A Flexible System for Automated Composition of Intelligent Video Analysis*

Gayathri Nadarajan, Yun-Heh Chen-Burger and Robert B. Fisher
School of Informatics
University of Edinburgh, U.K.
Email: gaya.n@ed.ac.uk, jessicac@inf.ed.ac.uk, rbf@inf.ed.ac.uk

Concetto Spampinato
DIEEI, University of Catania
Catania, Italy
Email: cspampin@dieei.unict.it

***Abstract*—This paper outlines the automatic construction of video processing solutions using multiple software components as opposed to traditional monolithic approaches undertaken by image processing experts. A combined top-down and bottom-up methodology was adopted for the derivation of a suitable level of granularity for a subset of image processing components that implement video classification, object detection, counting and tracking tasks. 90% of these components are generic and could be applied to any video processing task, indicating a high level of reusability for a spectrum of video analyses. Domain-specific video analysis approaches (that exploit combinations of the above components) are built by using an automatic workflow composition module that relies on decomposition-based planning and ontologies. Evaluation on a set of ecological videos indicate that the proposed approach is faster and more flexible to adapt to changes in domain descriptions than specialized components written from scratch by image processing experts.**

I. Introduction

Despite being a relatively young field, computer vision has advanced rapidly over the past few decades especially in the branch of **video and image processing (VIP)**. Generally, VIP includes tasks such as recognition, motion analysis, scene reconstruction and image restoration. Recognition typically involves the identification of pre-specified objects of interest while motion analysis includes tasks such as target detection and tracking that involve following the movement of a set of points of interest or objects in the image sequence. These are often conducted computationally by image processing experts using highly specialized software. This work aims at providing a more flexible methodology for performing VIP tasks automatically so that specialized software does not need to be written from scratch each time. Moreover, the proposed system allows also users without image processing expertise to conduct complex VIP tasks. The remainder of the paper is as follows: first, related work in knowledge-based vision is discussed (Section II) followed by an explanation of a motion detection system (Section III). Section IV elaborates the methodology undertaken by this work to derive a set of VIP components based on video classification, object detection, counting and tracking. The VIP components are introduced in Section V, then their use within a workflow context is outlined. An experiment to evaluate the efficiency of the proposed approach on a set of ecological videos is described in Section VI, while Section VII concludes.

*This research was funded by European Commission FP7 grant 257024, in the Fish4Knowledge project (www.fish4knowledge.eu).

II. Background

During the last twenty years, several notable efforts have contributed to providing knowledge assisted systems and frameworks for automating the tasks of video and image analysis. Indeed, incorporating knowledge into video and image analysis approaches appears as a promising approach for improving efficiency. LLVE [1] is a goal-directed image segmentation system which uses image features and transfer processes as fundamental descriptive terms to represent the knowledge about image segmentation. It utilizes production rules to guide its search for optimal image processing solutions. CONNY [2] was built to investigate the basic concepts for a self configuring image analysis system aimed at facilitating high flexibility in handling different types of images for different analysis tasks and the direct transfer of human expert knowledge into the knowledge base of the system. OCAPI [3] attempted to overcome the rigidity of other expert systems by integrating image processing procedures at three levels; physical, syntactical and semantic. It also modeled the relationships between the various entities in the system, making it one of the pioneering systems that attempted at semantic integration that is achieved by ontologies today. COLLAGE/KHOROS [4] is a NASA-driven initiative that aimed at integrating an action-based planner (COLLAGE) to a visual-based library of image processing modules (KHOROS) to aid earth system scientists who study earth's ecosystems.

The described approaches are limited to a list of restricted and well known goals. Therefore *a priori* knowledge on the application context (domain-specific concepts such as sensor type, noise, lighting, target's information, *etc.*) and on the goal to achieve were implicitly encoded in the knowledge base. This implicit knowledge restricts the range of application domains for these systems and it is one of the main reasons that impede the reusing for wider VIP tasks and solutions of the developed software components.

Furthermore, most vision-based efforts concentrate on providing highly specialised techniques for very specific application domains due to the high demands for performance and accuracy, and, image processing experts, often, design and develop such applications from scratch each time, using trial-and-error cycles and often not reusing already developed solutions [5].

