MASTER 1 MMAA

Projet C++

CISSY
THOMAS
MEHDI

Programme

COMPRÉHENSION DU SUJET

CREATION DES STRUCTURES

DESSIN DE DROITE

SUPPRESSION DE DROITES PROCHES

ESPACE DE HOUGH ET TRACERS

Contraintes du rendu

Le but du projet est d'implanter en langage C++ un programme permettant de détecter des droites et des portions de droites sur une image couleur au format ppm (format P3), voir

L'équipe

usamippiu/
Projet_C_M1_MMAA

1 Contributor | 0 Issues | 0 Stars | 1 Fork

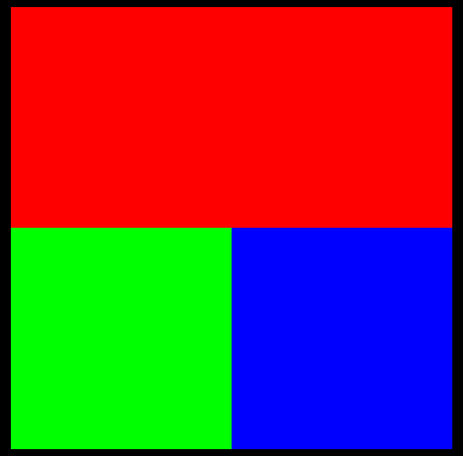
usamippiu/Projet_C_M1_MMAA

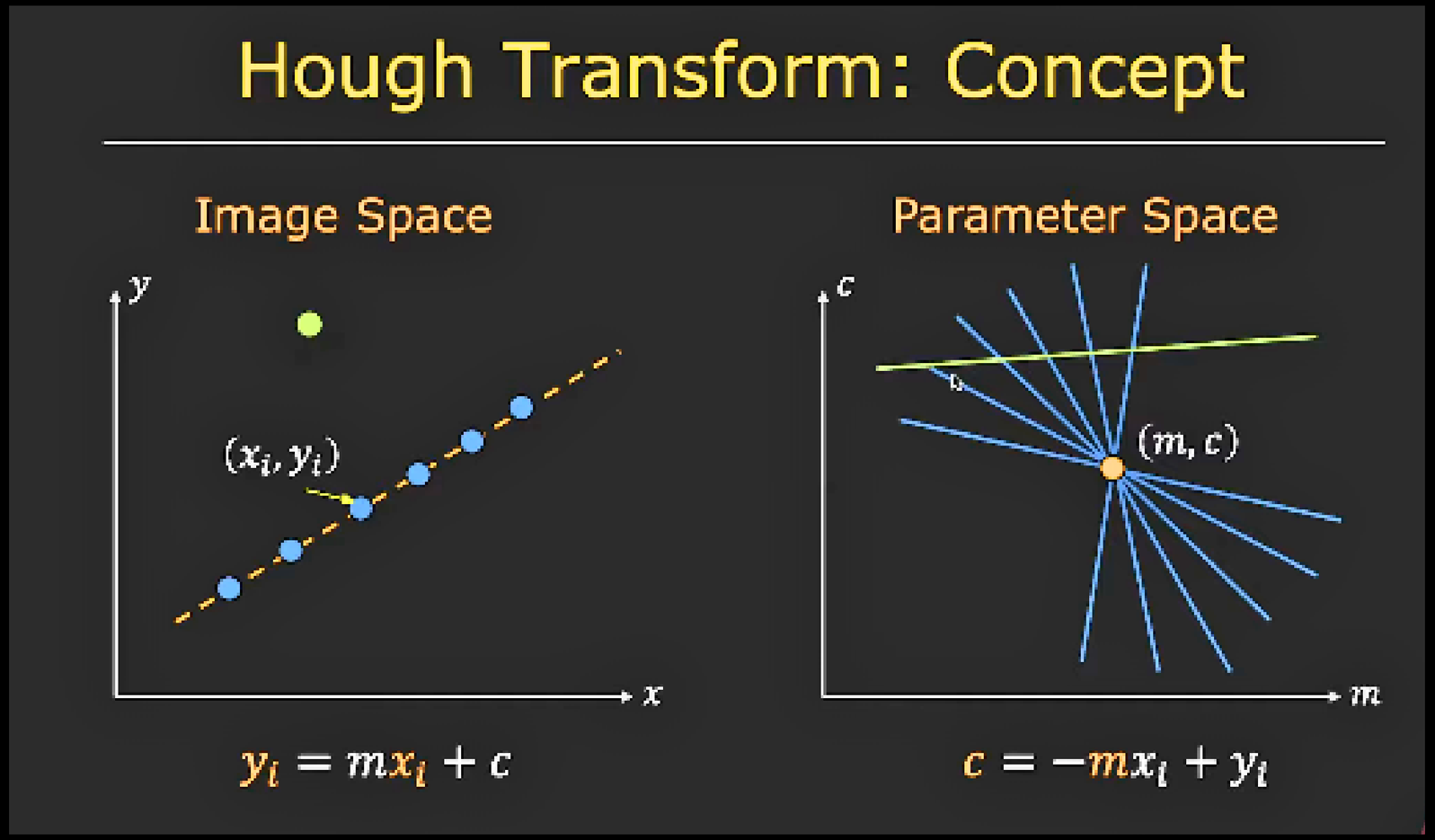
Contribute to usamippiu/Projet_C_M1_MMAA development by creating an account on GitHub.

GitHub

Compréhension du sujet

Retourner au programme





APERÇU DES CLASSES

```
P3
# resolution
2 2
# avec 255 comme val max
255
# debut de l image
255
0
0
255
0
0
0
255
0
0
0
255
```

```cpp
struct Pixel {
    unsigned int r, g, b;
    void setRGB(unsigned int r, unsigned int g, unsigned int b);
    std::vector<unsigned int> getPixel();

};
```

Aperçu des classes

```cpp
// Indice 1 -0 = en haut à gauche de l'image etc
class PPMImage {
private:
    unsigned int width, height, maxColor;

public:
    std::vector<Pixel> pixels; // Attribut matrice de pixel
    void PPMImageLoader(const std::string& filename); // Charger et initialiser l'image
    // Getters
    int getWidth() const;
    int getHeight() const;
    int getMaxColor() const;

    // Affichage et conversion matrice - image
    void afficheMatrice(std::vector<std::vector<Pixel>>& input);
    std::vector<std::vector<Pixel>> convertTo2D();
    void MatriceToImage(std::vector<std::vector<Pixel>>& input, std::string filename);

    // Tracer de droite et traitement pour espace de Hough
    std::vector<float> equationDroite(float x1, float y1, float x2, float y2);
    //void tracerDroite(std::vector<float> eqDroite, std::vector<std::vector<Pixel>>& input);
    std::vector<float> equationDroitePolaire(float x, float y, float rho, float theta);
    void tracerDroite(std::vector<float> eqDroite, std::vector<std::vector<Pixel>>& input);
    void tracerSegment(std::vector<float> eqDroite, std::vector<std::vector<Pixel>>& input);
    std::vector<std::tuple<double, double>> getNotWhite(std::vector<std::vector<Pixel>>& input);
    std::vector<std::tuple<double, double>> getLignes(std::vector<std::vector<Pixel>>& input, int nb_intersections, double threshold);
    std::pair<int, int> getMaximum(const std::vector<std::vector<Pixel>>& input); // Avoir les coordonnées du maximum de la matrice de pix

};
```

