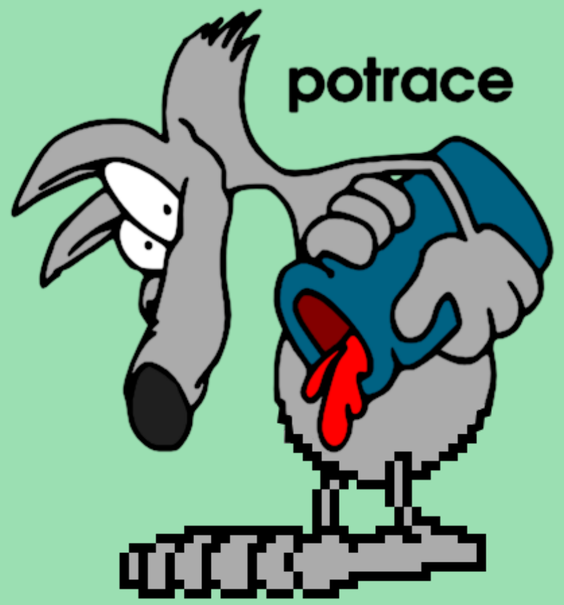
Università degli studi di Modena e Reggio Emilia

# Porting in Java dell'algoritmo di Potrace

**Candidato: Lorenzo Racca**

**Relatore: Prof. Riccardo Martoglia**

# Ambito applicativo




Gratuito e open source


## Grafica Raster

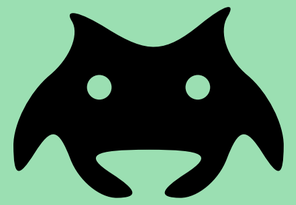
- Griglia di pixel

## Grafica Vettoriale

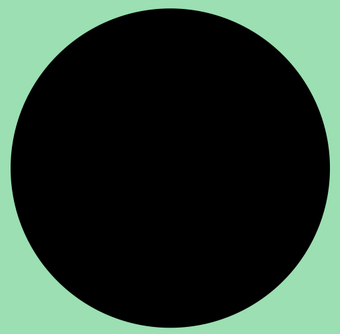
- Composizione di forme geometriche base

# Limiti di Potrace

- Tracing di immagini a colori
- Gestione delle immagini a bassa risoluzione



- Riconoscimento delle forme



```
<circle cx=”500” cy=”500” r=” 300” stroke-width=”3”
                fill=”black”/>
```

circa 90 byte

```
<path d=¨M331.000 72.582 C 329.075 72.801,322.550 ...¨
          stroke=¨none¨ fill=¨rgb(0,0,0)/>
```

