2017 年度桂田研究室卒業研究レポート

錯視と不可能立体の数理

明治大学 総合数理学部 現象数理学科
加王 由夏

指導教員　桂田 祐史 准教授

2017 年 2 月 15 日

目次

第１章　はじめに

人間の目と脳にはものを正しく見るための様々な機能が備わっている。しかし、いつも正しく見えているかというとそうではなく、それらの機能を巧妙に欺く錯視という現象が身の周りでしばしば起こり、私たちの視覚は意外にもあっさり騙されてしまう。錯視は古くから「だまし絵」として芸術分野で意図的に用いられており、多くの人々を楽しませてきた。

錯視についての科学的研究の歴史はまだまだ浅い。だまし絵が芸術として2000年もの歴史をもつという説もある一方で、20世紀半ば頃にようやく錯視が認知心理学や脳科学といった分野で学問として扱われるようになった。数理科学としての研究となるとさらに新しく、その歴史はわずか30年ほどである。

今回私はこの卒業研究を行うにあたって、錯視を用いて作られる立体である「不可能立体」に注目し、新しいパターンの不可能立体を発見することを目標とした。先行研究としては、明治大学大学院先端数理科学研究科の杉原厚吉教授の著書・論文を参考にした。

第２章　立体における錯視

2.1　立体視の原理

人間が世界を立体として捉えるためには、３つの原理が働いている。

『両眼立体視』

人間は目を２つ持っており、異なる２つの視点から捉えた像を脳で組み合わせることによって、ものの奥行きを把握することができる。この原理を両眼立体視という。遠くにあるものは左右どちらの目で見てもほとんど同じように見えるが、近くにあるものを見ると、左右の目からの距離の差が割合大きくなり、左右の視線のなす角が大きくなって異なる向きから物体を捉えることになるためである。

『運動立体視』

電車の窓から外を眺めたとき、近くの景色は速く流れ、遠くの景色はゆっくりと流れていくように見える。このように、自分の位置が変化したときにどのくらい対象が動いて見えるかによって、自分と対象との距離を把握する原理が運動立体視である。

『単眼立体視』

両眼立体視や運動立体視では、空間に存在する形や動きの情報を視覚のデータとして収集し、それを脳が数理的な原理で解析している。しかし、写真や絵などの２次元の画像を見たときにはそれらの原理は役に立たない。そのような奥行きの情報を持たない１枚の画像からそこに映る立体を復元するためには、予備知識や先入観、思い込みといった情報を用いる必要がある。こういった情報から立体を復元する原理が単眼立体視である。

第３章　不可能立体

3.1　不可能立体の定義と例

不可能図形と呼ばれる絵がある。これらは人間が見たときに、立体であるような印象をもつにもかかわらず、矛盾を含んでいて実際に立体として作ることができないように感じる絵である。版画家エッシャーが自身の作品に用いて、のちに数学者ペンローズが定型化した「ペンローズの階段」をはじめ、「ペンローズの三角形」、「悪魔のフォーク」などが有名である。

