌍곡면

는 값을 변화시켰을 때 이상한 점을 발견하게 되었다. 아래 그림의 경우, 초점을  $(0, 0, 0)$ 과  $(3, 4, 5)$ 로 하였을 때  $k = 2, 4, 6, 12$ 에서 Figure 9와 같은 돌출점들을 발견할 수 있었다.

처음에는 이를 프로그램상의 오류로 판단하였다. 그러나 조금 더 정밀한 분석을 위해, Dev-C++ 프로그램을 활용하여 쌍곡면의 단면을 출력하는 프로그램을 제작하였다. 프로그램을 이용해 단면을 출력한 결과, Mathematica 9에서 발견했던 돌출점들이 프로그램상의 오류가 아닌, 꽉 채워진 공간이라는 것을 알아냈다. 이를 통해 이 구역이 왜 빈 공간이 아닌 꽉 채워진 공간으로 표현되는지에 대해 생각해 보게 되었다. 직관적으로 보았을 때  $k$ 의 값이  $2 = 3 + 4 - 5, 4 = 3 + 5 - 4, 6 = 4 + 5 - 3, 12 = 3 + 4 + 5$ 와 같을 때 한 공간이 택시 쌍곡면으로 전부 채워지게 된다. 자세한 증명은 3.7절에서 다룰 것이다.

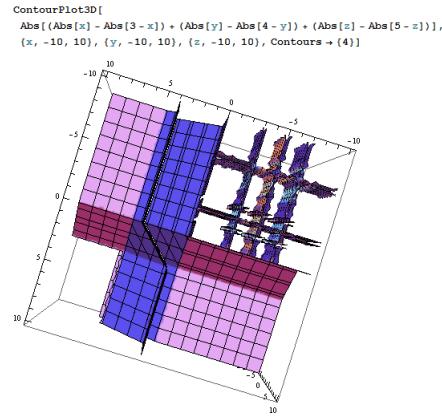


Figure 9. Mathematica 9를 이용한 택시 쌍곡면의 특이 돌출점

### 3.7 Taxicab Hyperboloid - Dev-C++

우리는 택시거리계의 정의에 입각하여 택시 쌍곡면의 방정식을 다음과 같이 수식으로 나타내었다. 이를 이용한 쌍곡면의 정의는 다음과 같다.

**정의 6.** 점  $A(x_1, y_1, z_1)$ 과  $B(x_2, y_2, z_2)$ 를 초점으로 가지는 쌍곡면의 방정식

$$|( |x - x_1| + |y - y_1| + |z - z_1| ) - ( |x - x_2| + |y - y_2| + |z - z_2| ) | = k$$

위의 정의식에 따라, 식을 전개하면,

$$|( |x - x_1| + |y - y_1| + |z - z_1| ) - ( |x - x_2| + |y - y_2| + |z - z_2| ) | = k$$

$$|( |x - x_1| - |x - x_2| ) + ( |y - y_1| - |y - y_2| ) + ( |z - z_1| - |z - z_2| ) | = k$$

$$(|x - x_1| - |x - x_2|) + (|y - y_1| - |y - y_2|) + (|z - z_1| - |z - z_2|) = \pm k (k \geq 0)$$

$x, y, z$ 를  $u$ 로 일반화시킬 때, 각 축에 따라  $|u - u_1| - |u - u_2|$ 가 나타나므로, 이 식의 값이  $u$ 에 따라 어떻게 변하는지 살펴보면, 다음과 같다.

$$1. u \leq \min(u_1, u_2)$$

$$|u - u_1| - |u - u_2| = u_1 - u_2$$

$$2. \min(u_1, u_2) < u < \max(u_1, u_2)$$

$$1. u_1 < u_2$$

$$|u - u_1| - |u - u_2| = 2u - u_1 - u_2$$

$$2. u_1 > u_2$$

$$|u - u_1| - |u - u_2| = -(2u - u_1 - u_2)$$

$$3. u \geq \max(u_1, u_2)$$

$$|u - u_1| - |u - u_2| = -u_1 + u_2$$

즉  $|u - u_1| - |u - u_2|$ 은  $u$ 에 관한 일차식( $au + b$ )으로 나타낼 수 있다. 여기서  $u$ 의 계수와 상수항을 각각 함수  $A(u), B(u)$ 로 정의하자.

**정의 7.**  $u$ 를 변수로 놓고  $u_1, u_2$ 가 그 축의 좌표인 수에 대해,

$$A(u) = \begin{cases} -\frac{|u_1 - u_2|}{u_1 - u_2} & \text{when } \min(u_1, u_2) < u < \max(u_1, u_2) \\ 0 & \text{otherwise} \end{cases}$$

**정의 8.**  $u$ 를 변수로 놓고  $u_1, u_2$ 가 그 축의 좌표인 수에 대해,

$$B(u) = \begin{cases} -u_1 + u_2 & \text{when } u \leq \min(u_1, u_2) \\ (-u_1 - u_2) * \frac{|u_1 - u_2|}{u_1 - u_2} & \text{when } u_1 < u < u_2 \\ u_1 - u_2 & \text{when } u \geq \max(u_1, u_2) \end{cases}$$

이 함수들은 각 경우에 맞추어 정의된 것이고, 그 결과 다음 식이 성립한다.

$$|u - u_1| - |u - u_2| = 2 * A(u) * u - B(u) \quad (\text{III.1})$$

그러므로 쌍곡면의 방정식은 다음과 같이 바꾸어 쓸 수 있다.

$$2 * (A(x) * x + A(y) * y + A(z) * z) = B(x) + B(y) + B(z) \pm k(k \geq 0)$$

이에 입각하여 우리는 Dev-C++을 사용하여 택시 쌍곡면의 단면을 출력하는 프로그램을 만들었다.

