**Apuntes y posibles preguntas**

1. Primero viene la captura de la imagen, después el mejoramiento, y luego la extracción de características. Entonces el impacto de la calidad de la huella no será solo en la extracción de características, sino también en el proceso anterior a este, que es el mejoramiento. Por lo tanto la calidad puede ser útil para:
   1. Decantar huellas de mala calidad del proceso de mejoramiento.
   2. Decantar huellas de mala calidad del proceso de extracción.
   3. Aplicar los algoritmos más complejos y costosos a las peores huellas.
   4. Según el tipo de huella, (Good, Normal, Wet and Smudge, Dry and Ligh Inked and Spoiled), definir que algoritmo aplicar. El algoritmo para las huellas Dry no sería el mismo que para las Wet.
   5. Darle una calidad a cada minucia extraida.
2. Hay que profundizar en la transformada de Fourier.

Aki no se q poner….q info provee la FFT a la calidad de una HD???

Pq coger un método basado en la FFT y no otro de medición global?

1. ¿Por qué usar un estado del arte de hace tanto tiempo (2007)?
2. ¿Por qué escogiste ese algoritmo y no otro de los estudiados?

Analiza dos enfoques de la imagen, de manera global y de manera específica en los bloques de la misma. En el análisis local emite un criterio basado en la combinación de varias caract locales (media, varianza, etc….)

Q representa la media, la varianza, la no homogeneidad, etc???

1. ¿Qué es la covarianza?
2. Profundizar en las características de RUP y XP.

Las caract del equipo de desarrollo eran adecuadas para una metod agil (equipo pekenno, con experiencia y con buena comunicacion). XP: se tiene el cliente cerca, se prioriza la implementación y disminuye la documentación, la cual no es necesaria en un componente de este tipo, al menos no más q la recogida en el doc de la tesis.

Xp es adecuada para proyectos con requisitos imprecisos y muy cambiantes.

1. ¿Por qué C# y no C o C++ que son más rápidos en el procesamiento de imágenes?
2. Cuidado con lo de los requisitos no funcionales.

Aki hay q ver lo de las prestaciones de la pc, q en el doc están muy rigurosas y al final no tiene pq ser asi. Ya está en el doc pero = debemos tenerlo claro.

1. Profundizar en los patrones de diseño. ¿Por qué no usamos GOF? Leer la documentación.

Pq son para escenarios muy específicos, generalmente cuando hay interacción con una interfaz de usuario. (Después te digo par de ejemplos de estos patrones).

1. Ver lo del patrón controlador, q aunque la definición de UML y Patrones esta rara, en el pdf q enviaste esta mas legible, y quizás, según esa última definición parezca q debiéramos usarlo.
2. **¿Por qué no usamos pruebas unitarias?** Durante la implementación nos concentramos más en la parte matemática de los algoritmos que implementabamos que en el bien funcionamiento del código como tal. Las pruebas unitarias tienen como uno de sus objetivos probar métodos, para cuando se pase a capas superiores de programación como las interfaces de usuario o las controladoras estar seguros que el código que dejamos atrás funciona bien. Pero nuestro caso el que el código funcionara bien no era garantía de que diera buenos resultados, porque influían factores más allá que la correcta escritura del código, como factores matemáticos y algorítmicos.