In short, knowledge-based vision approaches are still limited because they solve VIP problems using highly specialized hand-crafted solutions targeted at specific tasks, *e.g.* detection, classification, segmentation, *etc.* While this generally aims at higher accuracy, new solutions need to be rebuilt from scratch for new data or tasks. This work tries to overcome this limitation through a flexible approach based on decomposition-based planning and ontologies. In the next section an example motion detection system for fish detection and tracking in underwater videos is shown.

III. Fish Detection and Tracking

Typically image processing experts solve VIP tasks by designing and implementing, in form of single software components, ad-hoc solutions that exploit *a-priori* knowledge of the domain of interest. In this section, a software component, which performs video classification, object (fish) detection, counting and tracking tasks for undersea videos [6], is described in order to provide an understanding of the involved processes. In general, motion detection systems distinguish three levels of processing; pixel, frame and tracking [7]. To illustrate a concrete example, the pixel level processing, frame level and tracking level algorithms for fish detection, counting and tracking tasks are shown in Fig. 1. The shaded boxes indicate the specific algorithms that have been identified to perform subtasks at each processing level.

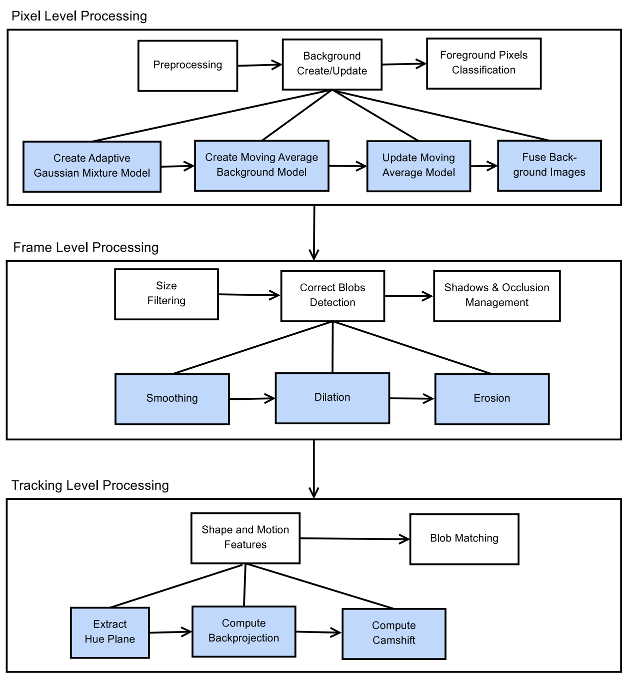

Fig. 1. Pixel, frame and tracking level processing algorithms for fish detection, counting and tracking task.

As can be seen from Fig. 1, the algorithms at the pixel level are first applied, followed by the ones at the frame level and finally the algorithms at the tracking level. The pixel processing level aims at identifying pixels belonging to objects (in this case, fish), hence at classifying pixels as foreground (objects) or background by a comparison with a background model, which is also created at this level. This motion detection system uses a fusion of two background models, Adaptive Gaussian Mixture and Moving Average models. The fusion of the two background models is achieved by finding the intersection between the two background images. Once a background model is created, the foreground objects in the current frame image are determined. This is achieved by removing occlusion and negligible (small) objects. The pixels identified can be visualised as a binary image (with black and white pixels) with the objects represented as white pixels and the background as black pixels. Fig. 2 illustrates an example.

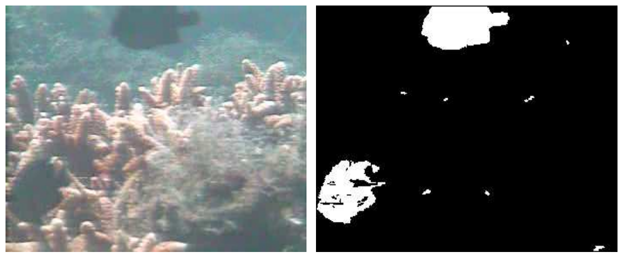

Fig. 2. Pixel level processing to identify foreground objects (right) from original frame image (left).

The frame processing level aims at analysing foreground pixels for grouping them into defined blobs, excluding the groups of pixels smaller than a certain size (size filtering). Moreover, the objects to be detected should be separated from their shadows and occlusion suppression should be done to separate blobs that represent more than one object. Shape filtering could also be applied to exclude objects of non interest. Basically this processing involves detecting the correct objects (blobs) among all the objects identified from the pixel level processing.