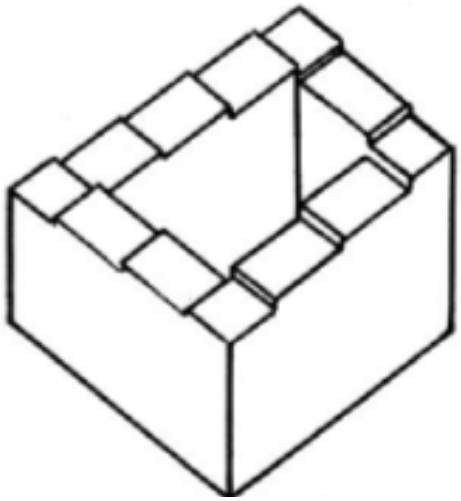


図１．ペンローズの階段

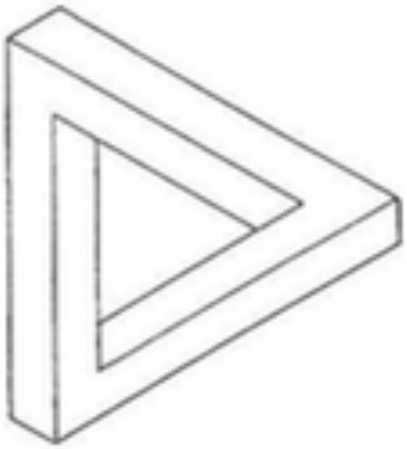


図２．ペンローズの三角形

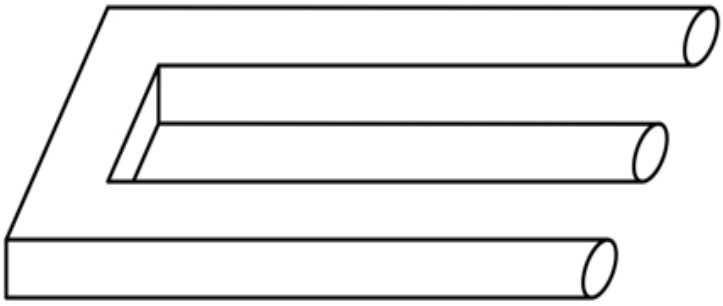


図３． 悪魔のフォーク

直感的には確かに作れない立体であると感じる数々の不可能図形だが、これらを立体として実現することは本当に不可能なのであろうか。

実は、不可能図形のなかには実現できるものがある。図のように、直角に見える部分に鋭角や鈍角を用いたり、連続に見える部分を非連続にしたりすることによって、ある特定の視点から見たときに限って不可能図形の絵に描いたとおりに見えるような立体を作ることができるのだ。このような、不可能図形を実現した立体のことを不可能立体という。