Dev-C++은 C/C++ 통합 개발 환경(IDE) 중의 하나로 C/C++ 언어를 코딩하여 컴파일 및 실행할 수 있다. Mathematica에서 나타난 돌출점이 오류라고 판단하여, C언어를 이용해 코딩을 해 보았다. 기본적으로 cmd창에 출력하기 때문에, 2차원으로 밖에 나타낼 수 밖에 없기 때문에 한  $z$ 값마다  $xy$ 평면을 출력하는 식으로 3차원 공간을 나타내었다.

컴퓨팅의 한계상,  $xyz$ 좌표 공간에서 격자점만을 따졌고, 또 좌변과 우변의 기우성이

같아야 하기에 다를 경우 다시 변수를 입력하도록 설정하였다. 이와 별개로, 기본적으로는 정의식만을 이용해 3차원 택시 쌍곡면을 출력할 수도 있지만, 함수  $A, B$ 로 구한 택시 쌍곡면의 방정식이 원래의 정의와 동치임을 다시 한번 검증하였다.

택시 쌍곡면은 택시 타원체와는 달리 많은 경우가 존재하기 때문에 cmd창으로만 분석하는 것에는 무리가 있었다. 실행 코드의 두 변수 FILEPRINT와 CONSOLEPRINT를 0 또는 1로 설정하여 결과를 cmd창에 출력할지 txt파일로 출력할지 선택할 수 있게 하였다. 실행 코드는 다음과 같다.

```
/* - 3-Dimensional Taxicab Hyperboloid (Plotted by C) - */

#include<stdio.h>

#include<string.h>

#include<algorithm>

#include<math.h>

#include<windows.h>

#include<time.h>

#include<conio.h>

/* - - - - - - - - - - - - - - - - - - - - - - - - */

int FILE_PRINT; //To print out the result as a file, 1

int CONSOLE_PRINT; //To print out the result on console, 1

/* - - - - - - - - - - - - - - - - - - - - - - - - */

#define col GetStdHandle(STD_OUTPUT_HANDLE)
```

```

/* - - - - - - - - - - - - - - - - - - - - - - */

FILE *fp;

/* - - - - - - - - - - - - - - - - - - - - - - */

struct Axis
{
    int f1;
    int f2;
};

Axis X, Y, Z;

int k, n;

/* - - - - - - - - - - - - - - - - - - - - - - */

int my_itoa(int value, char a[]); //Self itoa command.

int A(Axis U, int u); //The coefficient of variable

int B(Axis U, int u); //The constant term of the equation

int Hyperboloid(int x, int y, int z); //The arithmetic caculation.

int checkVertex(int x, int y); //To emphasize the focus

void Intro(); //Receives input.

void createFile(); //Create txt file as an output.

/* - - - - - - - - - - - - - - - - - - - - - - */

int main()

```

```

{
    Intro();

    if(FILE_PRINT)
    {
        createFile();
        if(fp == NULL) return 255;
    }

    unsigned int time_start,time_end;
    time_start=GetTickCount();
    for(int z=-n; z<=n; z++)
    {
        SetConsoleTextAttribute(col,15); //WHITE

        if(CONSOLE_PRINT)
            printf("z = %d\n", z);

        if(FILE_PRINT)
            fprintf(fp, "z = %d\n", z);

        for(int y=-n; y<=n; y++)
        {
            for(int x=-n; x<=n; x++)
            {
                int back=0;
                if(checkVertex(x,y))
                {
                    back=128; //GRAY
                }
                if(Hyperboloid(x,y,z)==k)
                {
                    SetConsoleTextAttribute(col,back+12);
                    //RED
                }
            }
        }
    }
}

```

```

- (B(X, x)+B(Y, y)+B(Z, z)) ;

if(abs(lvalue)==k)
SetConsoleTextAttribute(col, back+10);
//GREEN
if(CONSOLE_PRINT)
putchar('*');

if(FILE_PRINT)
fprintf(fp, "*");

}

else
{
    if(CONSOLE_PRINT)
SetConsoleTextAttribute(col, back+8),
putchar('.');

if(FILE_PRINT)
fprintf(fp, ".");

}

if(CONSOLE_PRINT)
SetConsoleTextAttribute(col, 8),
putchar(' ');

if(FILE_PRINT)
fprintf(fp, " ");

}

if(CONSOLE_PRINT)
putchar('\n');

if(FILE_PRINT)
fprintf(fp, "\n");

```

```

    }

    if(CONSOLE_PRINT)
        putchar ('\n');

    if(FILE_PRINT)
        fprintf(fp, "\n");

}

SetConsoleTextAttribute(col,15); //WHITE
time_end=GetTickCount ();
printf ("\nTime : %u(ms)\n",time_end-time_start); //Execution Time
if(FILE_PRINT)
    fclose(fp);

return 0;
}

/* - - - - - - - - - - - - - - - - - - - - - - - */

int my_itoa(int value,char a[])
{
    int len=0;
    int t_value=value;
    int parity=0; // 1 if value is negative
    if(value<0) //value<0
        a[0]='-',value*=-1,parity=1;

    if(value)
    {
        while(t_value)
        {
            len++;
            t_value/=10;
        }
    }
    else //value=0