DESSIN DE DROITES

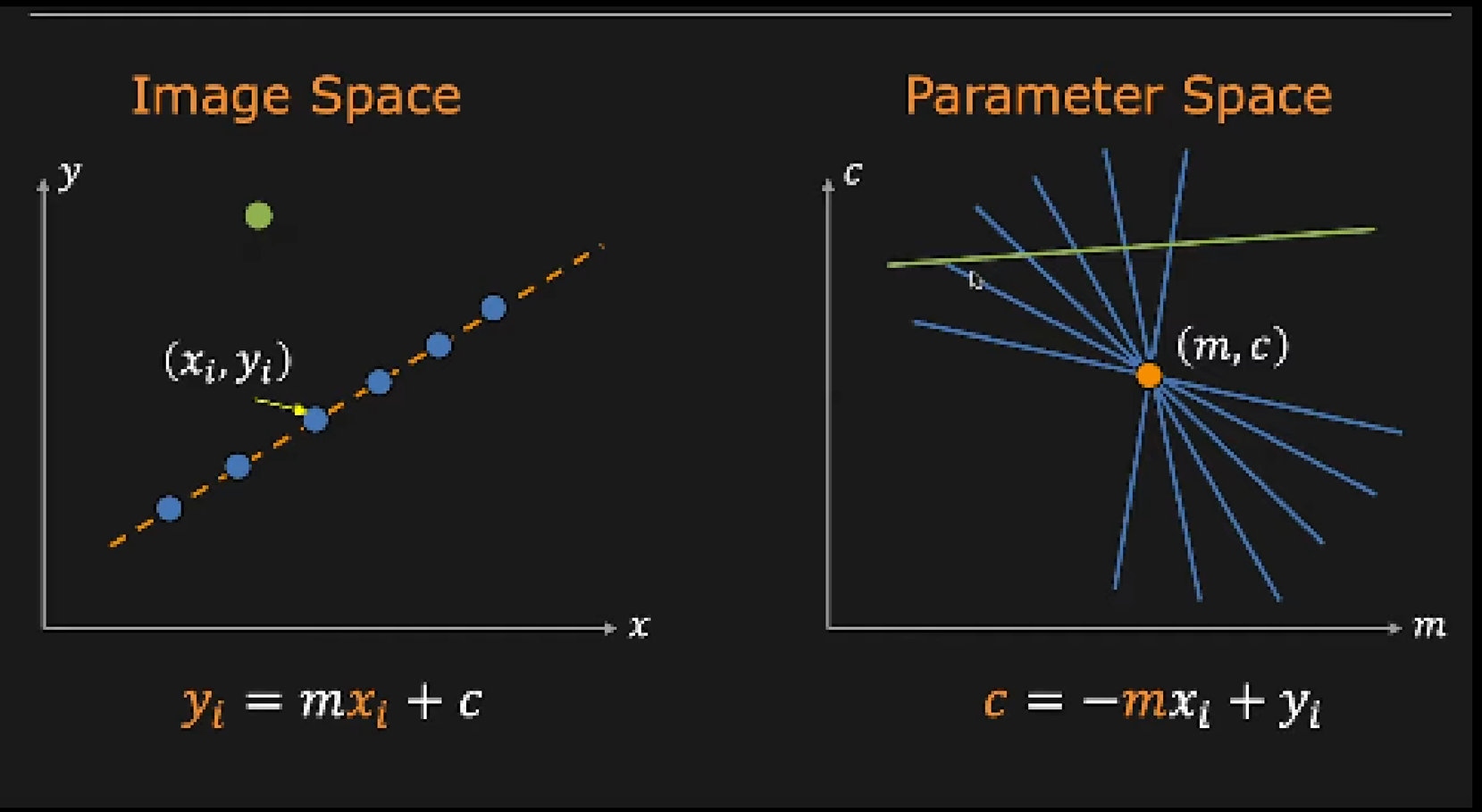
```cpp
void PPMImage::tracerDroite( std::vector<float> eqDroite, std::vector<std::vector<Pixel>>& input )
{
    for (unsigned int i = 0; i < height; i++)
    {
        float y = eqDroite[0] * i + eqDroite[1];
        if(y >=0){
            for (unsigned y_rounded = std::floor(y); y_rounded <= std::ceil(y); ++y_rounded) {
                double percent_y = 1 - std::abs(y - y_rounded);
                if (y_rounded < width)
                {
                    int col = std::round(255*(1-percent_y));
                    input[i][y_rounded].setRGB(col, col, col);
                }
            }
        }
    }
}
```

SUPPRESSION DE DROITES

```cpp
void remove_close_tuples(std::vector<std::tuple<double, double>>& vec, double threshold) {
    std::vector<std::vector<std::tuple<double, double>>> close_tuples_groups;
    for (const auto& tuple : vec) {
        bool grouped = false;
        for (auto& group : close_tuples_groups) {
            if (tuple_norm(group.front(), tuple) < threshold) {
                group.push_back(tuple);
                grouped = true;
                break;
            }
        }
        if (!grouped) {
            close_tuples_groups.push_back({tuple});
        }
    }

    vec.clear();
    for (const auto& group : close_tuples_groups) {
        vec.push_back(average_tuple(group));
    }
}
```

Méthode naïve



- la valeur d'une «case» va s'incrémenter lorsque la droite correspondante traversera le point concerné.

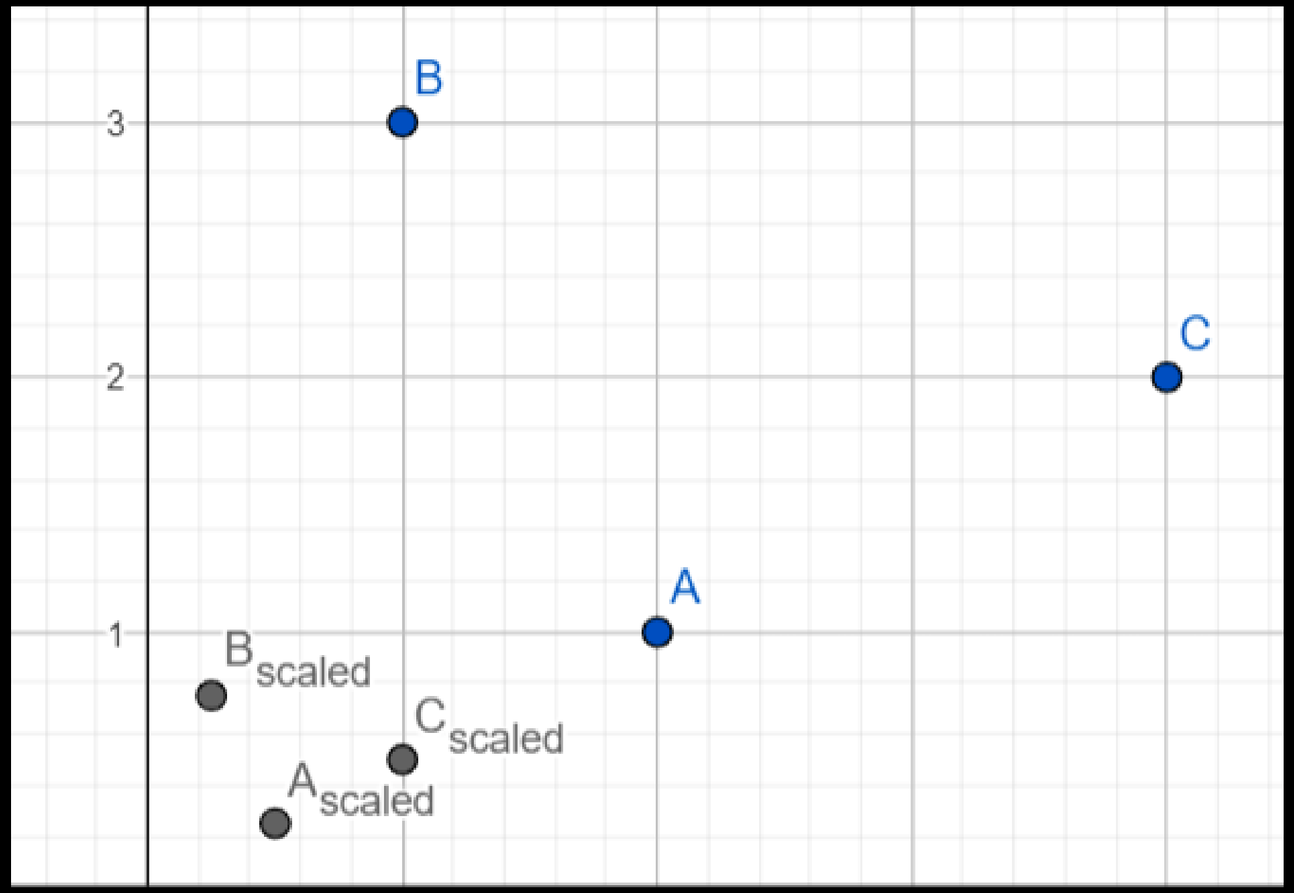
a/b	-4	-3	-2	-1	0	1	2	3	4
0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0
2	0	0	0	0	1	0	0	0	0
3	0	0	1	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0