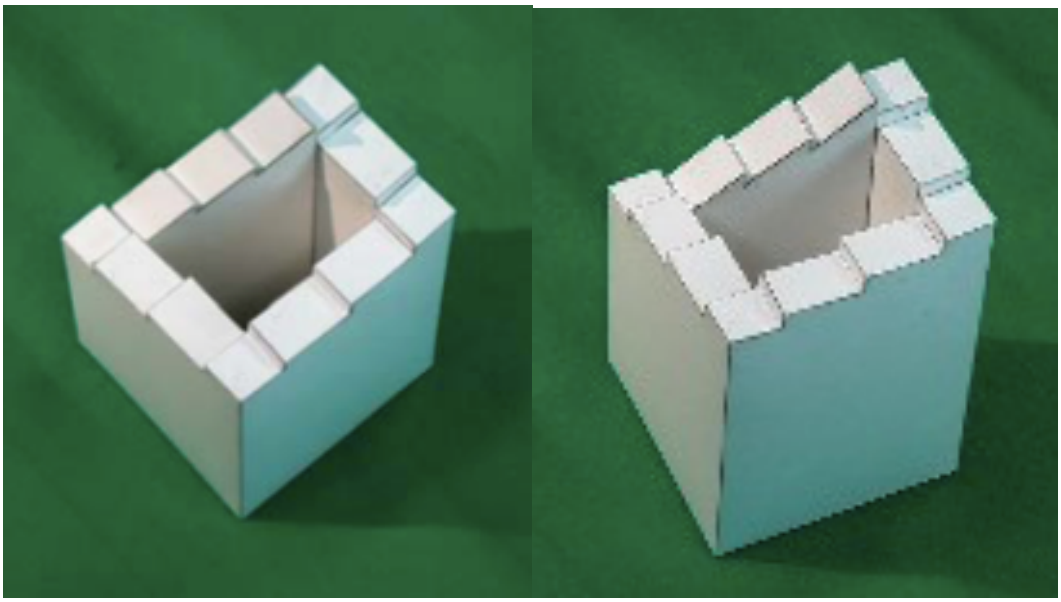


図４． ペンローズの階段の立体（杉原厚吉 HP より）

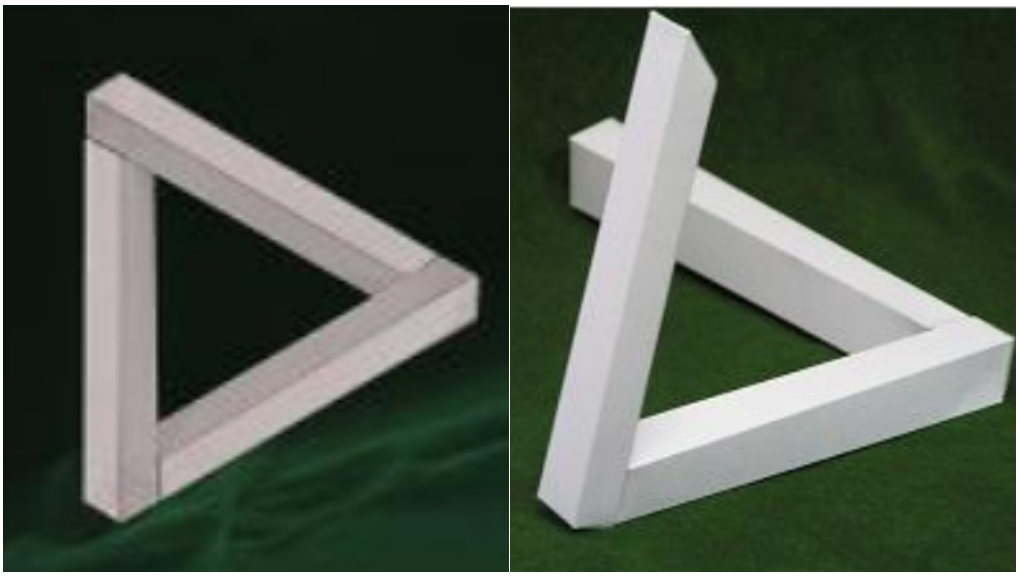


図５． ペンローズの三角形の立体（杉原厚吉 HP より）

不可能立体を作るためには、直角でないところを直角に見せる「非直角のトリック」、連続でないところを連続に見せる「非連続のトリック」といった錯視が不可欠である。人間は普段の生活で直角に慣れ親しんでいるため、直角の可能性がある部分を直角であると思い込みやすく、非直角のトリックは特に頻繁に用いられる。

これまでに、杉原教授によってさまざまなパターンの不可能立体が発見されてきた。これらは大きく分けて６世代に分類される。

第１世代『だまし絵立体』

単体で見て、どこかに矛盾が起こっているように感じる立体。奥行きの情報があれば実際の形がわかってしまうので、片目で見たときもしくは写真に撮って見たときにだけ錯視が起こる。

第２世代『不可能モーション立体』

単体で見たときにはおかしいところはないように見えるが、その周囲で棒やボールなどの別の物体を動かしたときにありえない動きをしているように感じる立体。だまし絵立体と同様に、片目もしくは写真で見る。

第３世代『変身立体』

単体で見たときにはおかしいところはないように見えるが、鏡に映すと全く違う形に映る立体。

第４世代『透身立体』

変身立体の派生。鏡に映すと、一部が消えたり、なかった部分が現れたりしたように見える立体。

第５世代『トポロジー撹乱立体』

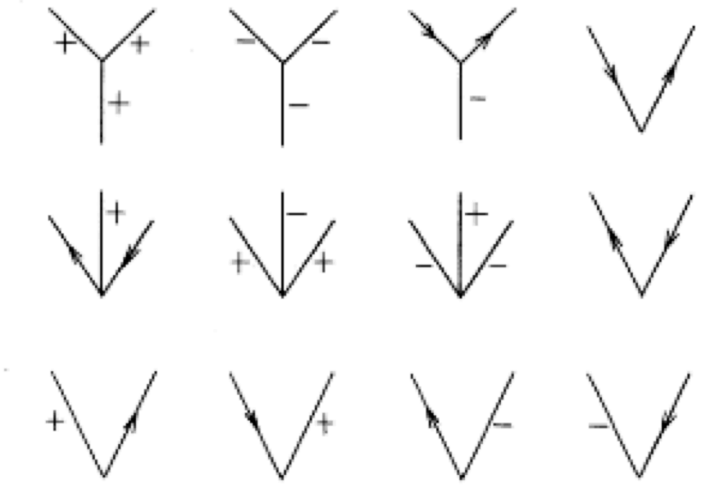
変身立体の派生。鏡に映すと違う形に映り、さらに、連続だった部分が非連続になったり、離れていた稜線が交差したりして見える立体。