Following the example from Fig. 2, once the foreground objects are determined, they will need to be reanalyzed to identify the objects that are of interest for the detection task, *i.e.* fish. This is often done via a shape and/or size filtering mechanism. In this example, morphological operations that include a smoothing, followed by a dilation and then an erosion are applied to the binary image produced by the pixel level processing. A shape filtering is also applied where the shape of a fish object is determined via the computation of the area of the convex hull of a blob over the area of the blob itself. Fig. 3 illustrates an example frame level processing to detect fish. Finally the tracking processing level aims at achieving, after an appropriate extraction of the blob's features, blob matching to track the objects over time. This involves comparisons between the blob in question with all the blobs in a fixed number of preceding frames to find the blob that matches it best.

Taking the example from Figures 2 and 3, once the correct blobs have been identified, the processing is passed to the tracking level. Here, objects in consecutive frames are

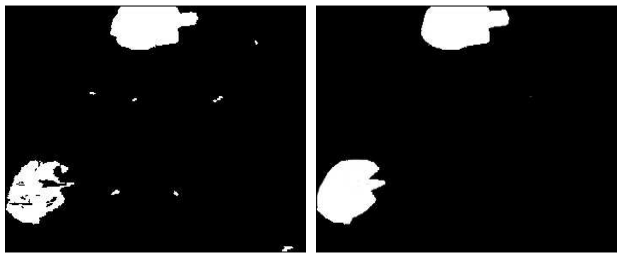

Fig. 3. Frame level processing to identify blobs (right) from binary image (left) with detected objects.

examined to identify which ones represent the same fish object (tracking). First, the backprojection image of the hue plane is computed. This image is used to predict what a blob will look like in the next frame using the Continuously Adaptive Mean Shift (Camshift) algorithm [8] by returning its centre, orientation and size. Using these three information, the Euclidean distance between the blobs in two frames are calculated to determine the closest matching pairs. This is repeated to compare blobs in a segment of ten consecutive frames. In this way, pairs of blobs that "match" in the segment refer to the same fish object. Fig. 4 shows the result of applying tracking level algorithms for fish counting and tracking. The next step foresees the recognition of fish species [9], but it is beyond the scope of the paper since we propose the knowledge-based method only for fish detection and tracking.

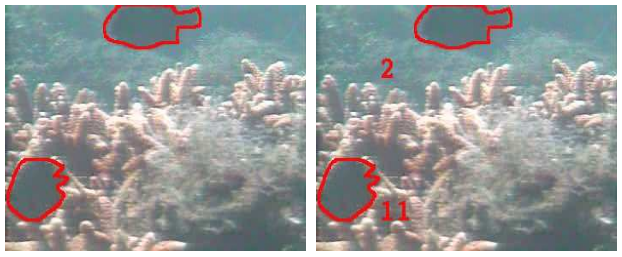

Fig. 4. Example tracking level processing for fish counting in a video. Sample results for fish detection (left) and fish counting (right). The top number indicates the number of fish in the current frame and the bottom number indicates the total number of fish in the video so far.

As described above, the fish detection and tracking system was implemented using VIP processes combined in a single monolithic software component. Using such processes, a combined top-down and bottom-up methodology was applied to derive a flexible system for video classification, fish detection, counting and tracking. In detail, this system selects the best combination of the IP processes (underlying the previous described system) to adapt to different domains and tasks and it is intended to provide the basis for a modular and reusable way to solve VIP tasks.

IV. Methodology

One of the most challenging aspects of conducting this research was identifying a suitable set of VIP tools that would represent a group of operators in a workflow composition and execution engine (Section V). Typically, an image processing task is solved by combining low level processes into a single component that works only on one task (domain) or a small subset of tasks. In order to construct image processing programs automatically, IP processes of a lower level of granularity would be required. In order to do this, a combined top-down and bottom-up approach was adopted, similar to the method advocated by Uschold and King for ontological building approach [10]. First, the aforementioned fish detection and tracking software component was inspected thoroughly and tasks were broken down in a top-down manner. This involved breaking down the steps used in solving the task into meaningful VIP processes (blocks or components). Subsequently, however, the bottom level processes were grouped by procedure to provide a coarser level of granularity that was more manageable. This methodology has been used effectively to accomplish the derivation of image processing components for this work. The approaches are outlined in more detail below.