```

```

{
    len=1;
}
for(int i=len-1+parity;i>=parity;i--)
{
    a[i]=value%10+48;
    value/=10;
}
}

int A(Axis U,int u)
{
    if(std::min(U.f1,U.f2)<u && u<std::max(U.f1,U.f2))
    {
        return -abs(U.f1-U.f2)/(U.f1-U.f2);
    }
    else return 0;
}

int B(Axis U,int u)
{
    if(u <= std::min(U.f1,U.f2)) return -U.f1+U.f2;
    else if(u >= std::max(U.f1,U.f2)) return U.f1-U.f2;
    else return -(U.f1+U.f2)*abs(U.f1-U.f2)/(U.f1-U.f2);
}

int Hyperboloid(int x,int y,int z)
{
    int output_Hyperboloid=abs(abs(x-X.f1)+abs(y-Y.f1)+abs(z-Z.f1)
    -abs(x-X.f2)-abs(y-Y.f2)-abs(z-Z.f2));
    return output_Hyperboloid;
}

int checkVertex(int x,int y)
{
    int check=2;

```

```

if(x==X.f1 || x==X.f2) check++;
if(y==Y.f1 || y==Y.f2) check++;
return check>>2;
}

void Intro()
{
    printf("The 3-dimensional Taxicab Hyperboloid -");
    puts(" Plotted by C.\n");
    printf("This programm plots the 3-dimensional Taxicab Hyperboloid ");
    puts("in integer range.\n");
    puts("Type foci of x/y/z axis, rvalue, and range.\n");
    SetConsoleTextAttribute(col,14); //YELLOW
    printf("Warning! The parity of the sum of three foci and rvalue ");
    puts("must be identical.\n");

    SetConsoleTextAttribute(col,15); //WHITE
    char ch=1;
    printf("To print as a file, enter 1. If not, enter 0. ");
    SetConsoleTextAttribute(col,11); //SKYBLUE
    reenter_file:
    ch=getch();
    if(ch!='0' && ch!='1') goto reenter_file;
    else if(ch=='1')
    {
        FILE_PRINT=1;
        puts("Print as a file.\n");
    }
    else //ch=='0'
    {
        puts("Do not print as a file.\n");
    }

    SetConsoleTextAttribute(col,15); //WHITE
    printf("To print as a console, enter 1. If not, enter 0. ");
    SetConsoleTextAttribute(col,11); //SKYBLUE
}

```

```

reenter_console:
ch=getch();
if(ch!='0'&&ch!='1')    goto  reenter_console;
else  if(ch=='1')
{
    CONSOLE_PRINT=1;
    puts("Print as a console.\n");
}
else //ch=='0'
{
    puts("Do not print as a console.\n");
}

reenter:
SetConsoleTextAttribute(col,15); //WHITE
printf("The x-coordinate of first focus : ");
scanf("%d",&X.f1);
printf("The y-coordinate of first focus : ");
scanf("%d",&Y.f1);
printf("The z-coordinate of first focus : ");
scanf("%d",&Z.f1);
printf("The x-coordinate of second focus : ");
scanf("%d",&X.f2);
printf("The y-coordinate of second focus : ");
scanf("%d",&Y.f2);
printf("The z-coordinate of second focus : ");
scanf("%d",&Z.f2);
printf("The rvalue(k) : ");
scanf("%d",&k);
if(abs(X.f1+Y.f1+Z.f1-X.f2-Y.f2-Z.f2-k)%2==1)
{
    SetConsoleTextAttribute(col,12); //RED
    puts("");
    printf("Error! The parity of the sum of three foci and rvalue ");
    puts("must be identical.");
    puts("Please re-enter the values.\n");
}

```

```

    goto reenter;
}
printf("The size of the graph(from -n to n) : ");
scanf("%d", &n);
puts("");
}

void createFile()
{
    char filename[200]={"3-Dimensional Taxicab Hyperboloid ("};
    char v_value[10]={};

    my_itoa(X.f1, v_value);
    strcat(filename, v_value);
    strcat(filename, ", ");
    memset(v_value, 0, sizeof(v_value)); //Initializing

    my_itoa(Y.f1, v_value);
    strcat(filename, v_value);
    strcat(filename, ", ");
    memset(v_value, 0, sizeof(v_value));

    my_itoa(Z.f1, v_value);
    strcat(filename, v_value);
    strcat(filename, ", ");
    memset(v_value, 0, sizeof(v_value));

    my_itoa(X.f2, v_value);
    strcat(filename, v_value);
    strcat(filename, ", ");
    memset(v_value, 0, sizeof(v_value));

    my_itoa(Y.f2, v_value);
    strcat(filename, v_value);
    strcat(filename, ", ");
    memset(v_value, 0, sizeof(v_value));
}