circa 900 byte

# Il progetto

Costruzione di una libreria Java per il tracing

## Porting

- Gestione Immagini in bianco e nero

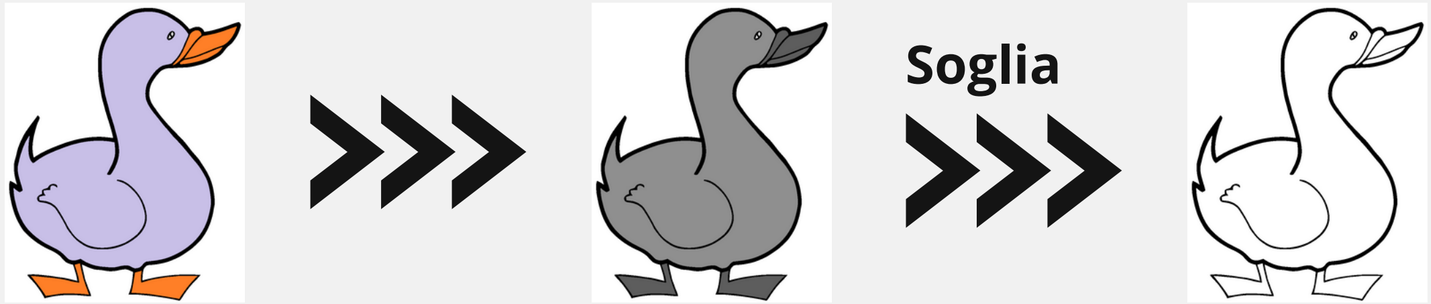
## Feature aggiuntive

- Tracing di immagini a colori
- Gestione delle immagini a bassa risoluzione

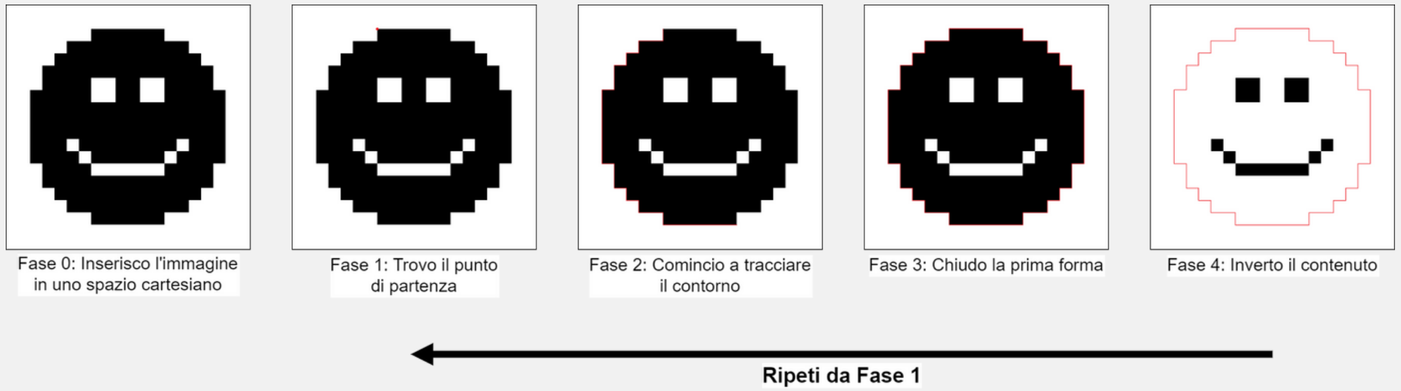
# Contenuti

# L'algoritmo base

**FASE 1:** Binarizzazione




**FASE 2:** Estrazione dei percorsi

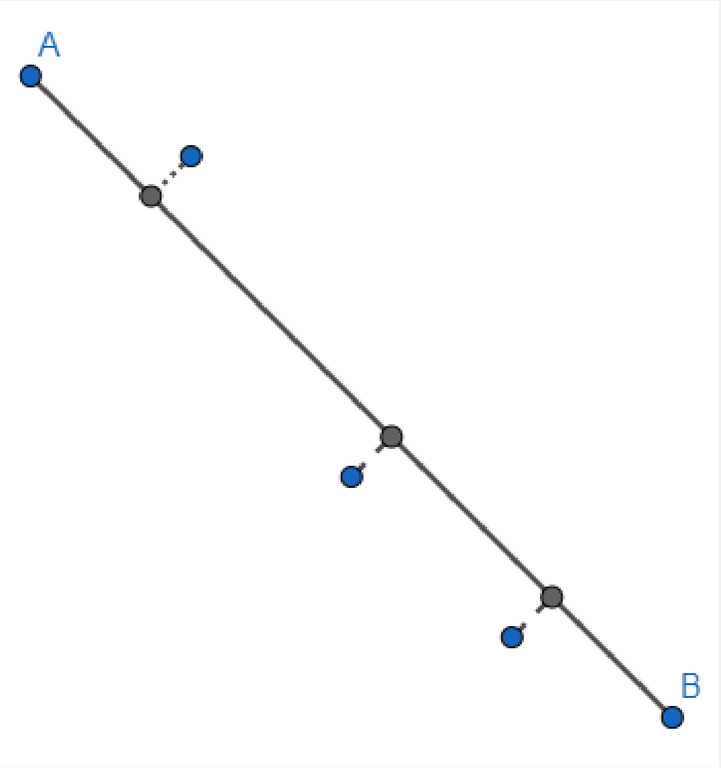


# L'algoritmo base

**FASE 3:** Costruzione dei poligoni

- Ricerca del poligono migliore
  - Meno vertici
  - A parità vertici, minor penalità

$$P_{i,j} = |v_j - v_i| \cdot \sqrt{\frac{1}{j \ominus i + 1} \sum_{k=i}^{j} dist(v_k, \overline{v_i v_j})^2}$$