第６世代『軟体立体』

回転させると連続的に形が変わって見える立体。点対称でない図形に見えるにもかかわらず、180 度回転させるともとの形に戻る。

3.2　頂点辞書

ある立体の絵を描いたときに、その絵が、脳が捉えたとおりに実際に立体として作れる絵であるか、もしくは不可能図形であるかどうかを調べたい。そこで用いられるのが頂点辞書である。ただし、頂点辞書は、次の条件のもとで用いるものとする。



3.3　不可能図形の描き方

典型的な不可能図形を描く方法がある。

第４章　オリジナル不可能立体

4.1　紙工作

3.3 で述べた典型的な描き方で不可能図形を描き、非直角のトリックを用いて、その絵と同じように見える立体を厚紙で製作した。

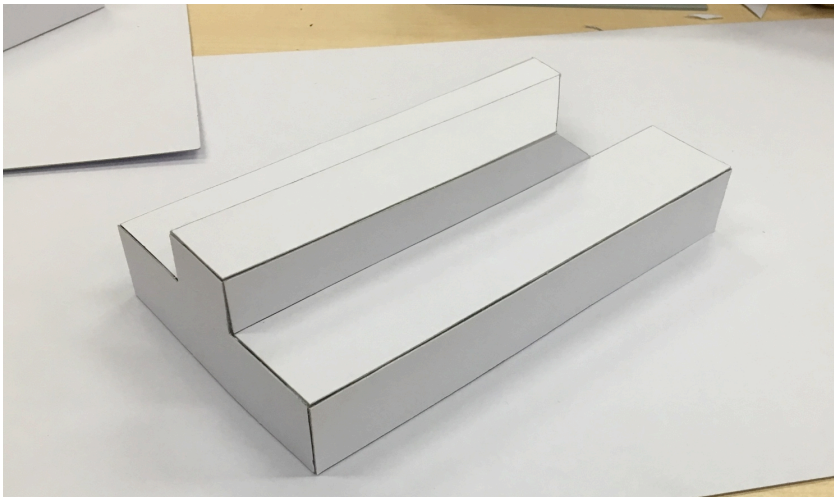


図８．オリジナル①

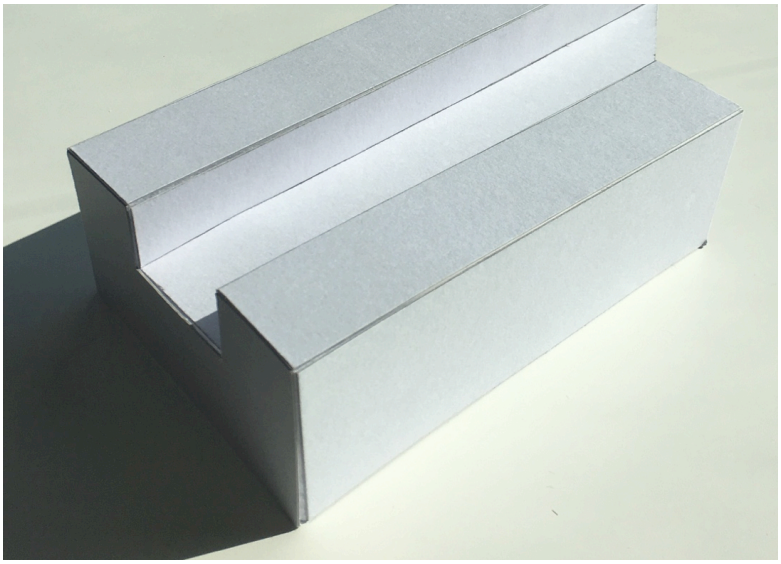


図９．オリジナル②

