Odisee Hogeschool
Gebroeders de Smetstraat 1, 9000 Gent

Applied Programming Project 2:

Clipart Identificatie

Steven Impens

Elektronica-Ict
2017-2018

Inhoudsopgave

Codefragmentenlijst

Figurenlijst

Inleiding

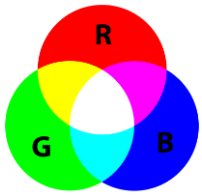
Het doel van dit project is een programma ontwerpen die clipart kan onderscheiden van gewone afbeeldingen. In dit verslag zal duidelijke worden wat cliparts zijn en hoe de identificatie gebeurd. Voor het schrijven van het programma werd gebruik gemaakt van Visual Studio 2017 en C# met als achtergronddoel een zo gebruiksvriendelijke interface te ontwerpen zodat men gemakkelijk kan experimenteren.

1 Onderzoek

1.1 RGB

RGB staat voor Rood-Groen-Blauw en is een model van het additieve kleurmenging type [1]. Een kleur ontstaat door de 3 primaire kleuren met elkaar te mengen zoals in **Figuur 1-1.**
Elke Primaire kleur wordt meestal uitgedrukt in een getal van 8 bits. Zo heeft deze een bereik van 0 tot en met 255, dit is de intensiteit van de kleur. Het is tevens ook mogelijk om RGB voor te stellen met meer bits waaronder 12, 16 of meer. Dit wordt vaak gebruikt waar hogere kwaliteit nodig is.

Een samengemengde kleur wordt uitgedrukt in 3 waardes van 0 tot en met 255 of meer naargelang de gekozen kwaliteit.



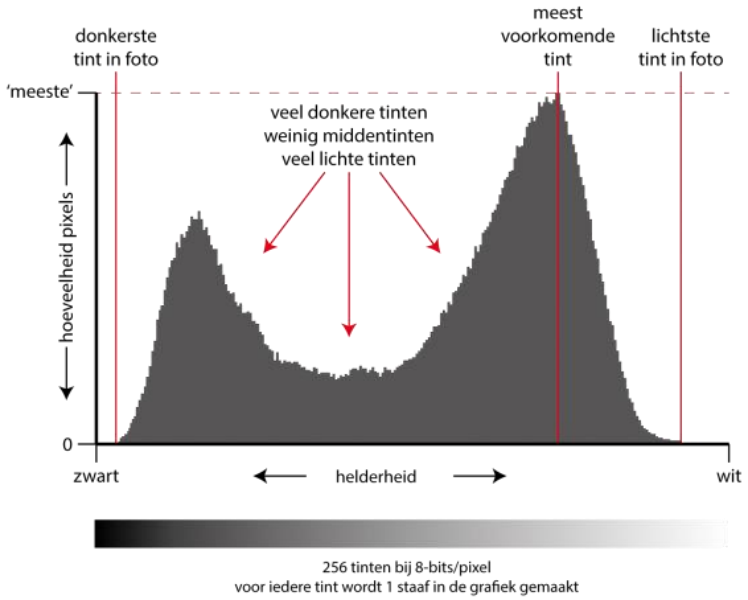
Figuur 1-1: RGB Model [1]

1.2 Histogram

1.2.1 Kleuren Histogram

Een Kleuren histogram is een grafiek waarbij voor elke helderheidswaarde het aantal keer dezelfde waarde weergegeven wordt. [2]

Figuur 1-3:
De x-as van het histogram bevat de helderheidswaarde.
De y-as bevat het aantal pixels. 1 staaf op de grafiek is het aantal pixels die dezelfde helderheidswaarde hebben.



Figuur 1-2: Histogram [3]

Voor een kleurenmodel kunnen er histogrammen gemaakt worden, één voor elk kleuren component. Het RGB-kleuren model zal totaal 3 verschillende histogrammen bevatten, een voor elke primaire kleur. Voor Rood, Groen en Blauw. Telkens van 0 tot en met 255.

Voor een histogram die alle kleur componenten bevat zal de afbeelding in principe eerst moeten geconverteerd worden naar greyscale. Hiervoor bestaan veel verschillenden methodes, in dit project wordt gebruik gemaakt van volgende formule: $0.3 * R + 0.59 * G + 0.11 * B$.