A. Top-down: Function Calls as Primitive Tasks

Initially, a top-down approach was adopted whereby operators were represented by primitive processes in the whole image processing software component. The VIP task can be seen as the high level goal that is decomposable into several major subtasks that are in turn decomposable into further subtasks until primitive processes are encountered. OpenCV[1] was selected as the basis for the image processing code after surveying several computer vision libraries. In a typical OpenCV program, the primitive processes correspond to function calls, assignments, arithmetic and logical operations. This tedious process involved separating variable declarations, headers and function prototypes from the body of the program, and then breaking down the program body into blocks of major subtasks, taking into consideration conditional statements (*e.g.* `if-then`) and loops. Once the major subtasks were identified, they were further decomposed until the primitive level.

The hierarchical decomposition was done on a program of approximately 1000 lines of code performing a video classification, fish detection, counting and tracking task [6]. This method decomposed the one big task into its primitive level operators. Among the major features or modules that were determined included i) Pre-processing that includes video capture and frame image grabbing; ii) A main loop that processes each frame which involves fish detection, extraction and tracking procedures; and iii) A classification and output phase that computes the final results, and creates an output video containing these results. Fig. 5 shows some example operators derived from applying the top-down approach to the OpenCV program. These are contained at the bottom layer of the diagram and each can be achieved using a single OpenCV or C++ function call.

This exercise yielded 85 unique primitive processes that were encoded as operators in the capability ontology and process library of the workflow system designed to evaluate

[1]http://sourceforge.net/projects/opencvlibrary

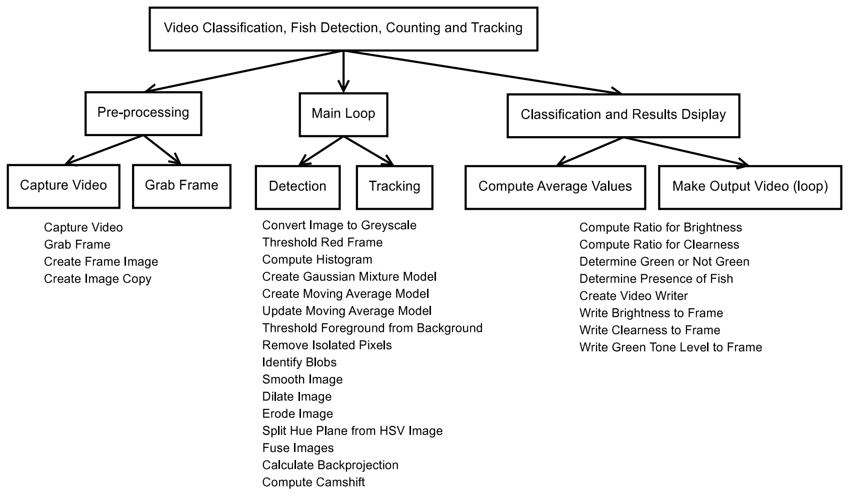
Video Classification, Fish Detection, Counting and Tracking
Pre-processing
Main Loop
Classification and Results Dsiplay
Capture Video
Grab Frame
Detection
Tracking
Compute Average Values
Make Output Video (loop)
Capture Video
Grab Frame
Create Frame Image
Create Image Copy
Convert Image to Greyscale
Threshold Red Frame
Compute Histogram
Create Gaussian Mixture Model
Create Moving Average Model
Update Moving Average Model
Threshold Foreground from Background
Remove Isolated Pixels
Identify Blobs
Smooth Image
Dilate Image
Erode Image
Split Hue Plane from HSV Image
Fuse Images
Calculate Backprojection
Compute Camshift
Compute Ratio for Brightness
Compute Ratio for Clearness
Determine Green or Not Green
Determine Presence of Fish
Create Video Writer
Write Brightness to Frame
Write Clearness to Frame
Write Green Tone Level to Frame

Fig. 5. Using top-down approach to identify some image processing operators for video classification, fish detection, counting and tracking task. Each text line underneath a set of boxes summarizes the primitive functions used to implement the tasks given in the boxes.

this work. When run on a one-minute clip containing 300 frames, 69,011 steps or operator invocations were produced. While the top level goals and their immediate subtasks (shown in white boxes in Fig. 5) provided an intuitive representation of the image processing tasks, the bottom level tasks or operators were too fine grained and did not provide a manageable level to work with. They were also too technical for an image processing-naive user to comprehend and make decisions upon (*e.g.* "Split Hue Plane from HSV Image"). Hence some of the low level tasks were merged to produce a coarser level of granularity, in order to provide a more manageable level for users (and the system) to work with.