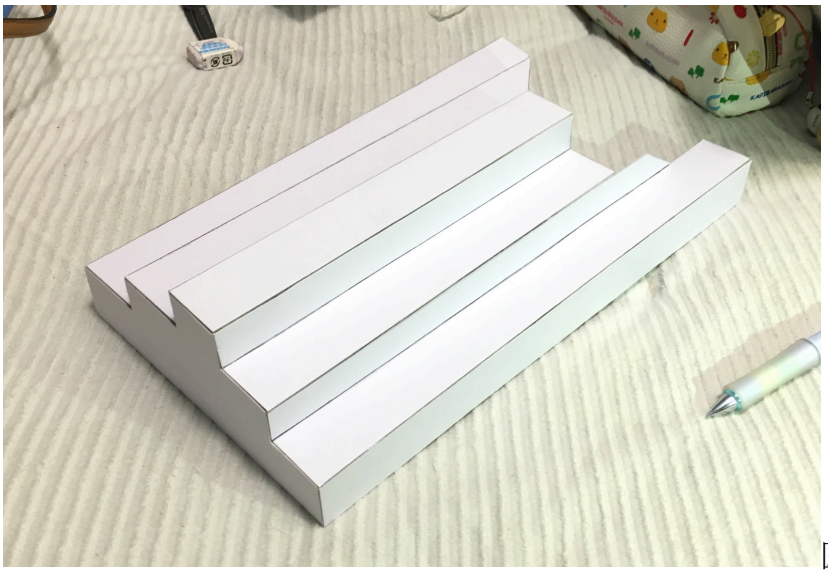


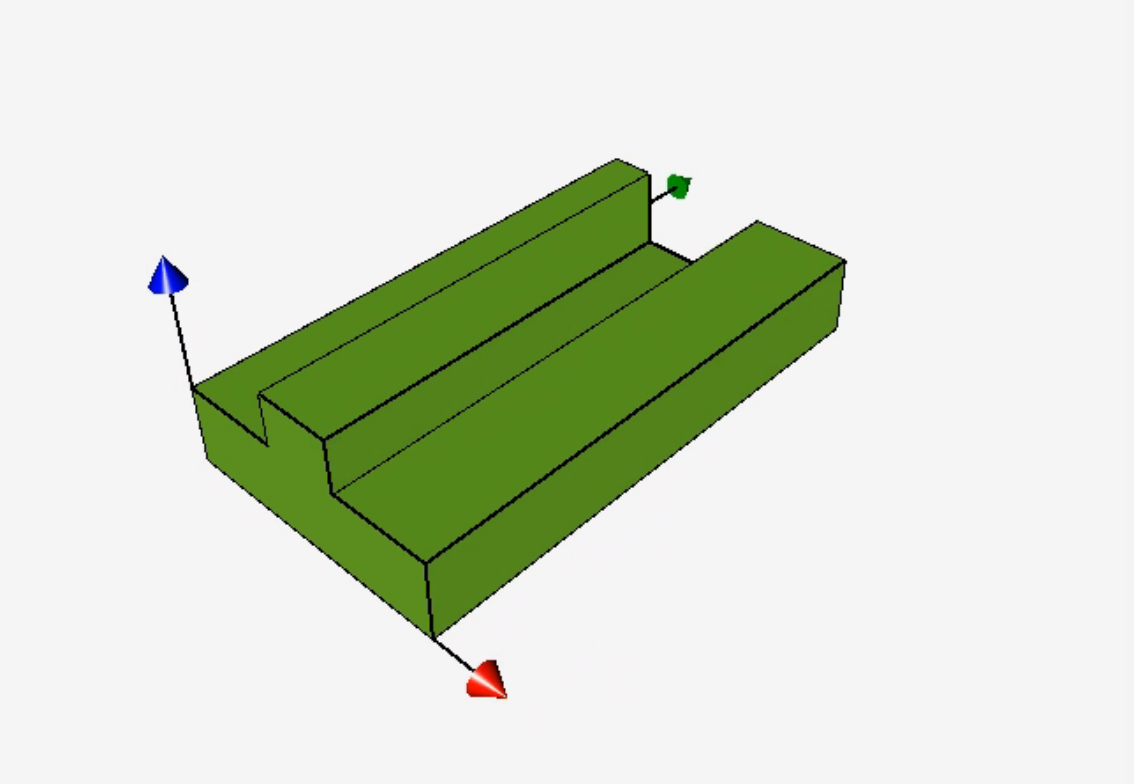
図 10.　オリジナル③

これらはだまし絵立体の一種であり、両目で見たときや少しでも視点がずれているときには錯視は起こらない。また、光の当たり方次第では、単眼立体視によって影から実際の面の向きや奥行きを推測できて錯視が起こりにくくなってしまうため、なるべくどの面にも均一に光を当てた状態で立体を見ることが望ましい。

4.2　GLSC３D

紙工作による立体では、視点を固定することや均一に光を当てることが容易でないため、オリジナル立体①と②について、GLSC3D で同じ立体を描画した。寸法は紙工作に忠実ではないが、しくみと概形は一致している。

３次元空間に長方形や正多角形でない多角形を描画する関数が用意されていなかったため、鋭角のある四角形の面は、直方体、扇形、長方形の組み合わせで描画した。画像とプログラムは以下である。