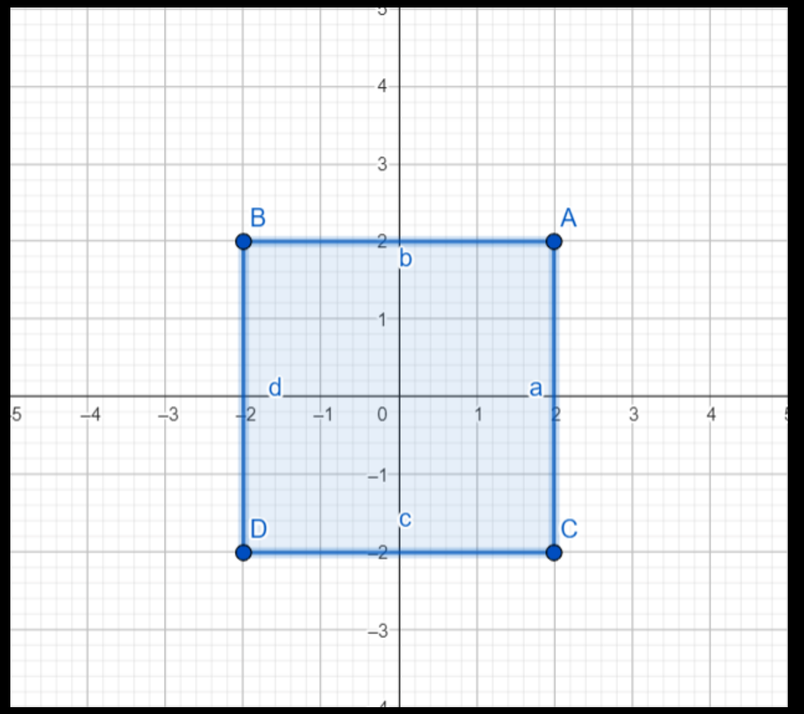
Tableau 2: Accumulation après calcul pour le point A(2,4)