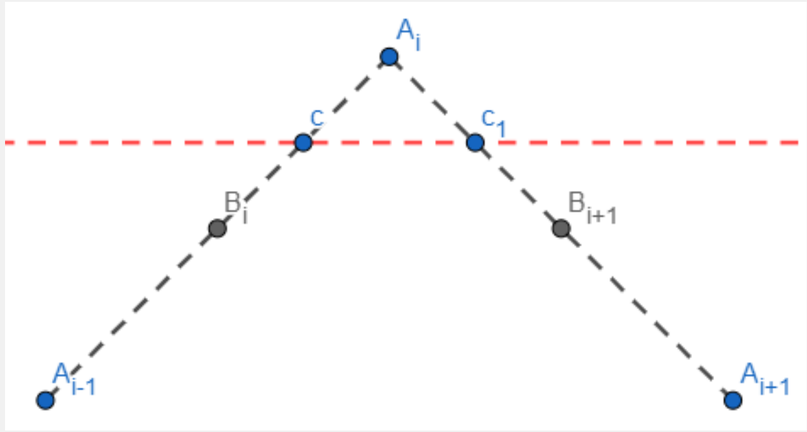


# L'algoritmo base

## **FASE 4:** Costruzione delle curve

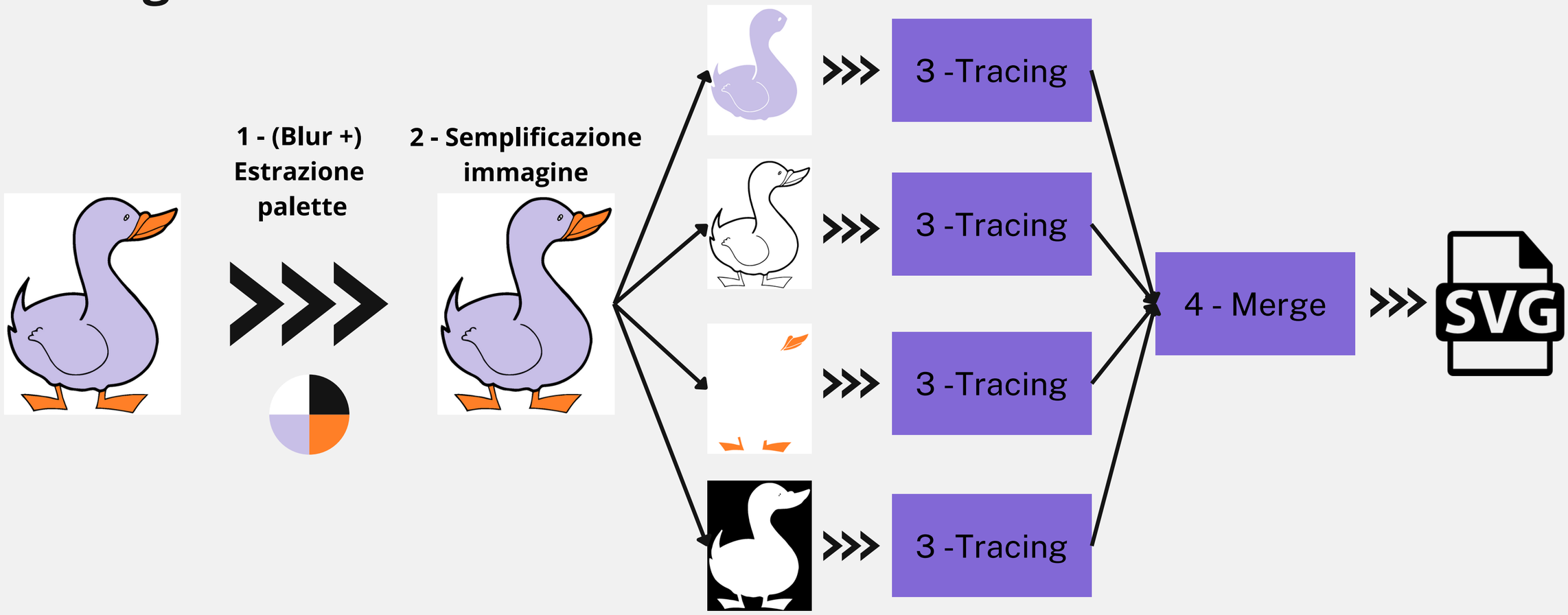
Capire se a cavallo di ogni punto si possa disegnare una curva o una coppia di segmenti