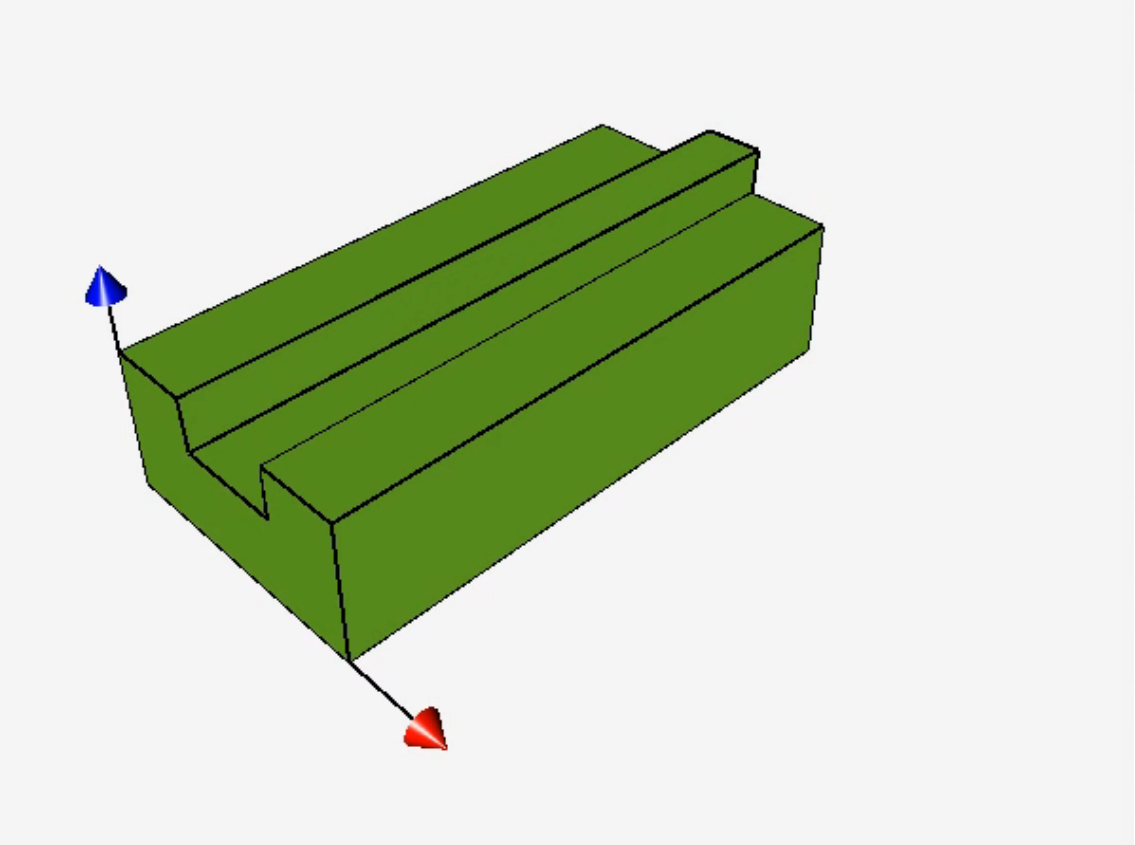
```
#include<stdio.h>
#include<math.h>
#include<glsc3d_3.h>

#define WX (800)
#define WY (600)

#define Pi (3.1415927)


int main()
{

    double x1[26]={0,11,11,7,7,4,4,0,0,
                   0,1.6,1.6,4,4,7,7,11,11,7.1,7.1,7,4,4,1.6,0,0};
    double y1[26]={0,0,0,0,0,0,0,0,0,

20,20,17.4,17.4,13.8,13.8,20,20,20,20,16.2,13.8,17.3,17.4,20,20,20};
```

```
        double z1[26]={0,0,3,3,5,5,3,3,0,
                       0,0,0,0,0,0,0,0,3,3,3,5.01,5.01,3,3,3,0};

        double x2[14]={0,0,3.9,3.9,4,4,7,7,7.1,7.1,11,11,11,11};
        double y2[14]={20,0,0,17.4,17.3,0,0,13.8,16.2,0,0,20,20,0};
        double z2[14]={3,3,3,3,5,5,5,5,3,3,3,3,0,0};

            g_init("Window", WX, WY);
            g_def_scale_3D_fix(0,
                                                    -1, 15,
                                                    -1, 23,
                                                    -1, 9,
                                                    20.0, 20.0,
                                                    WX - 40.0, WY - 40.0);

            for (int i_time = 0;; i_time++)
            {

            g_vision(0,
                        24,-26,25,
                        0,0,1,
                        1);

//              g_vision(0,
//
2*sqrt(313)*cos(i_time*0.01),2*sqrt(313)*sin(i_time*0.01),25,
//                          0,0,1,
//                          1);

                        g_cls();
                        g_sel_scale(0);
                        g_boundary();
```

```
//          g_box_3D(-1,15,-1,23,-1,9,1,0);



        g_area_color(1,0,0,1);

        g_arrow_3D(0,0,0,
                   1,0,0,
                   14,1,
                   0,1);

        g_area_color(0,1,0,1);

        g_arrow_3D(0,0,0,
                   0,1,0,
                   25,1,
                   0,1);

        g_area_color(0,0,1,1);

        g_arrow_3D(0,0,0,
                   0,0,1,
                   8,1,
                   0,1);

        g_area_color(0.6,1,0.2,1);


        g_box_3D(0,1.6,
                 0,20,
                 0,3,
                 0,1);
```

```
g_box_3D(1.6,4,
        0,17.4,
        0,3,
        0,1);

g_box_3D(4,7,
        0,13.8,
        0,5,
        0,1);

g_box_3D(7,11,
        0,20,
        0,3,
        0,1);


g_fan_3D(7,16.2,3,
        0,-6-sqrt(61),5,
        0.4*sqrt(61),atan(5.0/6),Pi/2,
        0,1);

g_fan_3D(4,17.4,5,
        5,-6-sqrt(61),0,
        0.6*sqrt(61),atan(5.0/6),Pi/2,
        0,1);

g_fan_3D(4,17.4,3,
        -1-sqrt(2),1,0,
        2.4*sqrt(2),Pi/4+0.1,Pi/2,
        0,1);

g_rectangle_3D(4,15.6,3.9,
            0,1,0,
```

```
                    3.6,2.2,0,
                    0,1);

        g_polyline_3D(x1,y1,z1,26);
