

Managing Aircraft Simulation Requirements with Content-Based Image Retrieval

Jimmy Secretan
The DiSTI Corporation

Copyright © 2009 SAE International

ABSTRACT

Requirements analysis for aircraft simulators is often driven by photographs and videos of the actual aircraft. An engineer may gather and organize hundreds or even thousands of source photos of various instruments and devices unique to the aircraft. Managing all of this source information and referencing it to generate software requirements can be challenging and time-consuming. This paper explores Content Based Image Retrieval (CBIR) techniques to automatically process and search those images to generate basic requirements and to facilitate reuse. An unsupervised clustering algorithm groups source images based on minimal user input. Images processed in this way can also be queried by image similarity, thereby allowing project managers to find common source material among projects. The effectiveness of these techniques is demonstrated on an example cockpit.

INTRODUCTION

Owing to the availability of inexpensive computers and advanced development software, the costs of creating computer-based aircraft trainers can be extremely low compared to physical implementations. Because of this lower cost and inherent flexibility, virtual cockpit trainers (figure 1) and virtual maintenance environments (figure 2), have begun to replace their physical counterparts. However, making photorealistic simulations can be a demanding task. Realistically implementing these virtual environments often requires extensive source imagery and video. Engineers may gather hundreds or thousands of source images, which typically must be sorted and examined by the engineer. Furthermore, the set of images may be used numerous times, in the cycle of modifications, upgrades, or new projects that can take

advantage of existing source. Therefore, it is advantageous for engineer to keep these source images organized and easily accessible. One way of doing so is through painstaking tagging and sorting. However, this can be costly and time-consuming. Instead, we seek ways that meaningful indexing of the images can be done automatically.



Figure 1. A virtual cockpit for the F-16, made with DiSTI's GL Studio tool. The instruments in the cockpit are composed of 2-D geometry, overlaid with photorealistic textures. The textures were created using source images of the actual instruments.

A growing body of techniques in Content-Based Image Retrieval (CBIR) research may enable engineers to manage and sort these source images automatically. In this paper, we evaluate one such process for automatically extracting requirements, based on an unsupervised clustering process.



Figure 2. A Virtual Maintenance Trainer for the FA-18. The models of the jet and the support equipment were developed with extensive source images from the actual aircraft.

CONTENT BASED IMAGE RETREIVAL

Content-Based Image Retrieval (CBIR) describes the field of research which aims to support automated searching, organization and management of image databases. CBIR systems can be designed to exploit various types of information, including visual and statistical features, as well as semantic association with other information (Vasconcelos, 2007). A common application of CBIR is the ability to query by image content (QBIC): that is, to submit a particular image or sketch to an image database and have the database return similar images.

In order to be searched, the images are typically converted with the aid of certain *image descriptors*. The image descriptors provide a summary of the image data, often in the form of a series of numerical values. While the full image can be converted to represent a numerical vector (each pixel representing a value in the vector), pixel counts in the millions typically preclude this representation from being helpful in any sort of search. Summarized image descriptors, on the other hand, can express more meaningful properties of the image into highly compressed forms (on the order of 10s or 100s of dimensions, instead of millions).

SYSTEM IMPLEMENTATION

The motivation in utilizing CBIR concepts for eliciting requirements in an aircraft simulation is that the easier it is to sort and summarize the source images, the easier it will be to generate specific requirements. The general process for generating requirements from source images is as follows:

1. Index source images to generate descriptors
2. Cluster images and determine representatives

3. Manually fix automated organization
4. Generate requirements skeleton and other needed structures

Indexing of the images in this paper was supported by a CBIR toolkit called LIRE (Lux, 2007). LIRE (Lucene Image REtrieval) builds an API over the open source Lucene search engine libraries. LIRE iterates through the collection of source photos, extracts numerical values for a specified set of MPEG-7 descriptors, and then compactly stores these images summaries in a search engine database.

To cluster the images, K-Medoid clustering was chosen, which is discussed in the next section. K-Medoids requires a distance metric to assess how close one object is to another. In our system, they are provided by MPEG-7 image descriptors. Finally, these descriptors are linearly combined for a more robust distance metric.

K-MEDIOID CLUSTERING

K-Medoid clustering is a type of data clustering which seeks to improve the performance of algorithms like K-means, by using cluster centers which are actual points. K-Medoid is most commonly realized by the Partitioning Around Medoids (PAM) algorithm (Kaufman and Rousseeuw, 1990). This paper utilizes the PAM algorithm for its clustering functions.

Select k initial points as medoids $M_1 \dots M_j \dots M_k$

For each point P_i

Group P_i with its closest medoid M_j

Select a random, non-medoid point P_R

Compute the cost S of swapping P_R with its associated medoid M_j

If new cost is lower then swap P_R and M_j

Repeat until there are no medoid changes or maximum number of iterations

Figure 3. A listing of the PAM K-Medoid algorithm.

The PAM algorithm (given in figure 3) is relatively simple. Initially k random points are selected as medoids. Then all other points are grouped with their closest medoid. A random non-medoid point is selected and the cost of swapping that point with its medoid is evaluated. This