```

```

my_itoa(z.f2, v_value);
strcat(filename, v_value);
strcat(filename, ", ");
memset(v_value, 0, sizeof(v_value));

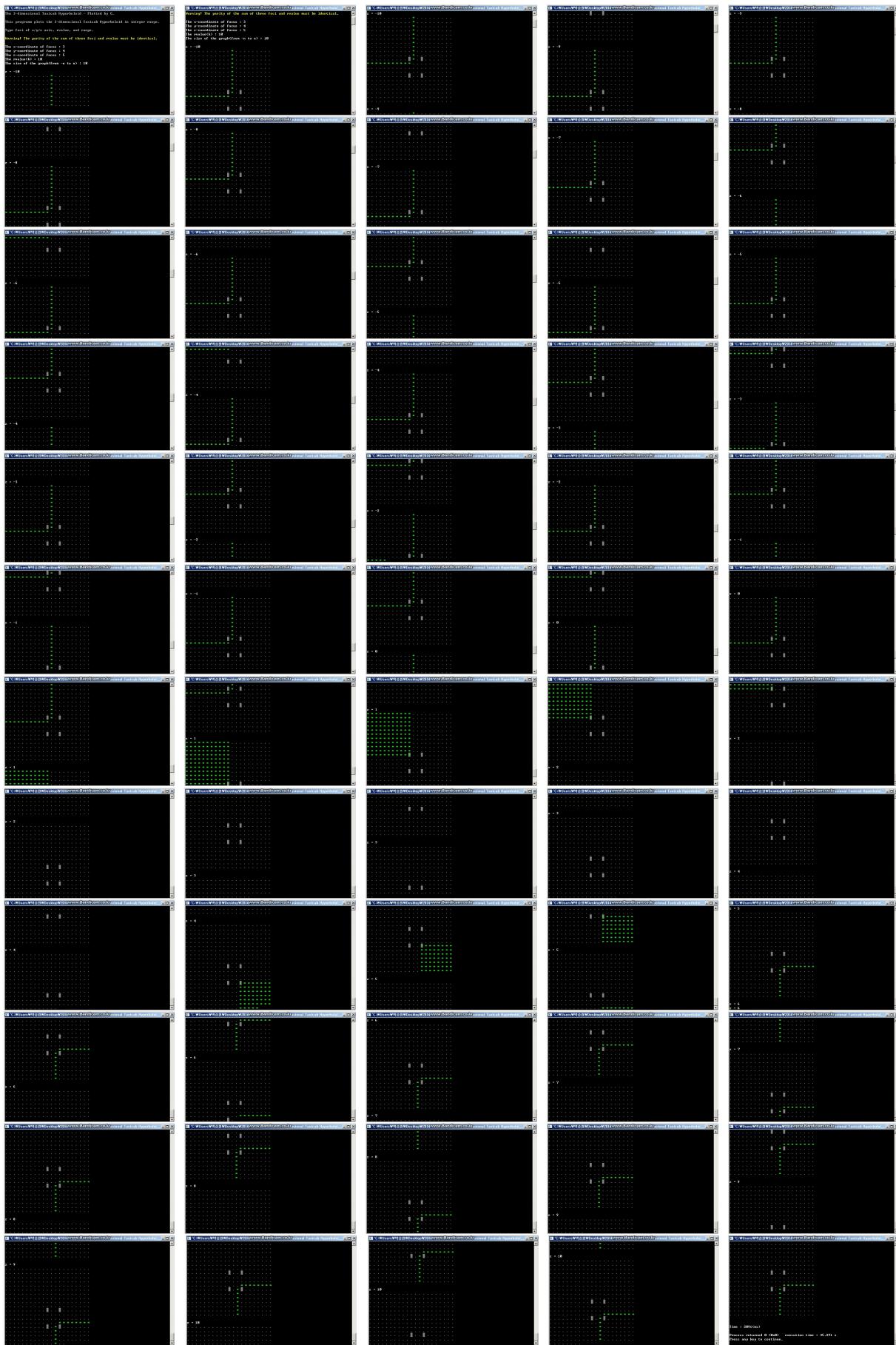
my_itoa(k, v_value);
strcat(filename, v_value);
strcat(filename, ", ");
memset(v_value, 0, sizeof(v_value));

my_itoa(n, v_value);
strcat(filename, v_value);
strcat(filename, ".txt");
memset(v_value, 0, sizeof(v_value));

fp = fopen(filename, "w");
fprintf(fp, "%s\n", filename);
}

```

이를 실행한 결과는 다음과 같다. 여기에서 택시 쌍곡면의 초점은  $(0, 0, 0), (3, 4, 5)$ 이며  $r = 10$ 인 경우  $-10 \sim 10$ 을 나타낸 것이다.



### 3.8 Taxicab Paraboloid - Cabri 3d

우선 택시 포물면을 정의하기 위해서는 점과 평면 사이의 거리를 재검토해야 한다.

흔히 우리가 유클리드 거리계에서 점과 평면 사이의 거리를 정의할 때는, 그 점을 중심으로 하는 구의 반지름을 점점 늘려갈 때 구가 평면에 접하는 순간 구의 반지름의 길이로 정의한다. 이는 곧 점에서 평면에 내린 수선의 발을 의미한다.

하지만 택시 거리계에서는 택시 구가 정팔면체로, 유클리드 거리계에서의 구와는 달리 특정 방향에 대해서만 대칭이기 때문에 점과 평면 사이의 거리가 새롭게 정의되었다. 이 상황을 Figure 10,11과 같이 Cabri 3d로 구현해 보았다.

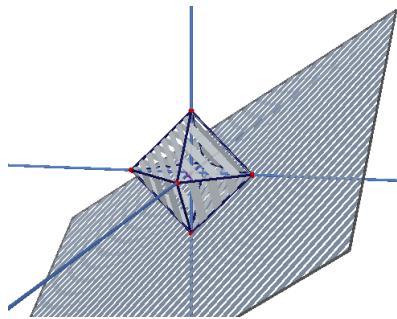


Figure 10. Cabri 3d를 이용하여 시각화한 점과 평면 사이의 거리

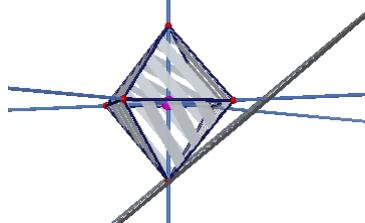


Figure 11. Figure 10과 같은 상황의 다른 시점을 통한 관찰

### 3.9 Taxicab Paraboloid - Mathematica 9

Cabri 3d로는 대수적인 구현이 힘들어, 이를 Mathematica 9로 구현해 보았다. 택시 타원체와 쌍곡면의 경우에는 그 곡면의 모양을 통해 초점의 위치를 추정할 수 있었다. 하지만, 택시 포물면의 경우 두개의 초점이 아닌 하나의 초점과 준면으로 정의되었고, 포물면의 모양을 통해서만은 준면의 위치를 추정할 수 없었다. 따라서 우리는 다음과 같은 수식을 통해 인위적으로 표면을 공간에 추가 출력하였다. 그를 통해 Figure 12와 같은 포물면의 다양한

형태를 관찰할 수 있었고, 그 모양을 수식적으로 증명하였다.