B. Bottom-up: Grouping of Function Calls

Having all the primitive level tasks at hand, they were further packaged where possible to obtain operators with a more suitable level of granularity. This involved grouping the bottom level processes (primitive tasks) by procedure. For the most part, the primitive tasks were grouped to represent the subtask one level immediately above them (see Fig. 6). This exercise yielded 30 operators, termed as *independent components*, that were much more manageable to work with. They are introduced in Section V.

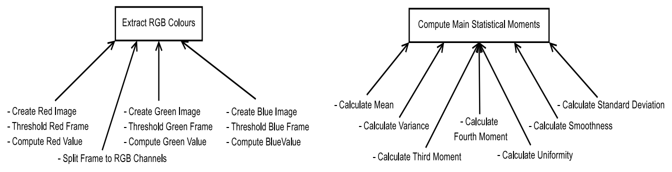

Fig. 6. Application of bottom-up refinement to derive the software components "Extract RGB Colours" and "Compute Main Statistical Moments".

The advantage of this bottom-up refinement approach has led to the identification of modules that could be reused for most VIP tasks. In addition, the components provided a more intuitive representation of the VIP tasks than their primitive level counterparts. For instance, in Fig. 6, the independent component "Compute Main Statistical Moments" which was derived by merging primitive tasks "Calculate Mean", "Calculate Variance", "Calculate Third Moment", "Calculate Fourth Moment", "Calculate Uniformity Calculate Smoothness" and "Calculate Standard Deviation", is a more compact and concise concept to represent a subtask to compute the mean, standard deviation and other statistical moments of an image.

Care was taken so as not to merge some tasks that need to be invoked independently into higher level subtasks. For example, in order to perform the classification of the video, three software components were developed independently as "Compute and Write Average Luminosity", "Compute and Write Presence of Fish" and "Compute and Write Presence of Algae" were developed independently. This then does not impose the classification task to include all of these criteria to be classified. "Compute and Write Presence of Fish", for instance is not required when performing only the video classification task, while it is required when performing video classification combined with fish detection, counting and tracking tasks. Hence the procedure involved thorough and repeated discussions with image processing experts in order to produce the operators with the most suitable level of granularity. A further refinement to reduce the number of steps included incorporation of loops within the operators where the number of iterations in the loop were known already or could be determined at run-time. With this reduction of almost threefold in the number of operators from 85 to 30, a sample run on the same one-minute clip of 300 frames tested on the operators from the top-down approach now yielded 8706 execution steps, a reduction of almost eightfold [11].

V. VIP Components & Use in Workflow Context

Each identified VIP component falls under one of six categories; "Pre-processing and Initialisation", "Compute Predominant Colours", "Compute Main Texture Features", "Perform Detection", "Perform Tracking" and "Perform Video Classification". 30 software components were developed for the given task, explanation of each component's function is given in [12]. These components were populated in a process library that was accessible to an automatic workflow composition and execution system, SWAV [13]. SWAV (whose architecture is shown in Fig. 7) utilizes decomposition-based planning and ontologies to compose VIP solutions using the aforementioned software components.