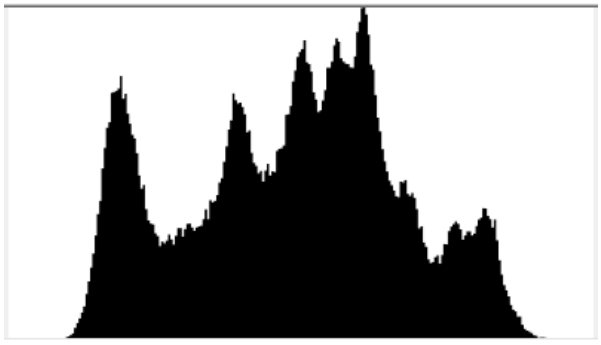
1.3 Clipart

Cliparts zijn voorgemaakte simpele afbeelding die iets voorstellen en vooral bestaan uit illustraties gemaakt door de hand of met software. Ze worden voorgesteld doormiddel van 2 soorten formaten waaronder bitmap en vector. De bitmap variant wordt gebruikt om afbeeldingen te maken die een grid vormen van pixels. Dit formaat is gelimiteerd in kwaliteit. Vector afbeeldingen daarentegen maken gebruik punten, lijnen, bochten en polygonen hierdoor kan een vector afbeelding vergroot worden zonder verlies van kwaliteit. [4]

1.3.1 Verschillen in het histogram

In volgende figuren 1-3 en 1-4 is te zien dat een histogram van een clipart afbeelding minder pieken heeft. Dit is te verklaren doordat een clipart meer gebruik maakt van eenzijdige kleuren en dus minder verschillende tinten van 1 kleur component gebruikt. Een clipart kan in principe wel meer tinten omvatten, dit zou de afbeelding complexer maken maar er zullen nog steeds grote pieken ontstaan door het overmatig gebruik van 1 soort kleur.

Figuur 1-3: histogram clipart afbeelding



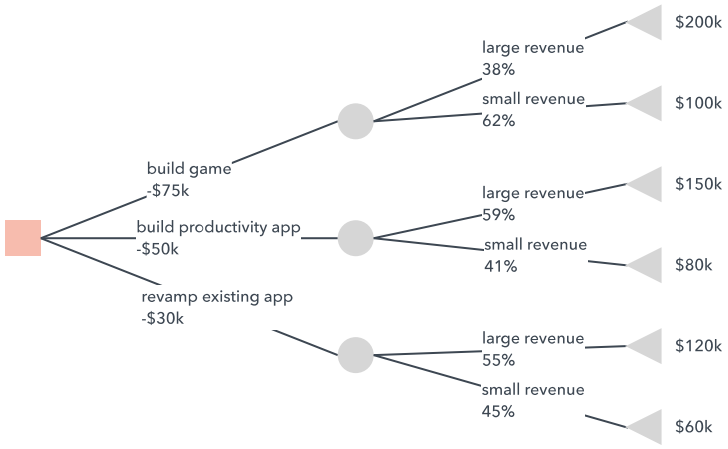
Figuur 1-4: histogram camera afbeelding

1.4 Weka en Machine Learning

Weka is software om machine learning toe te passen op grote data. Dit aan de hand van visualisatie tools en algoritmes. Er werd gebruik gemaakt van deze software voor het ontwerpen van een beslissingsboom. Met het gebruik van leer algoritmes, in dit project J48 en REP werden enkele bomen gegenereerd op data bestaande uit de histogrammen van clipart afbeeldingen en normale afbeeldingen.

Wat opvalt uit de Weka testen is dat hoe meer data met een groot onderling verschil van elkaar wordt meegegeven voor het genereren van een beslissingsboom, dat de tree groter wordt met meer voorwaarden op de attribueren. De tree wordt hierdoor ook nauwkeuriger. Er moet namelijk aan meer voorwaarden voldaan worden vooraleer een afbeelding kan worden geïdentificeerd.

Een beslissingsboom bevat bladen takken en knopen. De knopen die uittakken zijn in principe if-then statements en wanneer een attribuut van de ingegeven data aan een voorwaarde voldoet dan kan de data worden doorgevoerd door een specifieke tak, dit gebeurd tot een blad bereikt wordt. Een blad is een eindbestemming en geeft de uiteindelijke oplossing weer.



