Informe

01

CURSO 2016-2017

Tecnologías Multimedia

González Ramos. Juan Diego
Zapata Hernández. Diego
Guillén González. Sergio
Fernández Rubio. Antonio
Hagen. Daniel

GRADO EN INGENIERÍA INFORMÁTICA

Departamento de Informática
Universidad de Almería

Resolución del problema de los Heliostatos

Problema propuesto:

La idea es crear un programa en Python que sea capaz de detectar el reflejo generado por cada heliostato sobre el receptor, con el objetivo de identificar el reflejo del heliostato que no esté colocado en su posición. En el caso de que en un futuro se nos proporcione otro material de trabajo, en el cual se muestran varios haces lumínicos en movimiento, la idea es mejorar el programa para que sea capaz de identificar cada uno de ellos.

Planificación:
Esta es una planificación con el supuesto progreso deberíamos de tener.
En el apartado Diario se muestra lo conseguido semana a semana a groso modo.

Semana del:	**Planificación:**
3 octubre	Nos documentaremos sobre el lenguaje Python y la librería OpenCV
10 octubre	Nos decidiremos por el uso de una IDE
17 octubre	Comenzaremos a programar partes pequeñas de nuestro código.
24 octubre	Buscaremos ejemplos de segmentación que poder usar.
31 octubre	Buscaremos ejemplos de template matching que poder usar.
7 noviembre	Solventar problemas de ejecución.
14 noviembre	Pretendemos solucionar el problema de tiempo que tenemos a la hora de la ejecución del código. Presentaremos varias propuestas para solucionarlo
21 noviembre	Ya habremos resuelto el problema de tiempo. Comprobaremos el correcto funcionamiento para cuando hay más de 1 proyección en la pantalla receptora. Para ello necesitaremos el video completo.
28 noviembre	Estableceremos un umbral en el cual se permitirá un pequeño movimiento de la mancha. Haremos una pequeña interfaz ya sea mostrar por consola o que salga una imagen mostrando que heliostato se mueve para poder proceder a que se corrija su desviación.
5 diciembre	Haremos un descanso debido a que no hay clase.**

12 diciembre	Esta semana se pretende terminar el proyecto.
19 diciembre	Refinaremos el proyecto para una mejor ejecución y presentación.
26 diciembre	Haremos un descanso debido a que no hay clase.**
2 enero:	Haremos un descanso debido a que no hay clase.**
9 enero:	Esta semana pretendemos hacer una memoria que incluya el diario presente en este documento exponiendo los problemas presentados y su solución, con capturas del código y comentarios explicativos sobre su funcionamiento.
16 enero:	Entrega del proyecto finalizado.

****NOTA:** Los semanas que tengamos de descanso las usaremos para tranquilamente ir mirando cosas que poder hacer sin tener presión alguna por falta de tiempo.

Cómo resolver el problema:

Nuestra idea es hacer un programa en Phyton que sea capaz de dividir el video frame a frame para analizar, reconocer y guardar la forma de cada reflejo para así poder identificarlos.

Una vez guardada la proyección de cada heliostato procederemos a establecer un fondo que tenga todos los heliostatos juntos, para así cuando se mueva uno poder saber cual es para que el técnico encargado pueda proceder a mover el heliostato desviado.

Programas a usar:

1- Para la comunicación entre todos los miembros del grupo usaremos Whatsapp y la plataforma de Slack.
2- Para compartir archivos usaremos Google Drive y si lo vemos necesario GitHub.
3- Usaremos las IDE's de Sublime Text3 o Eclipse dependiendo cada uno cuál vea mejor.
4- Ejecutaremos el código con el ejecutable de Python, el de sublime y usaremos Jupyter para comprobar pequeñas partes del código.

Horario de trabajo:

En principio el horario de trabajo será todos los lunes de 9 de la mañana a 1 de la tarde, este horario es modificable puesto que todavía no sabemos cuánto tiempo nos llevará hacer todo el trabajo.

Diario

Semana del 3 de octubre.

Aquí comenzamos a pensar cómo íbamos a solucionar el problema, para ello decidimos usar Python y durante esa semana estuvimos documentándonos sobre cómo instalarlo y usarlo.

Aquí hay una pequeña explicación conceptual de cómo tendríamos que resolverlo (susceptible a cambios):
https://github.com/diegozahe/TecnologiasMultimedia

Semana del 10 de octubre.

Durante esta semana instalamos Python y decidimos utilizar Jupyter, así que lo instalamos también.

Tutorial para instalación:
https://drive.google.com/open?id=0B3EEGohsGQbfNUpsNEpHbjZ4Vms

Semana del 17 de octubre:

Comenzamos con hacer pequeños ejemplos de segmentación.