        g_polyline_3D(x2,y2,z2,14);



                g_finish();
        }
        return 0;
}
```



```
#include<stdio.h>
#include<math.h>
#include<glsc3d_3.h>
```

```
#define WX (800)
#define WY (600)
#define Pi (3.1415927)


int main()
{

    double x1[19]={0,10,10,7,6.9,3.1,3,0,0,
                   0,10,10,7,6.9,3.1,3,3,0,0};
    double y1[19]={0,0,0,0,0,0,0,0,0,
                   20,20,20,20,20,25.5,22.5,20,20,20};
    double z1[19]={0,0,5,5,3.1,3.1,5,5,0,
                   0,0,5,5,3.1,3.1,5,5,5,0};

    double x2[14]={0,0,3,3,3.1,3.1,6.9,6.9,7,7,10,10,10,10};
    double y2[14]={20,0,0,22.5,25.5,0,0,20,20,0,0,20,20,0};
    double z2[14]={5,5,5,5,3.1,3.1,3.1,3.1,5,5,5,5,0,0};

        g_init("Window", WX, WY);
        g_def_scale_3D_fix(0,
                                                -1, 15,
                                                -1, 23,
                                                -1, 9,
                                                20.0, 20.0,
                                                WX - 40.0, WY - 40.0);

        for (int i_time = 0;; i_time++)
        {

//          g_vision(0,
//                   25,-25,25,
```

```
//                          0,0,1,
//                          1);

            g_vision(0,

25*sqrt(2)*cos(i_time*0.01),25*sqrt(2)*sin(i_time*0.01),25,
                          0,0,1,
                          1);

                        g_cls();
                        g_sel_scale(0);
                        g_boundary();
//              g_box_3D(-1,15,-1,23,-1,9,1,0);



            g_area_color(1,0,0,1);

            g_arrow_3D(0,0,0,
                          1,0,0,
                          14,1,
                          0,1);

            g_area_color(0,1,0,1);

            g_arrow_3D(0,0,0,
                          0,1,0,
                          25,1,
                          0,1);

            g_area_color(0,0,1,1);

            g_arrow_3D(0,0,0,
```

```
                0,0,1,
                8,1,
                0,1);

g_area_color(0.6,1,0.2,1);


g_box_3D(0,3,
            0,20,
            0,5,
            0,1);
g_box_3D(3,7,
            0,20,
            0,3,
            0,1);

g_box_3D(7,10,
            0,20,
            0,5,
            0,1);


g_fan_3D(3,25.5,3,
            8,-11-sqrt(185),0,
            0.5*sqrt(185),atan(8.0/11),Pi/2,
            0,1);

g_fan_3D(3,25.5,3,
            0,-3-sqrt(13),2,
            sqrt(13),atan(2.0/3),Pi/2,
            0,1);

g_rectangle_3D(3,21,4,
```

```
                    0,1,0,
                    3,2,0,
                    0,1);

        g_polyline_3D(x1,y1,z1,19);
        g_polyline_3D(x2,y2,z2,14);



            g_finish();
        }
        return 0;
}
```