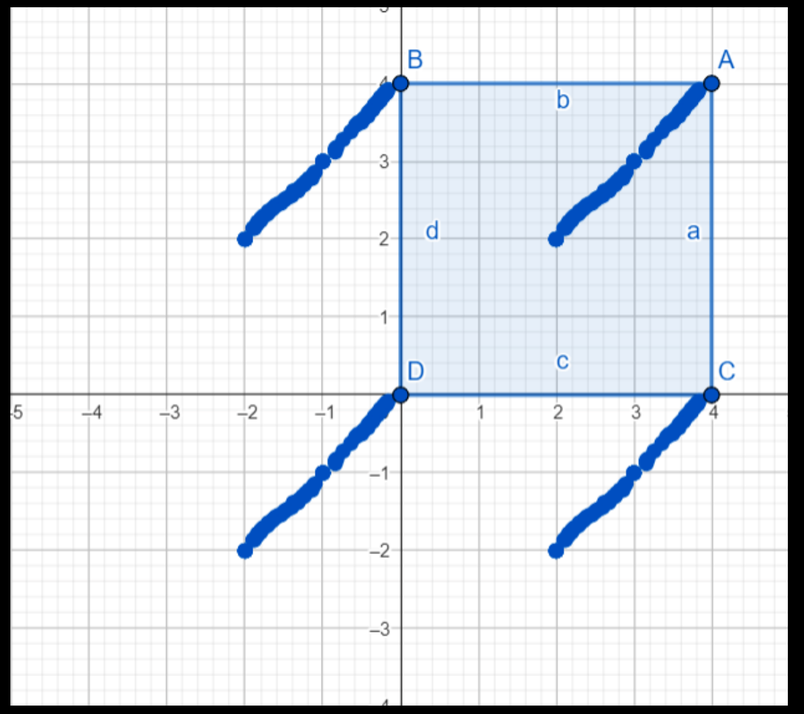
Démo

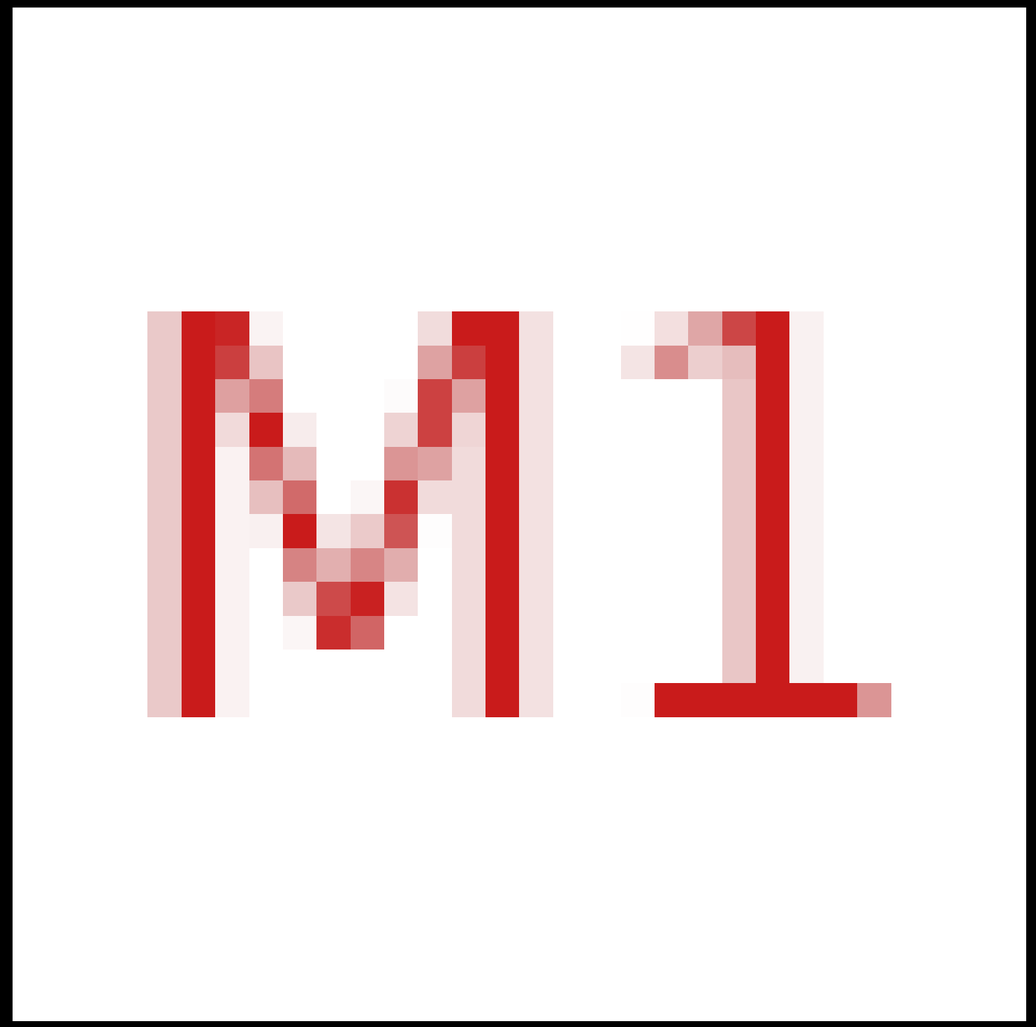
Geogebra

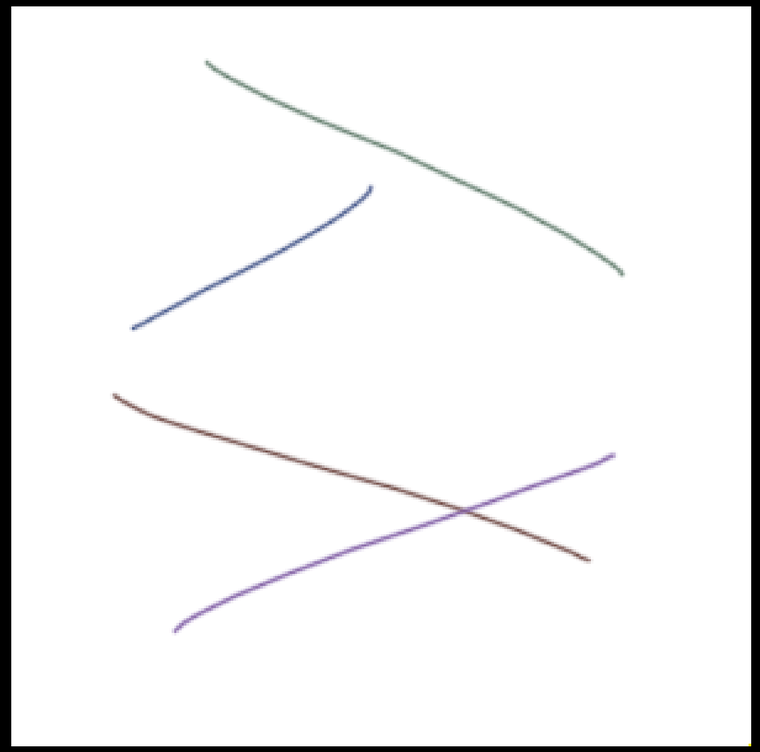
Rescaling et déplacement

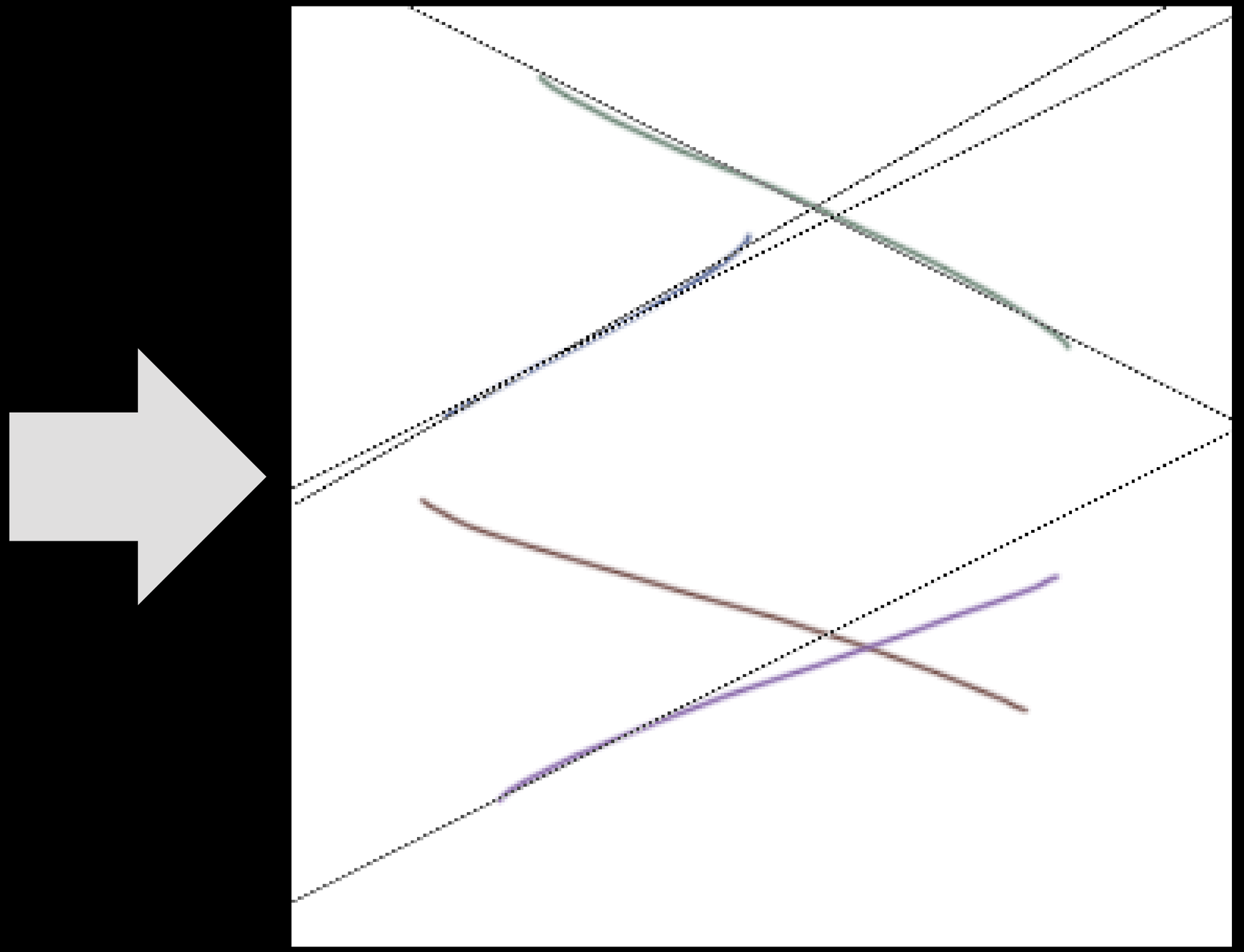


