PoC documentación

Jose Garita-Mario Naranjo-Manuel Calderón

20-08-2016

Algoritmo kMedias

Clase Punto.java

La clase tiene como aributos un x y un y, estos corresponden a su ubicación en el plano cartesiano; además de un id (no se usa).

Los metodos corresponden a Getters y Setters de sus variables, adjunto a esto hay dos metodos crear puntos al azar, ya sea uno o varios puntos y distancias de dos puntos.

Clase Cluster.java

El cluster corresponde a los conjuntos de puntos más cercanos según lo determina el algoritmo principal. Posee un arreglo de puntos, el centro y un identificador. Como parte de sus funciones son simples Getters y Setters con su constructor y toString().

Clase kMedias.Java

El algoritmo kMedías con la clase de su mismo nombre, tiene varias como: cantidad de cluster, cantidad de puntos, coordenada máxima y minima. Sus algoritmos son :

- Un inicializador: solo inicializa las varias y puntos que llama a la clase punto con random (al azar).
- Un impimir: recorre cada cluster y los imprime para ser apreciables.
- Un algorimo principal llamado calcular: con un verificador de finalizado continua en un ciclo:
  - Limpiar los clusters, verificando la limpieza.
  - Obtiene los centros de los clusters.
  - Asignar puntos: por cada punto busca la menor distancia y lo asigna a su cluster.

- Calcular centros: por cada cluster busca los puntos centrales sumandolos todos y promediandolos.
- Vuelve a obtener los centros y a la dictancia inicializada en 0 la suma con los centros ultimamente calculados
- El algoritmo termina cuando la distancia es igual a 0.

GUI Web - AngularJS

Explicación General

La GUI web consta de 3 partes:

- Restful: esta consta de 2 paquetes la de datos que se encarga de manejar las clases, y una de web service, el web service se encarga de manejar el funcionamiento de las clases y sus caminos en web, y la configuración.
- RestfulClient: este es un cliente en del restful, usa javascript y angular.
- Server: Servidor generado por el Tomvat v8.0.

Al ejecutar la GUI en la parte de localhost Restful/rest/product/findAll aparecera una lista de XML de productos. En localhost RestfulClient dará solo el resultado de exito en consola.

Restful.

Restful contiene 2 paquetes el de datos, que es donde se manejan las clases y objetos, para el ejemplo dado se uso solo un objeto ejemplo llamado Product, leste objeto contiene la clase Product, sus atributos y métodos. Y la ws esta consta de ApConfig y ProductRestful. En este lado del proyecto se agregan las librerias que manejan al Tomcat del servidor, se usaron las librerías del Jersey 1.9.

- ApConfiq.java esta clase reconfigura a ProductRestful, esta da el pasaje rest a la aplicaion web, para poder acceder a los caminos de ProductRestful. Agrega recursos para dar el camino en resumen.
- RestfulProduct: este solo tiene el comportamiento de los productos, pero a su vez le da una entrada camino o Path para ser accedido por medio de la web, para este ejemplo usa un arreglo de Products enlistado y los devuelve para su respectivo uso, además de imprimir en consola que se tuvo exito al ingresar.

RestfulClient

Esta carpeta no tiene ninguna programación en java, pero contiene un index.jsp, este index por el momento no esta completo, su función es ser la presentacion de la web, por ahora lo que contiene es la conexión por medio local de la web que contiene el tabajo realizado en el RestfulProduct, por ahora solo hace que se imprima en consola que obtuvo el acceso a esta. En este se linkeo el angular de eclipse y una libreria de angular para el manejo de ciertas funciones.

Indec.jsp consta por ahora de una funcio que sería el var myapp que utiliza un modulo de angular.

Server

.El server que se uso es el Tomcat v8.0, es autogenerado, contiene una serie de propiedades y politicas de uso además de XML. El Tomcat es descargado por eclipse y el control de este es descargado, ya que se pide direcorio para el uso, depues de eso se ligan los proyectos que intecractuarán.

Reconocimiento de la cancha

Se realizó la implementación parcial del algoritmo de segmentación de jugadores, y específicamente, el reconocimiento de la cancha de juego. Por lo tanto, mediante la implementación de la detección del campo de juego se demuestra la detección del espacio delimitando los <<blobs>> o espacio de los jugadores en el campo. Asimismo, existe el procesamiento del video, dividiéndolo en cuadros o <<frames>> que son procesados cada uno para realizar la detección del campo de juego a lo largo del video.

Descripción

La implementación parcial consiste en los siguientes pasos:

Se realiza la conversión de la imagen de entrada del espacio de color RGB a HSV. Seguidamente se toma a la capa H del canal de la imagen en HSV.

```
public Mat doBackgroundRemoval(Mat pFrame)
{
    List<Mat> hsvChannels = new ArrayList<>();
    Mat thresholdImage = new Mat();

    // a new HSV image matrix is created
    _hsvImage.create(pFrame.size(), CvType.CV_8U);

    // 1. Convert image from RGB to HSV
    Imgproc.cvtColor(pFrame, _hsvImage, Imgproc.COLOR_BGR2HSV);
    Core.split(_hsvImage, hsvChannels);

    // get the average hue value of the image
    double threshValue = this.getHistAverage(_hsvImage, hsvChannels.get(0));
```

Luego se obtiene un promedio de matiz (Hue) de la imagen para poder aplicar un umbral y separar el fondo (campo de juego) con el resto de elementos del cuadro del video.

El promedio de matiz se obtiene tomando cada pixel de la imagen sumarlo con el resto y obtener un promedio dividiendo por el tamaño del cuadro.

```
double average = 0.0;
Mat hist_hue = new Mat();

// 0 to 180: which is the range for Hue values
MatOfInt histSize = new MatOfInt(180);
List<Mat> hue = new ArrayList<>();
hue.add(pHueValues);

// Compute the histogram for the hue on the image
Imgproc.calcHist(hue, new MatOfInt(0), new Mat(), hist_hue, histSize, new MatOfFloat(0,179));


// Getting the average hue value from the image
// It gets the hue of each pixel in the image, add them and
// divide for the image size (height and width)
for (int h = 0; h < 180; h ++)
{
    average += (hist_hue.get(h, 0)[0] * h);
}

return average = average / pHsvImage.size().height / pHsvImage.size().width;
```

Luego se aplica un umbral sobre la imagen con el promedio de matiz del cuadro y se realiza un procesamiento morfológico del cuadro para rellenar huecos y suavizar contornos y espurias.

```
Imgproc.threshold(hsvChannels.get(0), thresholdImage, threshValue, 179.0, Imgproc.THRESH_BINARY_INV);

Imgproc.blur(thresholdImage, thresholdImage, new Size(5, 5));

// (2) dilate to fill gaps, erode to smooth edges
Imgproc.dilate(thresholdImage, thresholdImage, new Mat(), new Point(-1, -1), 1);
Imgproc.erode(thresholdImage, thresholdImage, new Mat(), new Point(-1, -1), 3);
```

Se obtiene un resultado como el de la siguiente imagen

Tiempos y dificultades:

Tiempo		Dificultad
GUI		
kMedias	3 días de investigación y 2.5 días en implementación con 2-3 horas por día.	Implementación del algoritmo. Connfigurar eclipse e instalarlo por ser una nueva herramienta
Reconocimiento de la cancha	5 dias con investigacion e implementacion	Falta de documentación de librerías.