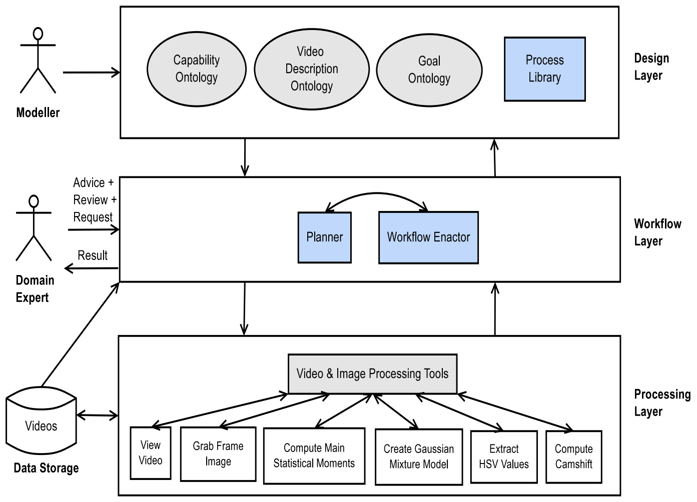

Fig. 7. Overview of workflow composition framework for video processing.

The high level user request is communicated by the user in the design layer. This is then fed to a planner in the workflow layer that translates the request to low level VIP invocations. The VIP components contained in the processing layer will be invoked by the workflow layer directly when needed. They are also represented formally in the process library and capability ontology in the design layer.

Consider the task "video classification according to brightness, clearness and algal (green tone) levels". The plan generated by the workflow layer is given in Fig. 8. The shaded boxes indicate the VIP components used for this task. It should be noted that while the overall process diagram for video classification task is the same, the workflow execution calls between different videos are not the same because different parameter values will be required for the components. As this is dealt with automatically by the workflow engine, modularity and reusability are achieved by the SWAV workflow system.

VI. Evaluation

An experiment was devised to show that the proposed approach adapts quicker to changes in user preferences than specialized VIP software components. This is the test of adaptability of the workflow system to reconstruct VIP solutions efficiently when the domain descriptions for a task are altered. This experiment will demonstrate that a solution constructed by an image processing expert using a specialized image processing software component takes longer to produce such changes in the domain descriptions for the same task. An image processing expert and a workflow modeller have access to the same set of VIP tools; the former has an OpenCV program with available image processing algorithms written as functions and the latter in the form of independent software components defined in the process library (VIP components described in Section V). 27 videos of varying quality from the Taiwanese Ecogrid project[2] were used as the data set.

Both subjects were familiar with the systems that they were manipulating. They were given an identical task to perform – fish detection, counting and tracking in a video. Both systems were able to perform this task using a default detecting and tracking algorithm. In the workflow tool, the Gaussian mixture model was defined as the detection algorithm, no methods were defined for the selection of any other detection algorithm. In the OpenCV program, the Gaussian mixture model was used as the detection algorithm. Six scenarios were presented to both subjects containing changes to domain conditions (see Table I). Both subjects were asked to make modifications or additions of code to their respective systems to cater for these changes in order to solve the VIP task as best as possible. For this purpose, they were both given which detection algorithm should be selected in each case. The number of lines of code (OpenCV for image processing expert and Prolog for workflow modeller) and the time taken to make these modifications were computed for both subjects. A line of code in OpenCV is represented by a valid C++ line of code, *i.e.* a line ending with a semi-colon (`;`), a looping or conditional statement (`if/for/while`). In Prolog, a line of code is a single predicate or fact ending with a full stop (`.`), a statement ending with a comma (`,`) or the head of a goal (line ending with `:-`).

The quality of the solutions was calculated as follows. There are two values to be considered, the first is the number of fish in the current frame and the second is the number of fish in the video so far. Each of these was given a score of 1 if there was a match with the ground truth. For each frame, the accuracy could be 0%, 50% or 100%. An average accuracy as a percentage is computed by taking the accuracy of 10 frames (1st, 6th, 11th, ..., 46th) from each video over all 27 videos.

Statistical hypothesis testing using the t-distribution [14] was conducted to measure the dependencies between the results obtained for the times (efficiency) taken to make changes to the workflow tool and OpenCV program. The hypothesis, H and null hypothesis, N for this experiment were:

H Constructing VIP solutions using the workflow tool is more **time-efficient** than modifying existing VIP programs each time a domain description is altered.

N There is no difference in the time taken to solve VIP tasks using the workflow tool and modifying existing programs each time a domain description is altered.