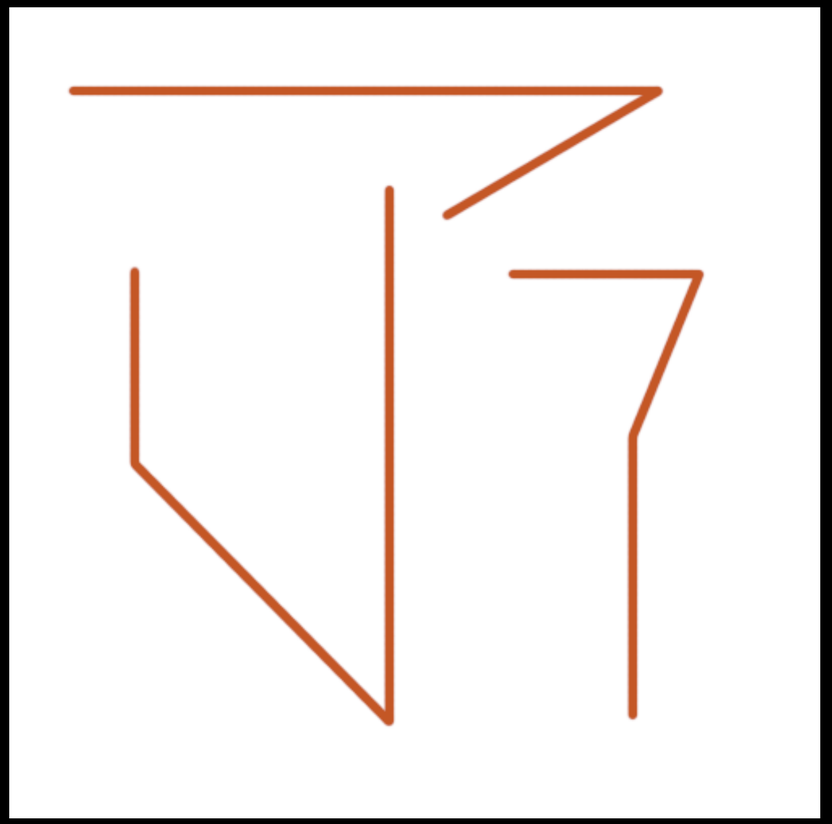


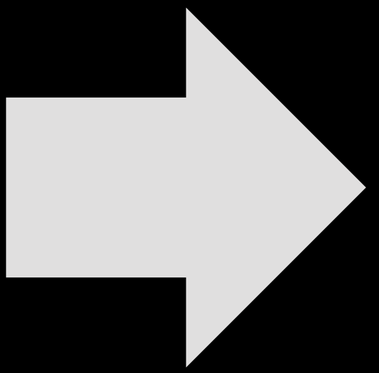


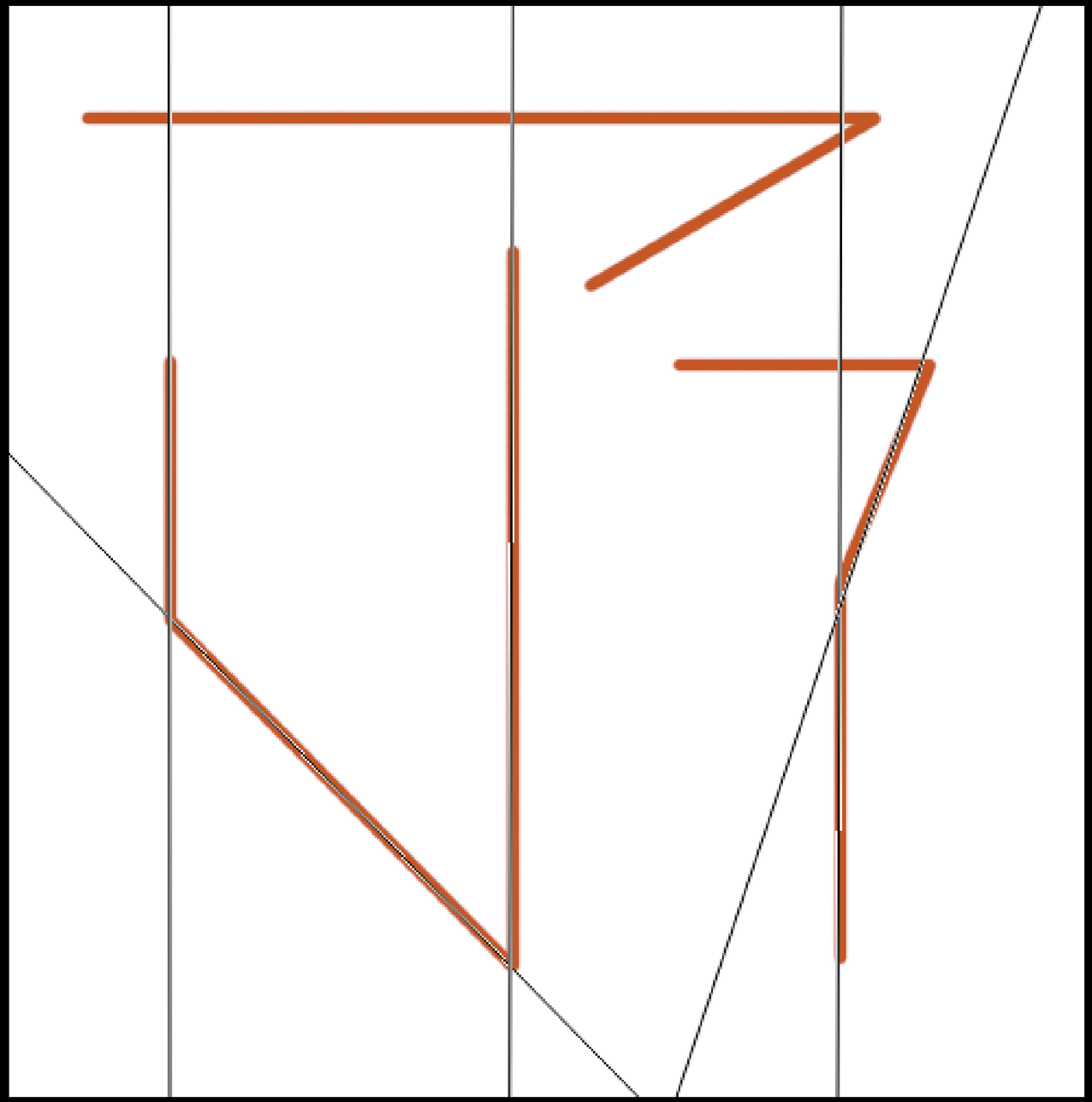

M1



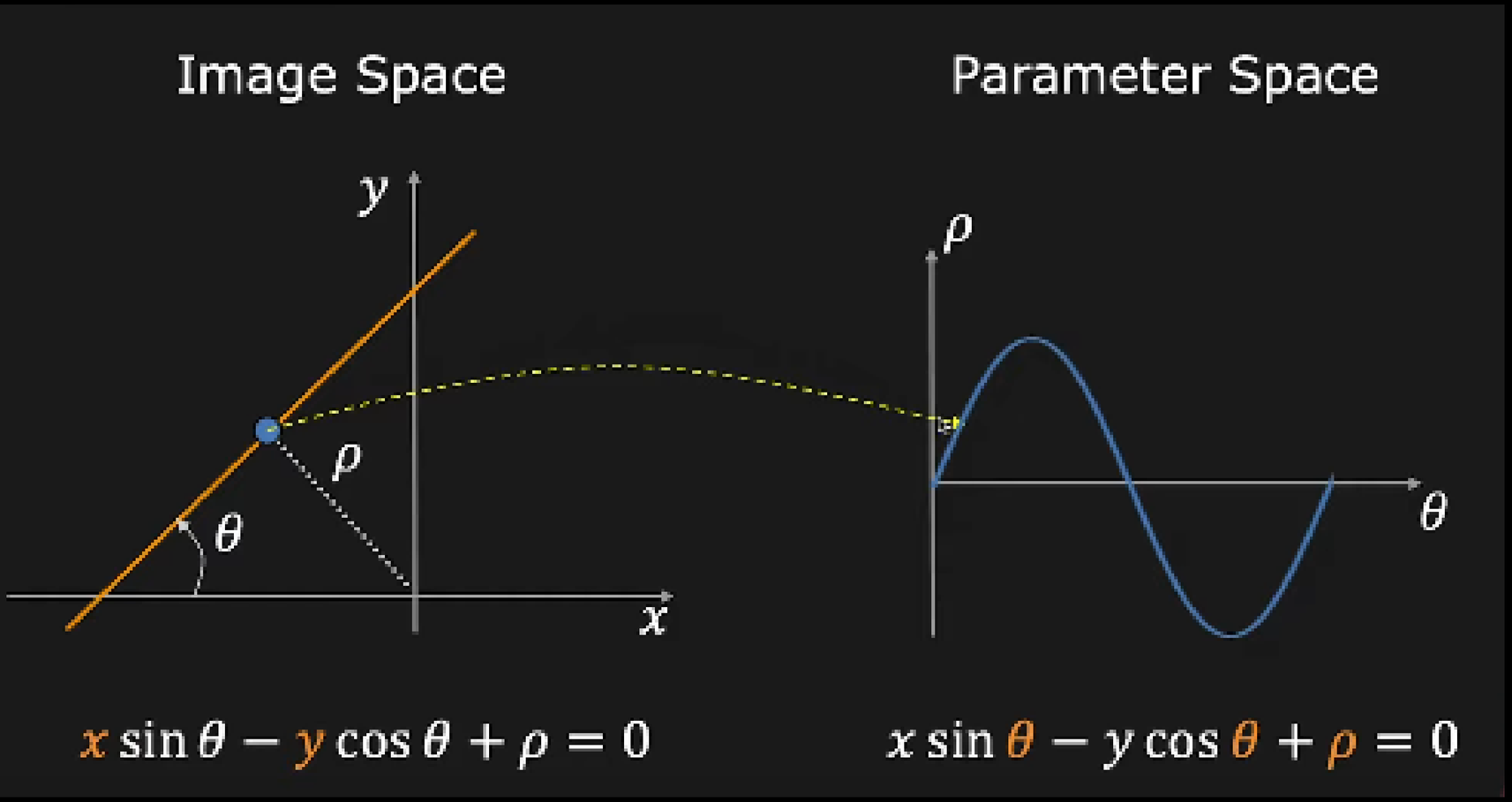