第５章　まとめ

錯視とこれまでに作られてきた多様な不可能立体のしくみについて学び、独自の新しい不可能立体を作ることができた。とはいえ、直感的にできるような初歩的な手法しか用いることができていないため、より高度な数理学を用いて、さらに新しいパターンの立体錯視を発見することを目標に、修士課程で研究を続けていきたい。そのためには 3D プリンターなどの機器を使いこなせることが不可欠になるので、PC 上での立体の描画技術をはじめとするプログラミングスキルを向上させなくてはならない。

また、私たちの身の周りでは、意図しない場面で錯視が起こってしまうこともあり、そのような錯視がときに交通事故などを引き起こすことが問題にもなっている。これからは、芸術としての錯視の効果を拡大する手法に加え、そのような意図しない場面での錯視を軽減する手法についても合わせて研究したい。

第６章　参考文献

杉原厚吉．立体イリュージョンの数理．　共立出版, 2006, 182p

杉原厚吉. “だまし絵の数理−人の視覚とロボットの視覚−”. 数学. 2002, 54, No.1, p.58-68

Kokichi, Sugihara. “Spatial realization of Escher’s impossible world”. Asia Pac. Math. Newsl. 1 (2011), no. 1, 1-5.

Kokichi Sugihara’s Homepage (Japanese). http://home.mims.meiji.ac.jp/~sugihara/Welcomej.html, (参照 2018-02-07)