CORE
Metadata, citation and similar papers at core.ac.uk
Provided by DCU Online Research Access Service

A Framework and User Interface for Automatic Region Based Segmentation Algorithms

Kevin McGuinness, Gordon Keenan, Tomasz Adamek, Noel O'Connor

***Abstract*— In this paper we describe a framework and tool developed for running and evaluating automatic region based segmentation algorithms. The tool was designed to allow simple integration of existing and future segmentation algorithms, both single image based algorithms and those that operate on video data. Our framework supports plug-in segmenters, media decoders, and region-map codecs. We provide several sophisticated implementations of these plug-ins, including a video decoder capable of frame accurate decoding of a large variety of video formats, an image decoder which also handles a comprehensive collection of formats, and a efficient implementation of a region-map codec. The tool includes both a graphical user interface to allow users to browse, visually inspect, and evaluate the algorithm output, and a batch processing interface for segmentation of large data collections.**

The application allows researchers to focus more on the development and evaluation of segmentation methods, relying on the framework for encoding/decoding input and output, and the front end for visualization.

***Index Terms*— Image Segmentation, Video Segmentation, Framework, User Interface, Integration, Evaluation.**

I. Introduction

Several different approaches to segmentation were developed and contributed by each of the partners in the K-Space[1] project. Each method has its own particular merits and limitations, often as a result of being designed with a different application domain in mind. Generally, each tool has its own unique interface, and can only accept one or two input formats. Output formats also tend to vary across tools. With such a rich set of tools, the task of selecting and integrating the best tool for a given experiment or domain is time consuming and non-trivial.

Automatic evaluation of segmentation algorithms is a very difficult task. The effectiveness of an algorithm in a domain (semantic reasoning applications, search tasks) is often not possible to evaluate automatically. Most automatic evaluation methods compare, in some way, a manual human segmentation with an automatic segmentation, and produce a measure of the match. This is not usually a adequate representation of the usefulness of a segmentation in an application context. A user, however, may be able to intuitively determine what algorithm would be best for a particular domain context by simply examining some segmentation results.

As one of out research activities is development, testing and evaluation of segmentation algorithms, we decided that a tool that would allow us to easily integrate currently available algorithms, and develop future ones would be invaluable.

II. Features and Functionality

The following is an overview of the main features of the platform.

Image and Video Formats: The framework provides an interface for seek-able, frame accurate video decoding. The built in video decoder supports many video formats, including MPEG-1, 2, 4, Motion-JPEG, Quicktime and WMF. We also provide an image decoder capable of decoding both individual images and sequences of key-frames transparently. It supports a large range of image formats, including JPEG, PNG, PNM, GIF and BMP.

Centre for Digital Video Processing, Dublin City University, Glasnevin, Dublin 9, Ireland.

[1]K-Space - Knowledge Space of Semantic inference for automaic automatic annotation and retrieval of multimedia content.

Region-Map Format: The framework encodes region-maps using an efficient, portable format based on a subset of PNG. This allows segmenting video sequences with minimal space overhead.

User Interface: The user interface provides a lot of functionallity, including automatic decoder selection, concurrent browsing of video frames and segmented images, selected-range segmentation, useful visualization methods, and a simple interface for selecting algorithms and their parameters.

Batch Processing Interface: The batch processing interface allows command line segmentation of large image/video collections. All the parameters that can be selected in the graphical user interface can be input into a parameter file. Files, ranges and increments can be selected for highly configurable segmentation.