Figuur 1-5: Beslissingsboom [5]

2 Uitwerking

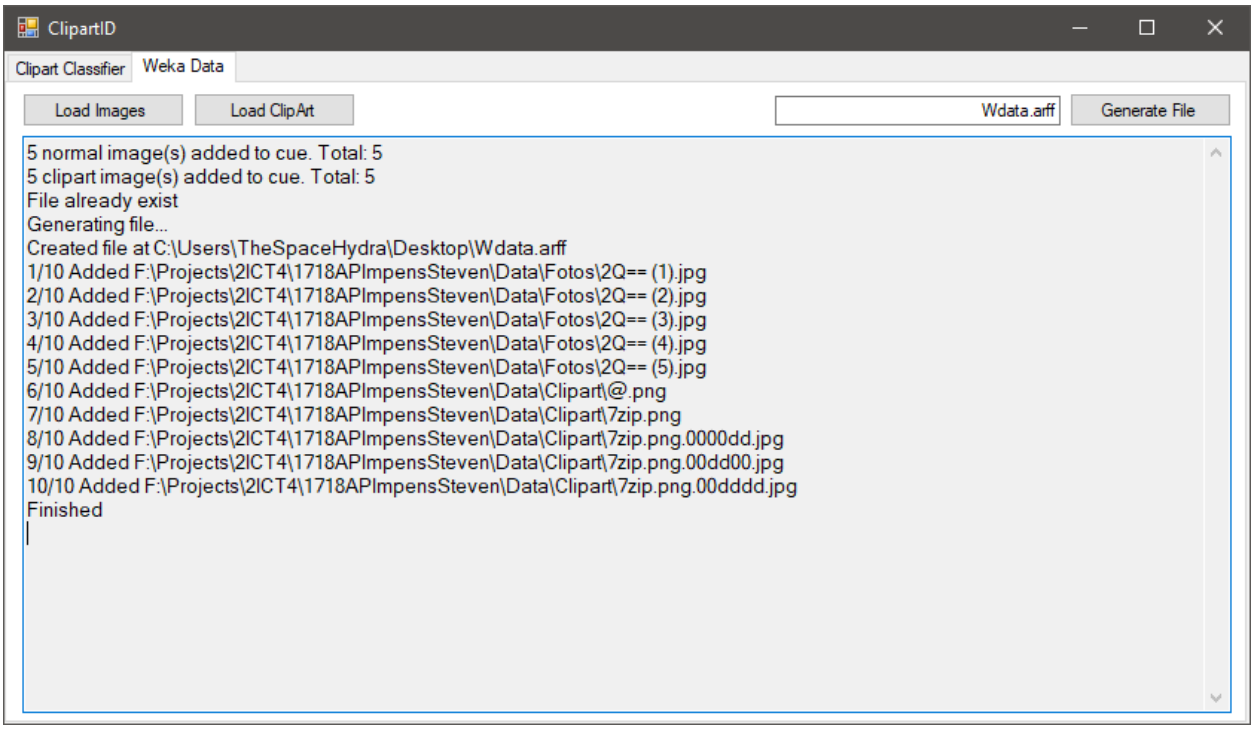
2.1 Grafische User Interface

De Grafische User Interface afgekort GUI bevat 2 tabbladen zoals te zien in **Figuur 2-1** en **Figuur 2-2**.

Tabblad 1 genaamd “Clipart Classifier” is het meest relevante in dit project. Dit tabblad is voorzien om een clipart te kunnen identificeren. Het linker gedeelte wordt gebruikt voor een afbeelding in te laden en weer te geven. Het rechter geeft weer of de afbeelding een clipart is aan de hand van de gebruikte beslissingsbomen die geïmplementeerd zijn. Als een beslissingsboom “True” weergeeft dan is het een clipart volgens diezelfde boom. Het is ook mogelijk dat niet elke boom hetzelfde antwoord heeft.

Tabblad 2 genaamd “Weka data” wordt gebruikt voor het creëren van data die kan gebruikt worden in het programma Weka om een beslissingsboom te genereren. Hier is het de bedoeling om meerdere clipart en normale afbeeldingen in te laden en zo een bestand voor Weka aan te maken.

Figuur 2-1: Grafische User Interface Tab 1



Figuur 2-2: Grafische User Interface Tab 2

2.2 Implementatie beslissingsboom

De beslissingsboom verkregen met Weka wordt omgezet naar C# en geïmplementeerd. De histogram data van de ingeladen afbeelding wordt in een array geplaatst genaamd “data”. Hierin staan de attributen waarop getest kan worden in de beslissingsboom.