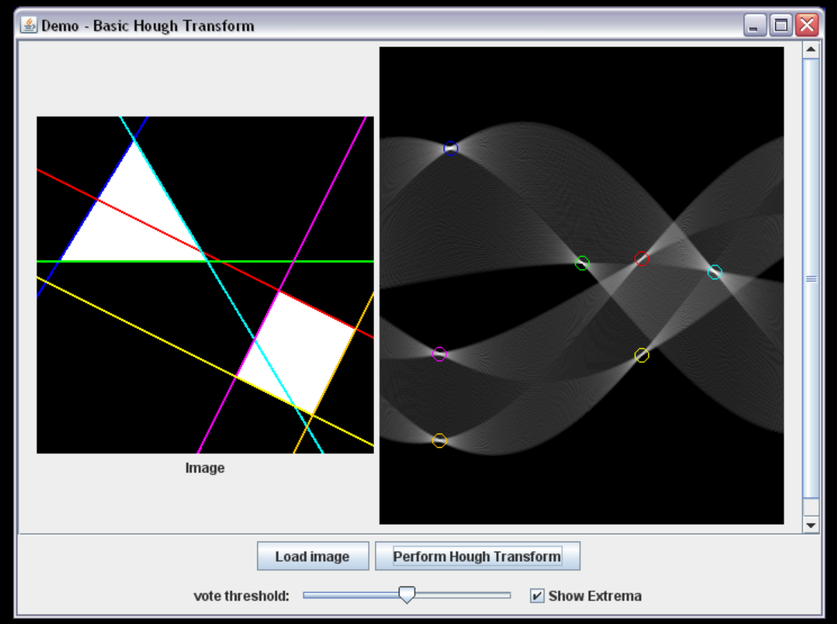






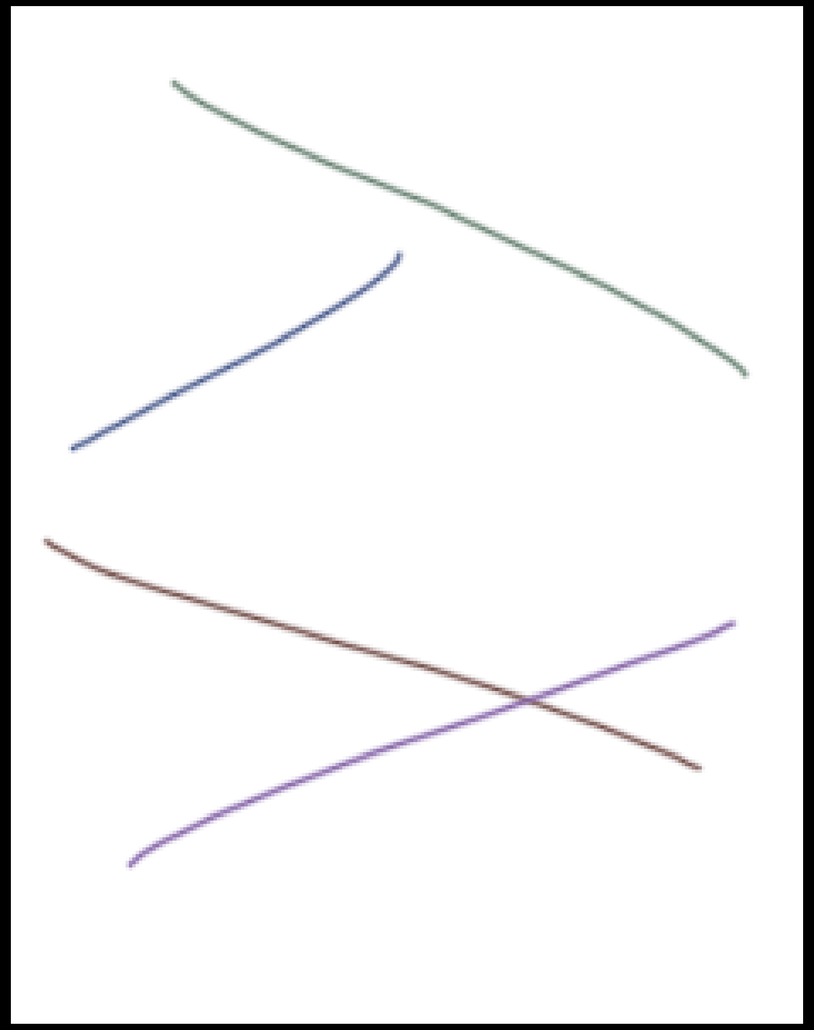
Méthode moins naïve

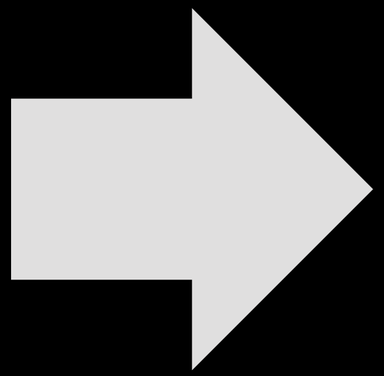


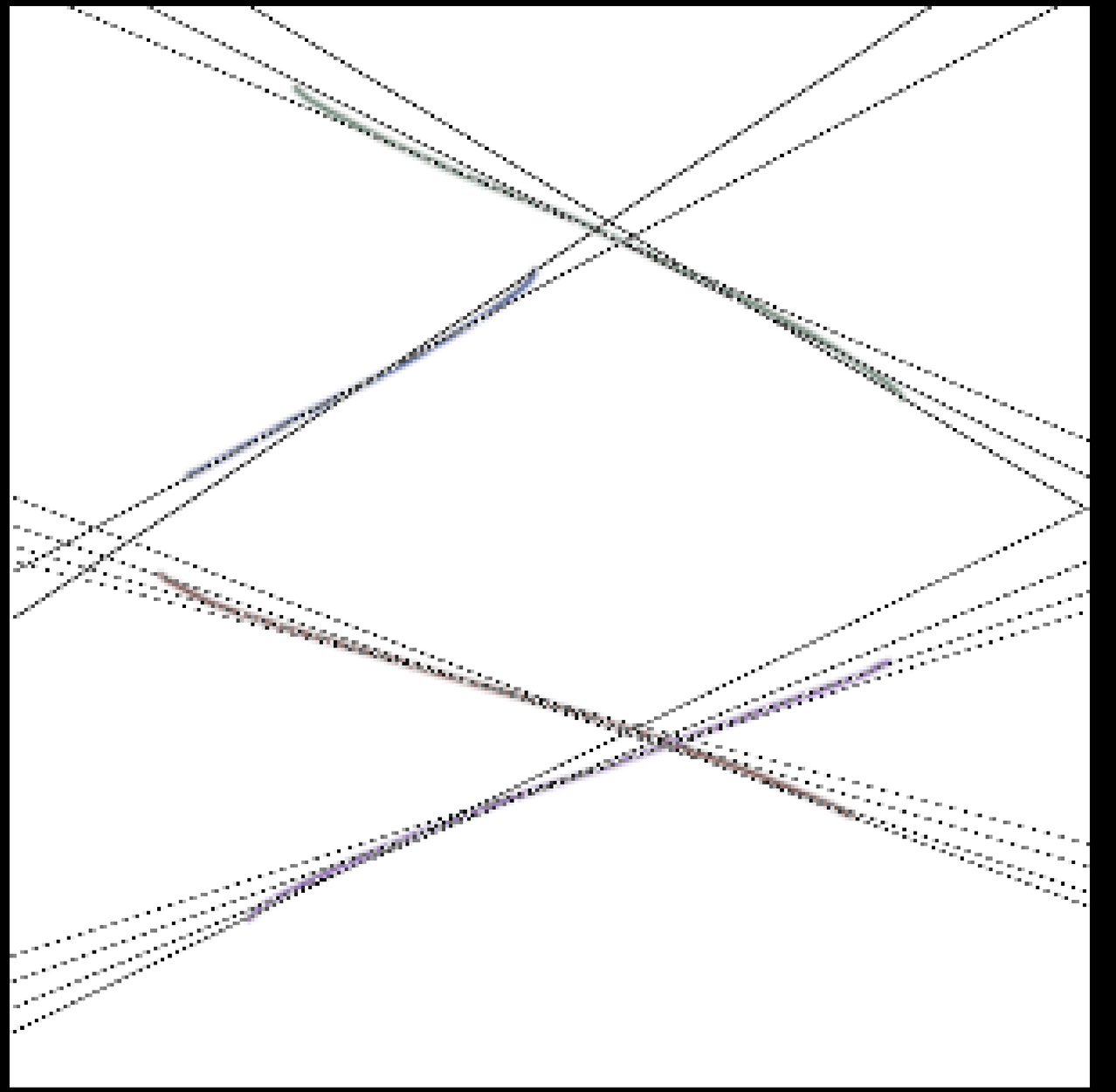