III. Architectural Overview

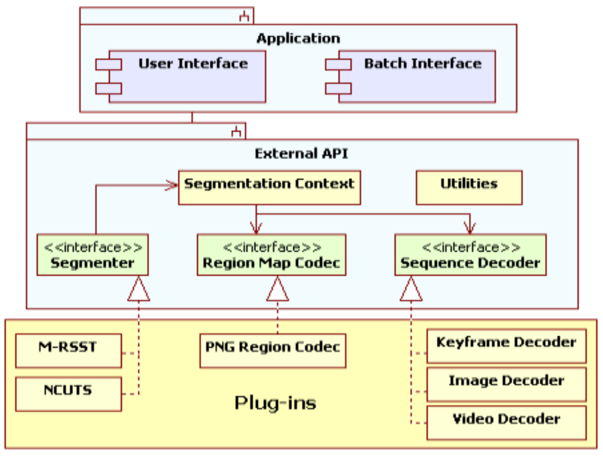


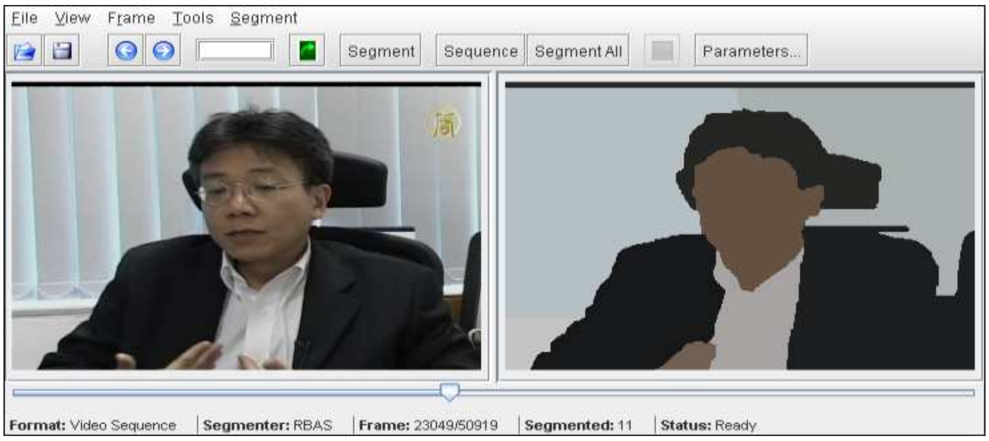
Fig. 2. High Level Overview of Software Architecture.

The framework is arranged into three main areas. The top-level module, the Application, hosts the user interface, user preferences, batch processing interface and integration logic. The application layer implements all of its encoding, decoding and segmentation via interfaces specified in the module below this, the External API. This API consists of a set of interfaces for plug-in developers, as well as commonly required utilities to simplify development. The bottom layer contains of all the plug-ins; built-in plug-ins and externally developed plug-ins are treated the same.

Application: The main components hosted by the Application module are the user interface and the batch interface.

The user interface provides a convenient and powerful way to perform segmentation operations, parameter selection, frame browsing, region visualization and plug-in configuration. This interface provides two visualization modes for viewing region maps, contrast stretching and color averaging mode.

The batch interface is designed for off-line processing of larger data sets. It is completely configurable from a parameter file, including decoder/segmenter/output selection and parameters, input files, ranges and increments. Output of batch operations can later be loaded and browsed in the user interface.



Format: Video Sequence | Segmenter: RBAS | Frame: 23049/50919 | Segmented: 11 | Status: Ready

Fig. 1. Screenshot of the Application User Interface

Segmentation: Developers wishing to integrate segmenters must implement the Segmenter interface. This includes all the functions required to configure parameters and perform the segmentation. When a segmenter is implemented and added to the platform, the algorithm name and parameter configuration will appear in the user interface.

The segmentation interface contains a segment responsible for performing segmentation on a single frame. For each frame, the segment method is passed a context object. This contains information that may be required to perform the operation, including the frame and index, a frame decoder, region map object, and an interface for acquiring previously segmented frames. This design allows each segmentation to be a single operation, while also providing enough contextual information for segmenters that require previous segmentations or frames. It simplifies the integration of single frame based segmenters, but provides enough information for segmenters that operate in the temporal domain. Of course, the internal implementation of a segmenter is entirely up to the developer, who may decide to buffer previous segmentations internally. In this case, no runtime overhead is incurred by the segmentation.