Weka J48 beslissingsboom	C# beslissingsboom
J48 pruned tree ------------------ 42 <= 99 | 191 <= 183: clipart (167.0) | 191 > 183 | | 161 <= 205: clipart (7.0) | | 161 > 205: normal (2.0) 42 > 99 | 0 <= 4612: normal (95.0) | 0 > 4612: clipart (8.0/1.0) Number of Leaves : 5 Size of the tree : 9	*Codefragment 2-1: Beslissingsboom in C#*

```
private Boolean TreeLumSmallJ48()
 {
     if(this.data[42] <= 99)
     {
         if (this.data[191] <= 183)
         {
             return true;
         }
         else
         {
             if (this.data[161] <= 205) return true;
             else return false;
         }
     }
     else
     {
         if (this.data[0] <= 4612) return false;
         else return true;
     }
 }
```

2.3 Afbeelding naar histogram data

De code zal over elke pixel van de afbeelding gaan en het aantal keer de waarde van een kleur component voorkomt bijhouden. De componenten zijn gedefinieerd als “sleutels” in een variabel van het type dictionary.

```
private void CalculateHistogram()
{
      Rectangle rect = new Rectangle(0, 0, this.OriginalImage.Width,
      this.OriginalImage.Height);

      BitmapData bitmapData = this.OriginalImage.LockBits(rect,
      ImageLockMode.ReadOnly, this.OriginalImage.PixelFormat);

      IntPtr ptr = bitmapData.Scan0;
      int bytes = Math.Abs(bitmapData.Stride) * this.OriginalImage.Height;
      byte[] rgbValues = new byte[bytes];

      Marshal.Copy(ptr, rgbValues, 0, bytes);
      this.OriginalImage.UnlockBits(bitmapData);

      for (int i = 0; i < bitmapData.Width; i++)
      {
            for (int j = 0; j < bitmapData.Height; j++)
            {
                  byte b = (byte)(rgbValues[(j * bitmapData.Stride) + (i * 3)]);
                  byte g = (byte)(rgbValues[(j * bitmapData.Stride) + (i * 3) + 1]);
                  byte r = (byte)(rgbValues[(j * bitmapData.Stride) + (i * 3) + 2]);

                   Color color = Color.FromArgb(r, g, b);

                   base.AddRgbColor(color);
            }
       }
 }
```

Codefragment 2-2: Itereren over pixels van een afbeelding

```
private void AddRgbColor(Color color)
{
    this.ValueCollectionRGB["R"][color.R]++;
    this.ValueCollectionRGB["G"][color.G]++;
    this.ValueCollectionRGB["B"][color.B]++;
    this.ValueCollectionRGB["LUM"]
      [(int)(0.3 * color.R + 0.59 * color.G + 0.11 * color.B)]++;
}
```

Codefragment 2-3: Toevoegen kleurwaardes aan dictionary

Besluit

Het is mogelijk clipart te identificeren met beslissingsbomen die gegenereerd zijn aan de hand van machine learning. Hoe meer data meegegeven wordt aan het machine learning programma hoe nauwkeuriger en uitgebreider de beslissingsboom zal zijn. De data moet hiervoor uiteraard verschillend van elkaar zijn. Door telkens dezelfde soort clipart en of gewone afbeelding mee te geven zal dus enkel op soortgelijke afbeelding correct getest kunnen worden. Het is dus best om een grote en uitgebreide verzameling van cliparts en gewone afbeeldingen te gebruiken voor het genereren van een beslissingsboom.

Literatuurlijst

[1] "RGB color model," Wikipedia, 8 November 2017. [Online]. Available: https://en.wikipedia.org/wiki/RGB_color_model.

[2] "Hoe gebruik je een histogram?," EMDAY, [Online]. Available: http://emday.nl/2013/03/hoe-lees-je-een-histogram/.

[3] "Histogram," emday, [Online]. Available: http://emday.nl/files/2013/03/histogram_B_1368px-560x460.png.

[4] Wikipedia, "Clip Art," Wikipedia, 30 November 2017. [Online]. Available: https://en.wikipedia.org/wiki/Clip_art.

[5] "Decision tree," LucidChart. [Online].