Aquí dejaré un enlace de nuestro código. Lo actualizaremos a medida que vayamos completando el programa:
https://github.com/diegozahe/light-beams-unmixing/blob/master/conjuntoCodigo.py

Semana del 24 de octubre:

En esta semana se une Daniel al grupo y conseguimos hacer un avance significativo en el proyecto, encontramos parte del código en internet. Lo adaptamos a nuestro proyecto consiguiendo obtener el contorno de la proyección del heliostato.

Semana del 31 de octubre:

Día 31:
Conseguimos encontrar un código de reconocimiento de formas en fotos, estuvimos mirándolo y reutilizamos parte del mismo para hacer nuestro programa.

Mientras hacíamos pruebas con el código, nos surgió un problema, y era que al reproducir el video con nuestro programa se veía cortado por la parte derecha. Despues de preguntar al profesor y buscar por internet conseguimos hacer que el video se vea completo.

Día 4:

1º Estructuramos el código y creamos dos métodos:
El primero sirve para encontrar la máxima similitud del trozo de mancha que detectamos con cada una de las manchas completas. De esta forma detectamos cual es.

```
#le paso como parametro la imagen grande, luego la imagen pequeña que tiene que buscar
#y por ultimo el array donde estan guardadas las proyecciones
def contener(array, template):
```

Este método comprueba que cuando queramos introducir una nueva mancha no sea una que ya esté guardada.

```
#no deberia tardar mucho en hacer la comprobacion
def comprobarMancha(array, mancha):
```

2º Intentamos resolver un problema de tiempo, ya que al intentar establecer un nuevo "fondo" con el cual comparar los nuevos frame para saber si ha habido movimiento, esto nos cuesta una gran cantidad de tiempo ya que hace una comprobación cada 3 frame. Esta comprobación consiste en mirar si el primer y tercer frame son iguales. Para esto aplicaremos el filtro de la Pirámide Gaussiana para reducir la imagen 2 veces y comparar.

```
#Comprobamos si la imagen a cambiado y si lo ha hecho lo establecemos como fondo
#hacemos 2 capas de la piramide para reducir el tamaño de forma inmensa
if i == 0:
    compr[i] = gris
    compr[i+1] = cv2.pyrDown(cv2.pyrDown(gris))
    i+=2
elif i == 2:
    compr[i] = cv2.pyrDown(cv2.pyrDown(gris))
    i+=1
elif i == 3:
    compr[i] = cv2.pyrDown(cv2.pyrDown(gris))
    i+=1
elif i == 4:
    if (compr[1] == compr[2]).all() or (compr[1] == compr[3]).all():
        fondo = compr[0]
    else:
        i = 0
else:
    i = 0
```

Semana del 7 de noviembre:

Hemos encontrado varios problemas.

1º No sabemos cómo mirar si una imagen está ya dentro ya que si comparamos una mancha con otra es posible que compare dos manchas de tamaño diferente.

Semana del 14 de noviembre:

Hemos encontrado un problema con una librería que usamos en el programa a la hora de calcular la entropía, causando que el intérprete de Python se cierre inesperadamente, estamos intentando encontrar una solución.

Semana del 21 de noviembre:

Seguimos buscando una solución al problema que causa el cierre inesperado de Python.

Semana del 28 de noviembre:

Grabamos un video para intentar hacer funcionar nuestro programa, ya que el que tenemos no nos sirve ya que no obtenemos los resultados esperados, este video tampoco nos vale porque al grabarlo las luces reflejan en la superficie en la que intentamos enfocarlas.

Semana del 12 de diciembre:

Creamos el método **comprobarPAP()**

Semana del 19 de diciembre:

Creamos el método **comprobrarPAP2**, igual que el anterior pero solo comprueba en la parte de la izquierda del contorno de las manchas guardadas.

Además creamos un método para aumentar el tamaño de la imagen.

Semana del 9 de enero:

Probamos un metodo llamado **matchTemplate()** pero no nos funciona como deseamos en un principio. Además, surgen problemas a la hora de comparar una imagen con el patrón cuando el propio patrón a buscar es incluso más grande que la propia imagen.

Este método termina siendo llamado desde uno propio que llamamos **lorenzo()**

Semana del 16 de enero:

Ejecución de más pruebas con el código, intentando dar una solución correcta al problema, porque la identificación de las proyecciones sigue sin ser correcta.

El mayor problema que hay es cómo preparar al código para que tenga en cuenta la evolución de las proyecciones con el tiempo, puesto que la proyección aún siendo de un mismo heliostato no mantiene el mismo tamaño durante todo el período del vídeo.

Semana del 23 de enero:

En vista a la inexactitud del método de resolución implementado, probamos la ejecución del algoritmo implementado sobre un nuevo vídeo de entrada ajustado a unos requisitos más estrictos que comprueben dos aspectos del código desarrollado:

1) La eficacia del algoritmo para identificar una figura geométrica simple
2) La eficacia del algoritmo para identificar una proyección cuando ésta no cambia a lo largo del video.

En estos casos, el algoritmo resuelve el problema satisfactoriamente.