*Slider[Dynamic[a], -3, 3, 0.1]*

*Slider[Dynamic[b], -3, 3, 0.1]*

*Slider[Dynamic[c], -3, 3, 0.1]*

*Slider[Dynamic[d], -3, 3, 0.1]*

*ContourPlot3D[*

*(Abs[(a\*x+b\*y+c\*z+d)]/Max[a,b,c]-(Abs[x]+Abs[y]+Abs[z]))\*(a\*x+b\*y+c\*z+d)==0,*

*x, -10, 10, y, -10, 10, z, -10, 10]*

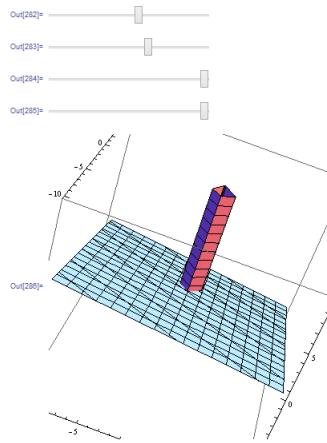


Figure 12. Mathematica 9를 이용하여 그린 택시 포물면

### 3.10 Taxicab Paraboloid - Dev-C++

앞에서의 증명을 이용하여, 택시 포물면의 식을 다음과 같이 쓸 수 있다.

**정의 9.** 한 평면( $\circ$ )하 준면이라 통칭한다)  $\alpha : ax + by + cz + d = 0$ 과

점  $P : (k, l, m)$ 를 초점으로 가지는 포물면의 방정식은

$$|x - k| + |y - l| + |z - m| = \frac{|ax + by + cz + d|}{\max(|a|, |b|, |c|)} \circ \text{다.}$$

포물면의 식을 각 조건에 따라 풀어보기 전에 다음 보조정리를 증명하겠다.

**보조정리 1.** 포물면에 속해있는  $(x, y, z)$ 에 대하여  $\frac{(ax + by + cz + d)}{\max(|a|, |b|, |c|)}$ 의 부호는 일정하다.

*Proof.* 평면  $ax + by + cz + d = 0$ 은 좌표공간을  $ax + by + cz + d > 0$ 인 공간과  $ax + by + cz + d < 0$ 인 공간으로 나눈다. 이 중 초점은 둘 중 하나의 공간에 존재한다. 만일 포물면이  $ax + by + cz + d = 0$ 을 기준으로 초점의 반대편에 존재할 경우, 초점에서의 거리가 평면까지의 거리보다 무조건 크므로 포물면의 정의에 모순이 된다. 그러므로 포물면은  $ax + by + cz + d = 0$  기준으로 초점과 같은 쪽에 존재한다. 그 결과  $ax + by + cz + d$ 의 부호가 일정하고,  $\max(|a|, |b|, |c|)$ 의 부호 역시 일정하기에  $(ax + by + cz + d)/\max(|a|, |b|, |c|)$ 의 부호는 일정하다.  $\square$

그럼 포물면의 식을 풀어보겠다.  $WLOG |a| \geq |b| \geq |c|$ 라 하자. 이 경우 우변은  $\frac{|ax + by + cz + d|}{|a|}$ 이 된다. 편의상  $\frac{ax + by + cz + d}{a} > 0$ 으로,  $a > 0$ 으로 설정하겠다. 이렇 게 설정해도 타당한 이유는  $a$ 가 음수라 해도  $\frac{ax + by + cz + d}{a}$ 에서  $a, b, c, d$ 에 각각  $-1$ 씩  $-1$ 을 곱할 때, 식의 변화가 없기 때문이다.

1.  $x = k$

$$|y - l| + |z - m| = \frac{|ak + by + cz + d|}{|a|}$$

$y' = y - l, z' = z - m$ 이라 하자.

$$|y'| + |z'| = \frac{|ak + bl + cm + d + by' + cz'|}{|a|}$$

$F = ak + bl + cm + d$ 이라 하면

$$|y'| + |z'| = \frac{|F + by' + cz'|}{|a|}$$

$$|y'| + |z'| = \frac{F + by' + cz'}{a}$$

위 식을  $y', z'$ 의 부호에 따라 4가지로 나누어,  $y$ 와  $z$ 에 관한 식으로 나타날 때

$x = k, F = ak + bl + cm + d$		
$y \geq l$	$z \geq m$	$(a - b)(y - l) + (a - c)(z - m) = F$
	$z \leq m$	$(a - b)(y - l) + (-a - c)(z - m) = F$
$y \leq l$	$z \geq m$	$(-a - b)(y - l) + (a - c)(z - m) = F$
	$z \leq m$	$(-a - b)(y - l) + (-a - c)(z - m) = F$

일반적으로는 네 개의 선분이 생기면서  $(k, \frac{F}{a-b} + l, m), (k, l, \frac{F}{a-c} + m), (k, \frac{F}{a-b} + l, m), (k, l, \frac{F}{-a-c} + m)$ 을 꼭짓점으로 하는 사각형이 자취로 나오게 된다. 그러나  $|a| = |b|$ 이나  $|a| = |c|$ 일 경우, 인접한 사분면의 선분이 서로 평행한 반직선이 되어 꼭짓점이 생기지 않게 된다.