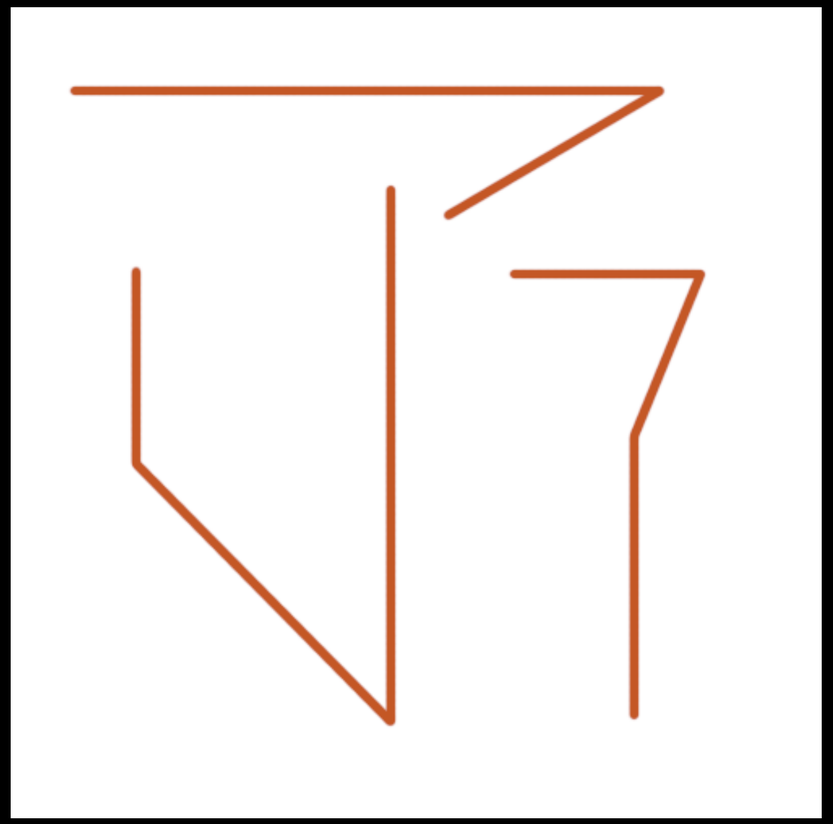
Démo

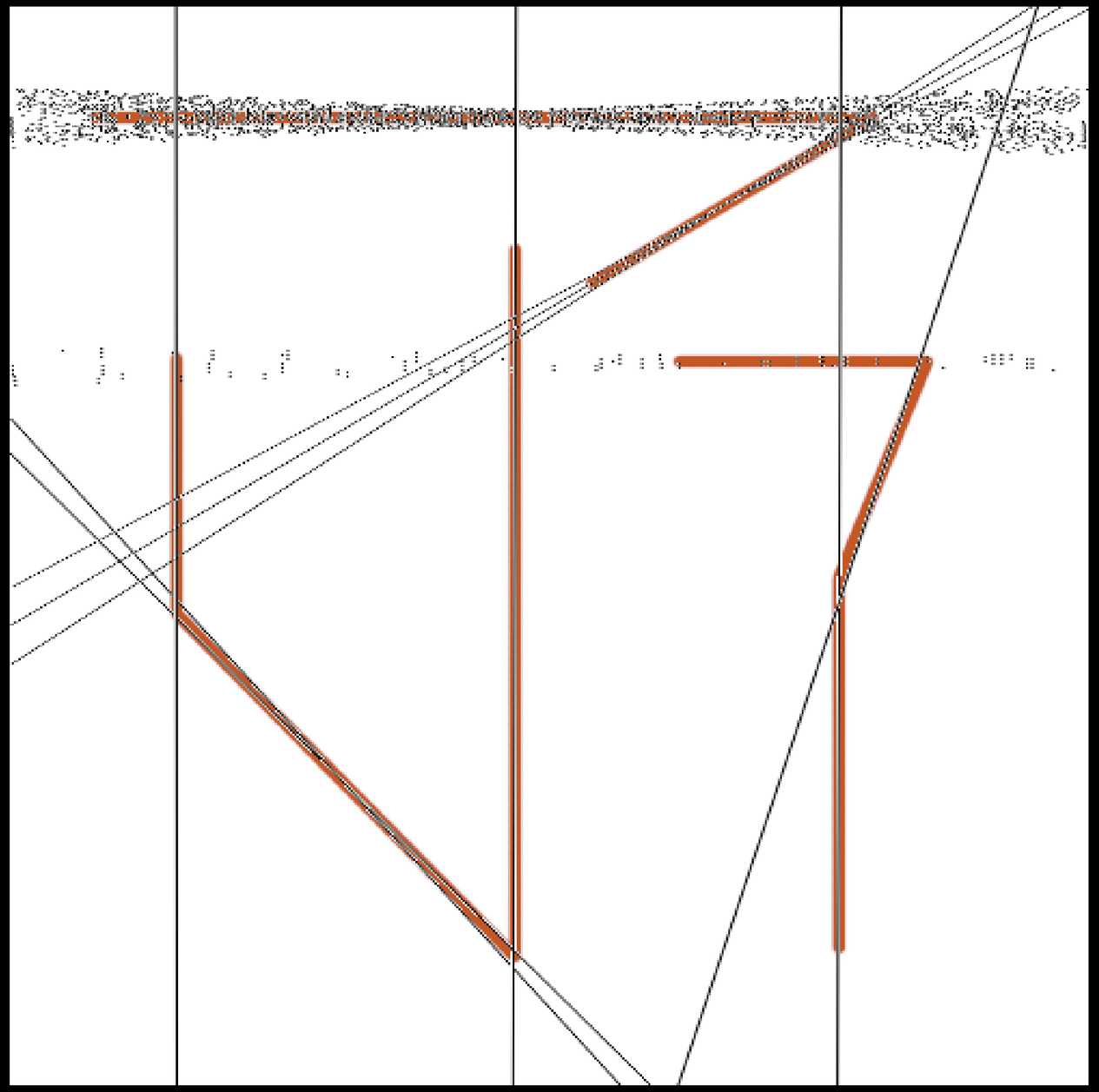
Geogebra

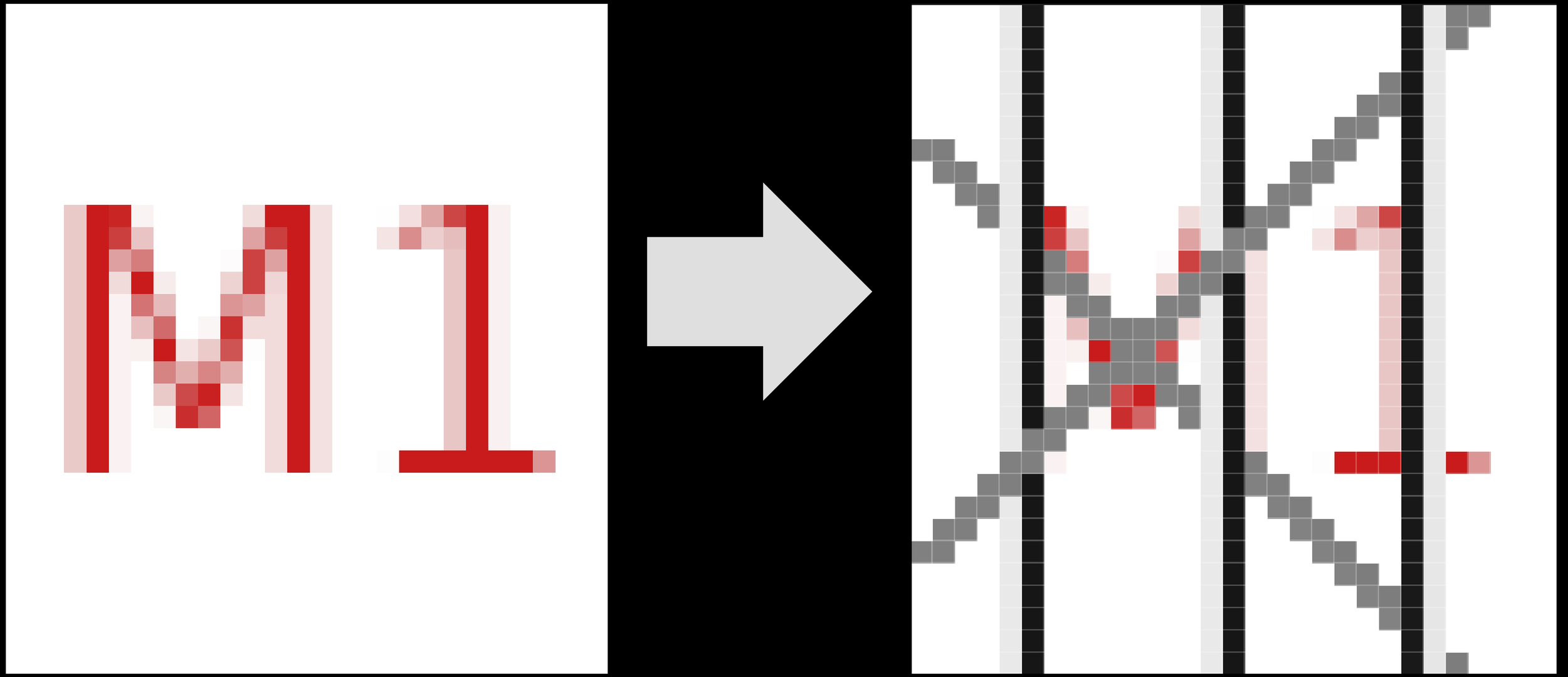











M1

Issue: Slope of the line $-\infty \leq m \leq \infty$

- Large Accumulator
- More Memory and Computation

Solution: Use $x \sin\theta - y \cos\theta + \rho = 0$

- Orientation θ is finite: $0 \leq \theta < \pi$
- Distance ρ is finite