Image and Video Decoder: As the tool is frame based, a single interface is provided for both image and video decoders. This way the segmenter can handle single images (sequences of length 1), multiple images (e.g. key-frames) and videos in the same way. A powerful set of decoders are provided with the application, and the framework's plug-in mechanism ensures additional decoders can easily added.

The tool's integrated video decoder provides frame-accurate decoding of a multiple video formats. To achieve this, we decided to use the ffmpeg audio visual codec library [7] as a base for the video decoder. FFmpeg supports many video formats, so was ideal for our purposes. However, ffmpeg does not natively support frame-accurate video seeking. A frame-accurate decoder is required to ensure consistency across runs and for frame-accurate segmentation.

To attain fast, frame-accurate decoding from an arbitrary stream index, it was necessary to add a video packet parsing layer to determine (and sometimes interpolate) packet presentation timestamps, durations and other necessary information in advance of seeking in a stream. This and some additional functionallity is provided by the ffmpeg proxy layer. A standalone C++ and Java interface were built for this layer, and are fully re-usable.

One advantage of using ffmpeg as a base for the video decoder is that new codecs and improvements are constantly being added to it. As ffmpeg grows to support more formats, a simple recompilation of the ffmpeg proxy layer automatically adds this support to the tool.

The provided image and key-frame decoder plug-ins use the built-in Java image decoders as well as the JAI Image-IO library [6], which together support a comprehensive collection of image formats.

Region Storage: For the standard region map codec provided, we decided to utilize the open and widely accepted PNG format [1]. Specifically, the 8 and 16 bit gray-level PNG compression strategies.

For region maps of less than 256 regions, we employ the 8-bit gray-level encoding strategy, for more regions, the 16-bit gray-level format. The codec can thus support up to 65536 regions. Our experiments revealed that the compression rate of the codec was quite favorable. A typical segmentation of 10 seconds of MPEG-1 video (resolution 352x240, frame rate 29.97fps), required less than 500KB of storage.

Advantages of our chosen format are that it can be viewed in various imaging applications, simply by stretching the contrast between the regions. There are several software libraries for decoding PNG images freely available, like libpng [5], ImageMagick, and JAI ImageIO, making the format suitable for interchange.

IV. Integrated Algorithms

For our experiments, we integrated the Syntactic Modified RSST Algorithm [2], a fast Mean-Shift Algorithm [3], and a version of the Normalized Cuts [4] algorithm. Work is currently in progress to integrate more algorithms into the framework.

V. Future Work

Possible enhancements for the framework include; more visualization algorithms, MPEG-7 region description output, API for integrating automatic evaluation tools, and the ability to label regions for semantic reasoning applications. We would also like to use the framework components to develop a semi-automatic segmentation tool for ground truth generation.

VI. Acknowledgment

This material is based upon work supported by by the European Commission under contract FP6-027026, K-Space: Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content.

References

[1] *Portable Network Graphics (PNG): Functional specification*, ISO/IEC 15948:2004, March, 2004.
[2] N. OConnor, T. Adamek, S. Sav, N. Murphy, S. Marlow, *Qimera: a software platform for video object segmentation and tracking*, WIAMIS 2003, London, pp. 204-209, Apr., 2003.
[3] W. Bailerand, P. Schallauer, H. B. Haraldsson, H. Rehatschek, *Optimized mean shift algorithm for color segmentation in image sequences* Proceedings of the SPIE, Volume 5685, pp. 522-529 (2005).
[4] J. Shi, J. Malik, *Normalized Cuts and Image Segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 888-905, Aug., 2000.
[5] libpng, *PNG reference library*: `http://www.libpng.org/`.
[6] *JAI Image I/O*: `https://jai-imageio.dev.java.net/`.
[7] *FFmpeg Multimedia System*: `http://ffmpeg.mplayerhq.hu/`.