2.  $x > k$

$$|x - k| + |y - l| + |z - m| = \frac{|ak + by + cz + d|}{|a|}$$

$x' = x - k, y' = y - l, z' = z - m$ 이라 하자.

$$|x'| + |y'| + |z'| = \frac{|ak + bl + cm + d + ax' + by' + cz'|}{|a|}$$

$F = ak + bl + cm + d$ 이라 하면

$$|x'| + |y'| + |z'| = \frac{|F + ax' + by' + cz'|}{|a|}$$

앞에서  $\frac{F + ax' + by' + cz'}{a}$  양수라고 가정하였으므로

$$|x'| + |y'| + |z'| = \frac{F + ax' + by' + cz}{a}$$

또,  $x' = x - k > 0$ 므로

$$x' + |y'| + |z'| = \frac{F + ax' + by' + cz}{a}$$

$$|y'| + |z'| = \frac{F + by' + cz}{a}$$

즉  $x = k$ 일 때와 동일한 식이 나온다. 그러므로 다음과 같이 나타낼 수 있다.

$x \geq k, F = ak + bl + cm + d$		
$y \geq l$	$z \geq m$	$(a - b)(y - l) + (a - c)(z - m) = F$
	$z \leq m$	$(a - b)(y - l) + (-a - c)(z - m) = F$
$y \leq l$	$z \geq m$	$(-a - b)(y - l) + (a - c)(z - m) = F$
	$z \leq m$	$(-a - b)(y - l) + (-a - c)(z - m) = F$

3.  $x < k$

1.의 경우와 똑같이 전개시

$$|x'| + |y'| + |z'| = \frac{|F + ax' + by' + cz'|}{|a|}$$

$$x' < 0 \text{므로}, -x + |y'| + |z'| = \frac{F + ax' + by' + cz'}{a}$$

$$|y'| + |z'| = \frac{F + ax' + by' + cz'}{a} + 2x'$$

만일  $x' < -\frac{F}{2a}$  일때,  $x' = -\frac{F}{2a} - \alpha (\alpha > 0)$ 라 하자.

$$|y'| + |z'| = \frac{F + ax' + by' + cz'}{a} - \frac{F}{a} - 2\alpha$$

$$|y'| + |z'| = \frac{by'}{a} + \frac{cz'}{a} - 2\alpha$$

$$\text{그런데 } \frac{by'}{a} \leq |y'|, \frac{cz'}{a} \leq |z'|, 2\alpha > 0.$$

$$\therefore |y'| + |z'| > \frac{by'}{a} + \frac{cz'}{a} - 2\alpha$$

이는 정의식에 모순이므로,  $x' < -\frac{F}{2a}$  일 때는 해가 없다.

그러므로 주어진 식  $|y'| + |z'| = \frac{F + ax' + by' + cz'}{a} + 2x'$  을 전개시

$x < k, F = ak + bl + cm + d$		
$y \geq l$	$z \geq m$	$-2(x - k) + (a - b)(y - l) + (a - c)(z - m) = F$
	$z \leq m$	$-2(x - k) + (a - b)(y - l) + (-a - c)(z - m) = F$
$y \leq l$	$z \geq m$	$-2(x - k) + (-a - b)(y - l) + (a - c)(z - m) = F$
	$z \leq m$	$-2(x - k) + (-a - b)(y - l) + (-a - c)(z - m) = F$

이는  $X + 2(x - k)$ 를 상수로 취급시,  $x \geq k$ 의 경우와 닮은 도형이 나오는 형태이다. 그 결과 그래프 자체는  $x = k - \frac{F}{2a}$  일 때,  $(k - \frac{F}{2a}, l, m)$ 이 꼭짓점의 형태로 나타나게 된다.

그러나 이 역시  $|a| = |b|$ 이거나  $|a| = |c|$ 일 때 방정식에서 해당하는 변수가 사라져, 꼭짓점이 점의 형태가 아닌 선 내지는 면으로 나타날 수 있다.

포물면의 특성상 점과 평면 사이의 거리가 정수가 아닌 유리수가 나온다.

따라서 우리는 Taxicab Ellipsoid, Hyperboloid에서와는 다르게 단위격자의 간격을 1에서 0.1로 더 세분화하고, 식 (준평면과 점의 거리) = (초점과 점의 거리)에서 양변의 값의 차이가 특정 값(det1) 이하이면 그 점은 대략적으로 포물면 위에 있는 것으로 인정해 주는 방법으로 포물면을 나타내었다. 여러 번의 실행을 통해 그 특정 값은 0.01정도가 적당했다.

또한 포물면의 모양을 더욱 효과적으로 보기 위해 모양을 나타내는 좌표의 범위를 적당히 조절해 가면서 프로그램을 실행하였고, 이를 이용하여 Mathematica 9에 의해 그려진 택시 포물면의 모양과 일치함을 확인할 수 있었다.