- Per ogni punto **A(i)**
  - Trovo **B(i)** e **B(i+1)**, punti medi tra **A(i)** e i due punti adiacenti
  - Dati **B(i)** e **B(i-1)** trovo **c**, punto di controllo della potenziale curva
  - Se la distanza tra **c** e **A(i)** è minore di **1** disegno una curva, altrimenti due segmenti



# Immagini a colori

## Strategia base



# Immagini a colori Estrazione della palette

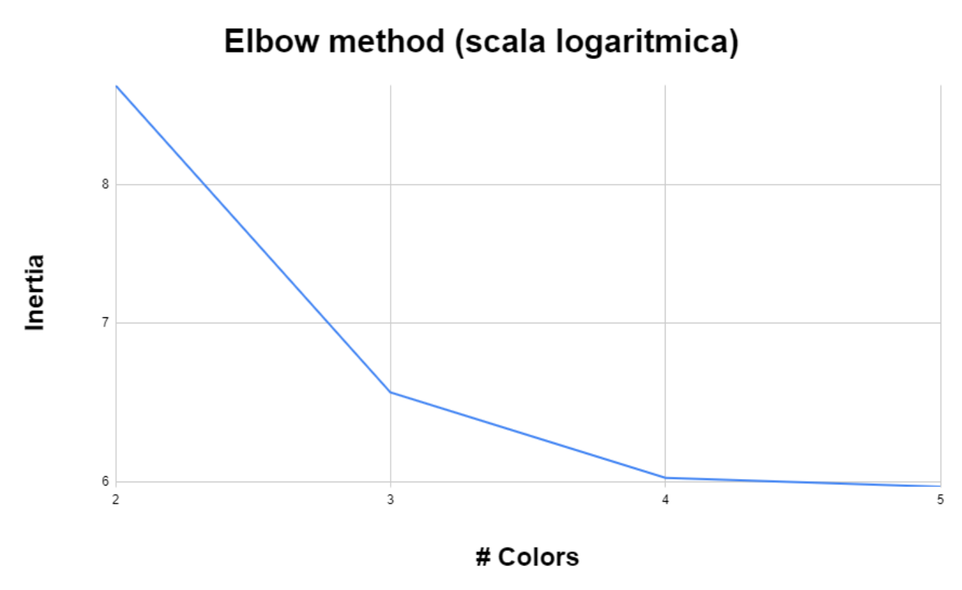
## K-Means


**DUE OPERAZIONI**

1. Associazione centroide-pixel
2. Centraggio dei centroidi

# Immagini a colori Estrazione della palette

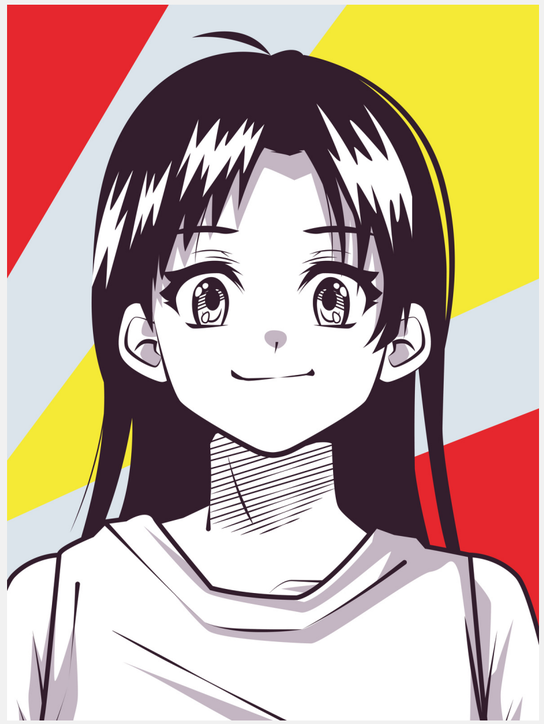
## K-Means con Elbow Method

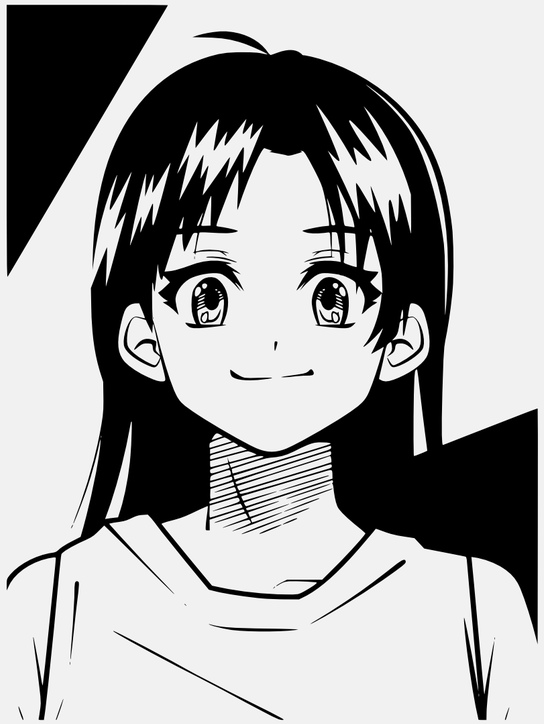


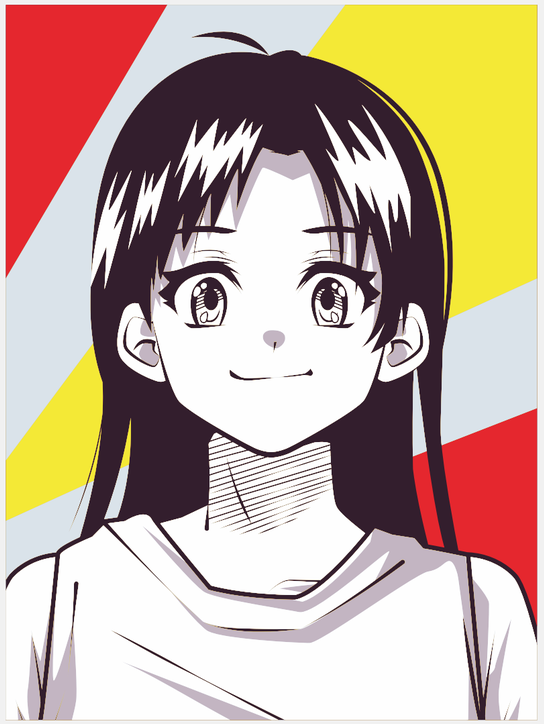

- Partendo con **N = 2**
  - Estraggo una palette da **N** colori con K-Means
  - Calcolo l'**inerzia** come la media delle distanze dei pixel dal relativo **centroide**
  - Se la differenza tra l'inerzia tra la palette con **N-1** colori e quella attuale è minore di **K**. Mi fermo a **N-1** colori
  - Altrimenti provo con un colore in più

# Test efficacia

## Caso Ideale



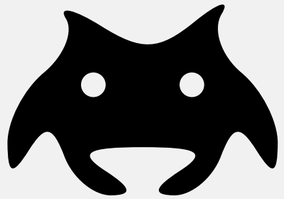




# Test efficacia

## Confronto con Potrace

Immagine Originale



Potrace

Il mio progetto

### Potrace

- Solo Bianco e Nero
- Problemi con le pixelart
- Integrabile con chiamate all'eseguibile o modificando il sorgente

### Il mio progetto

- Bianco e nero e a colori
- Fix problema pixelart
- Integrabile con semplici chiamate di funzioni
- Possibilità di modificare il flusso di processo

```
Conversion conversion = new BinaryConversion(threshold, settings);
String svg = conversion.convert(img, scale);
```

# Conclusioni e sviluppi futuri

Il progetto è gratuitamente disponibile, anche in versione eseguibile, su GitHub.
In futuro quando la libreria sarà più completa si prevede la pubblicazione anche per l'inclusione tramite Maven.

## Feature future

- Riconoscimento delle forme base e alleggerimento del file SVG in output.
- Aggiunta di parallelismo all’interno dei processi
- Costruzione di un algoritmo di merge dei livelli colore che garantisca vettoriali più smooth
- Ampliamento della GUI per trasformare la libreria in un software più utilizzabile e user friendly

# Grazie per l'attenzione