실행 코드는 다음과 같다.

```
#include<stdio.h>
#include<windows.h>
#define db double
#define det1 0.15
#define det2 0.1
#define ratio 6.0
```

```

#define col GetStdHandle(STD_OUTPUT_HANDLE)
#define scta SetConsoleTextAttribute

int x,y,z,a,b,c,d;
db abs(db x){return x>0?x:-x;}
db MAX(db A, db B, db C){
    int t;
    if(A>B) t=A, A=B, B=t;
    if(A>C) t=A, A=C, C=t;
    if(B>C) t=B, B=C, C=t;
    return C;
}
db dis(db p, db q, db r){
    return abs((db)(a*p+b*q+c*r+d))/db(MAX(abs(a),abs(b),abs(c)));
}
db taxi(db x1, db y1, db z1, db x2, db y2, db z2){
    return abs(x1-x2)+abs(y1-y2)+abs(z1-z2);
}
int det(db p, db q, db r)
{
    if(abs(taxi(x,y,z,p,q,r)-dis(p,q,r))<det1) return 1;
    else return 0;
}
int checkvertex(db p, db q, db r)
{
    if((q==0 && r==0) || (abs(q-y)<=det2&&abs(r-z)<=det2)) return 1;
    else return 0;
}
int main()
{
    int i,j,k;
    int r;
    scta(col,15);
    puts("Enter the coordinates(x,y,z) of focus");
    scanf("%d%d%d",&x,&y,&z);
    puts("Enter the equation of plane(ax+by+cz+d=0)");
    scanf("%d%d%d%d",&a,&b,&c,&d);
}

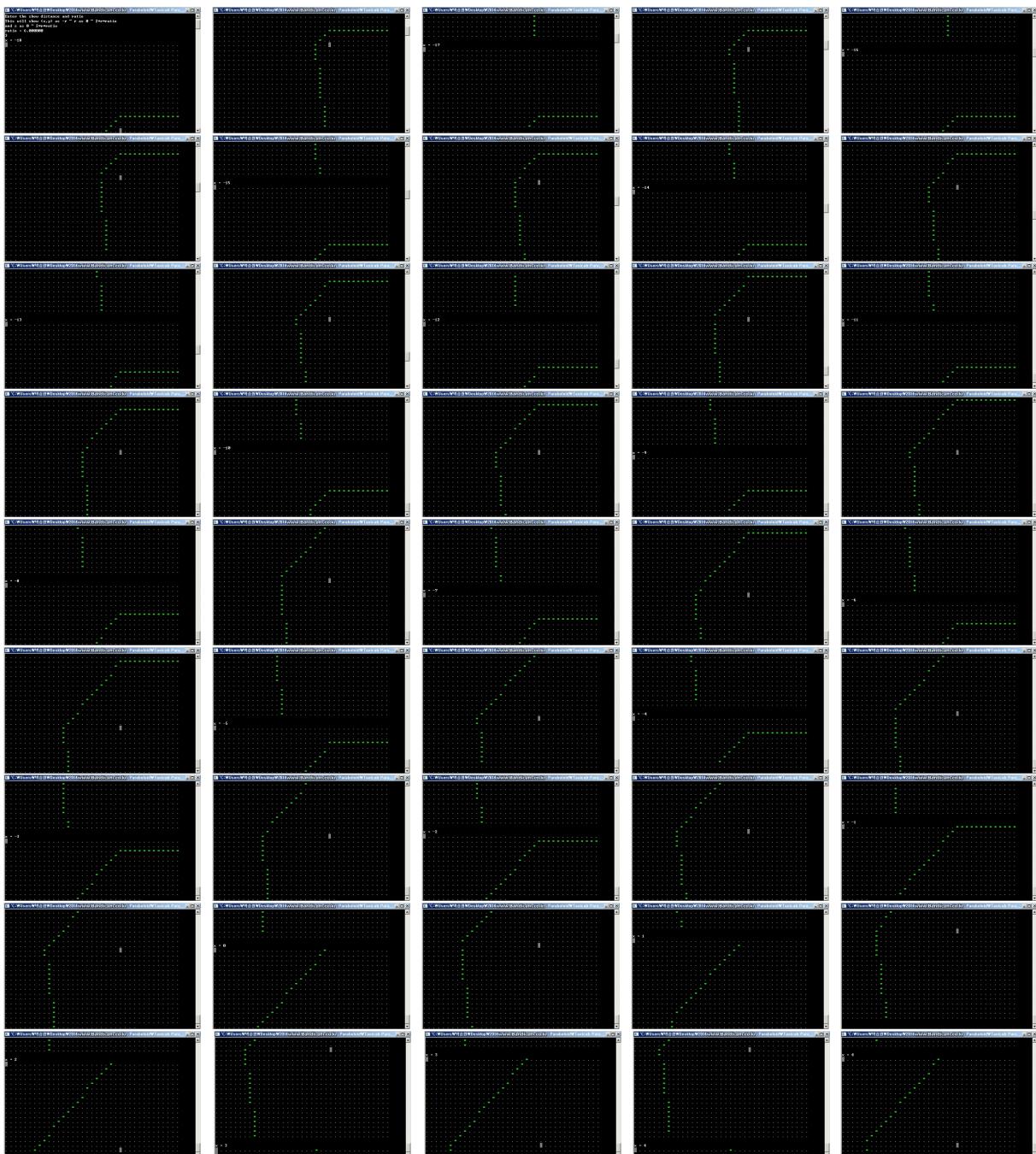
```

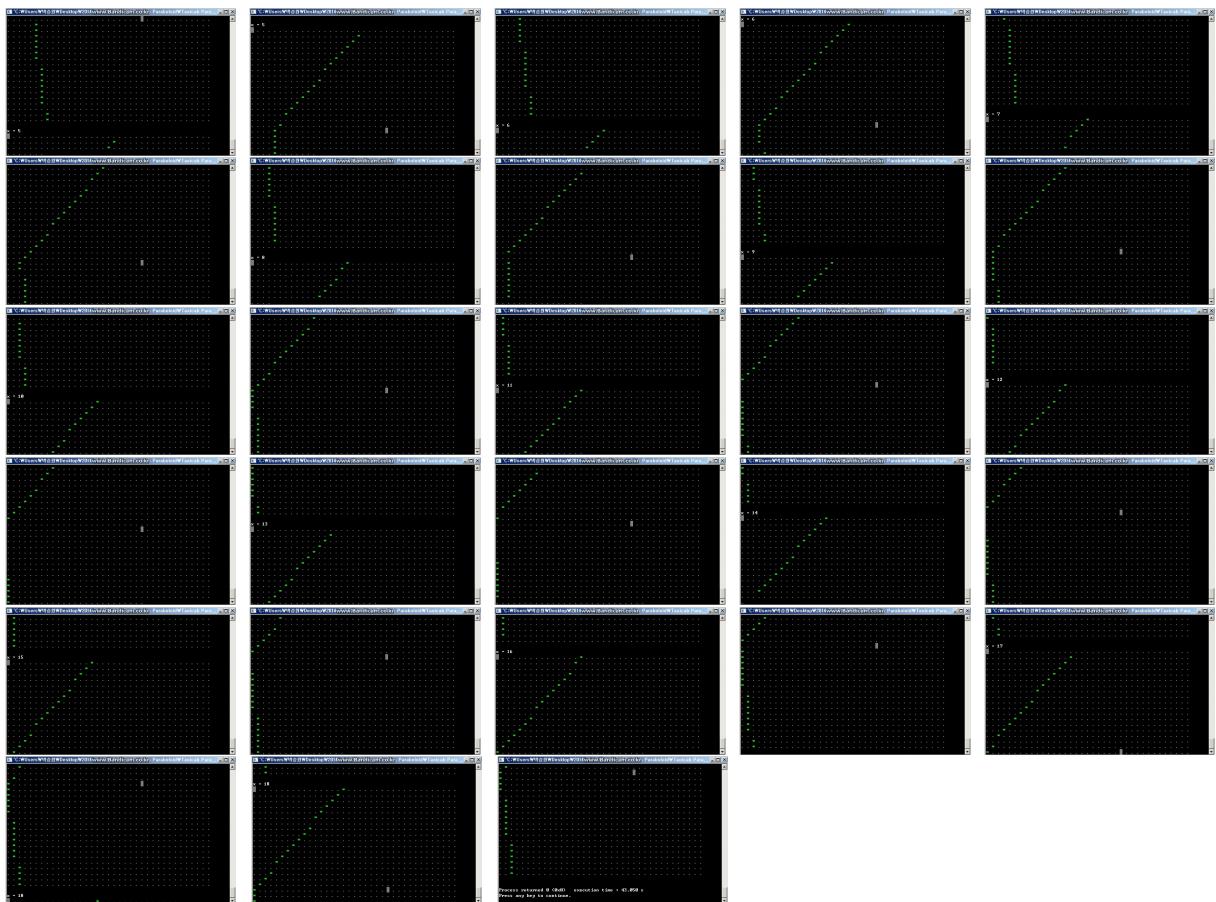
```

puts ("Enter the show distance and ratio");
puts ("This will show (x,y) as -r ~ r as 0 ~ 2*r*ratio");
puts ("and z as 0 ~ 2*r*ratio");
printf ("ratio : %lf\n",ratio);
scanf ("%d",&r);
for(i=-r*ratio;i<=r*ratio;i++)
{
    scta(col,15);
    printf ("x = %d\n",i);
    for(j=0;j<=2*r*ratio;j++)
    {
        for(k=0;k<=2*r*ratio;k++)
        { // (i/ratio, j/ratio, k/ratio)
            int back=0;
            if(checkvertex(i/ratio,j/ratio,k/ratio)) back=128;
            if(det(i/ratio,j/ratio,k/ratio))
            {
                scta(col,back+10);
                putchar('*');
            }
            else
            {
                scta(col,back+8);
                putchar('.');
            }
            scta(col,8);
            putchar(' ');
        }
        putchar('\n');
    }
    putchar('\n');
}
scta(col,15);
}

```

이를 실행한 결과는 다음과 같다. 이는 초점의 좌표로  $(2, 3, 4)$ , 준평면을  $2x + 3y + 4z + 5 = 0$ 으로 하여 평면  $z * ratio = 0 \sim 36$ 으로 도형을 자른 단면을  $(x * ratio, y * ratio) = (-18 \sim 18, 0 \sim 36)$ 의 범위에서 출력한 것이다.





## IV. Conclusion & Suggestion

- ① 다양한 프로그램을 이용, 3차원 택시거리계를 기하 및 대수적으로 접근하였다.
- ② 3차원 택시거리계에서의 이차곡면, 특히 쌍곡면과 포물면의 개형 및 특징을 분석하였다.
- ③ 택시거리계 이차곡면에서 적용되었던 논리들이 유클리드 거리계 이차곡면에서도 적용되는지 비교할 수 있다. 또한  $N$ 차원으로 확장한다면 어떠한 특징이 나타날 지 연구할 수 있다.
- ④ 궁극적으로는 택시거리계의 이차곡면을 이용하여 직교좌표계가 아닌 새로운 좌표계를 만들거나, 유클리드 거리계에서의 이차곡면들이 빛, 소리와 같은 파동의 수렴과 발산에 쓰이듯이 택시 거리계에서의 이차곡면들도 특수렌즈, 특수거울의 모양을 결정짓는 데 사용될 수 있을 것이다.

## V. References

- [1] Von Rüstem Kaya, Ziya Akça, İbrahim Günaltılı, Münevver Özcan, 2000 “General Equation For Taxicab Conics And Their Classification”, Mitt. Math. Ges. Hamb, Vol. 19, pp.135-148
- [2] 양형준, 고일석 (2012) “Study of quadratic curves on Generalized Absolute-Value Metric”, 경기과학고등학교 2012 R&E 결과보고서, pp.323-338
- [3] 김민성 외 5명 (2011) “On the plane geometry using non-Euclidean distance function”, 경기과학고등학교 2011 R&E 결과보고서, pp.67-128