For this sample set, the two sample dependent t-test was performed to determine the t value and its corresponding p-value in order to accept or reject the null hypothesis. The

[2]http://ecogrid.nchc.org.tw

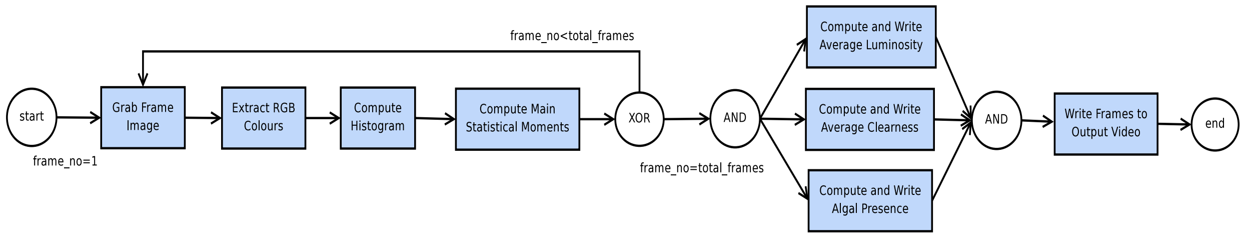
start
frame_no=1
Grab Frame Image
Extract RGB Colours
Compute Histogram
Compute Main Statistical Moments
frame_no<total_frames
XOR
frame_no=total_frames
AND
Compute and Write Average Luminosity
Compute and Write Average Clearness
Compute and Write Algal Presence
AND
Write Frames to Output Video
end

Fig. 8. Plan for "video classification according to brightness, clearness and algal levels".

TABLE I
COMPARISONS OF NUMBER OF NEW LINES OF CODE WRITTEN, PROCESSING TIMES AND ACCURACIES OF SOLUTIONS BETWEEN SINGLE-COMPONENT VIP PROGRAM AND MULTIPLE-COMPONENT WORKFLOW TOOL TO ADAPT TO CHANGING DOMAIN DESCRIPTIONS.

Domain Descriptions	Image Processing Expert			Workflow Modeller		
(User Preference)	New Lines of Code	Time (min.)	Accuracy %	New Lines of Code	Time (min.)	Accuracy %
Prefer false alarm than miss	43	16	58.25	3	3	59.30
Prefer miss than false alarm	56	23	62.55	2	2	64.80
Clear, no background movement	43	16	58.46	3	3	60.71
Clear, background movement	61	27	60.42	2	2	60.10
Blur, no background movement	43	16	60.88	3	3	62.09
Blur, background movement	57	32	63.80	2	2	61.22
Average	50.50	21.67	60.73	2.50	2.50	61.37

achieved significance level was of $p \ll 0.05$ and, assuming a significance level of $p < 0.05$, the null hypothesis was rejected. Thus the workflow tool is faster to adapt to changes in domain descriptions than the image processing program.

VII. CONCLUSIONS

In this paper we have proposed a flexible approach for intelligent video analysis based on a combination of top-down and bottom-up approaches to be used in a workflow context. A set of 30 low level VIP components have been identified useful for a typical set of VIP tasks that include video classification, object detection and object tracking tasks. 27 out of 30 of these VIP components have been identified as reusable with respect to generic video processing tasks by image processing experts, under the assumptions that the input values they depend on (text files, images and videos) are specified in the process library. The experimental results have shown that the proposed approach easily and reliably adapts to changes of tasks and domains. Moreover, this approach also enables the derivation of multiple combinations of VIP solutions for the same task, making it more flexible than previous approaches that can only derive a single sequential way for solving a VIP task. New VIP algorithms will be added and the system will be evaluated on different domains (*e.g.* video surveillance and human detection). This approach will also be tested on a distributed pipeline execution environment.

REFERENCES

[1] T. Matsuyama, "Expert Systems for Image Processing: Knowledge-based Composition of Image Analysis Processes," *Computer Vision, Graphics, and Image Processing*, vol. 48, no. 1, pp. 22–49, 1989.
[2] C. E. Liedtke and A. Blömers, "Architecture of the Knowledge Based Configuration System for Image Analysis "Conny"," in *International Conference on Pattern Recognition (ICPR'92)*, 1992, pp. 375–378.
[3] V. Clément and M. Thonnat, "A Knowledge-Based Approach to Integration of Image Procedures Processing," *CVGIP: Image Understanding*, vol. 57, no. 2, pp. 166–184, 1993.
[4] A. Lansky, M. Friedman, L. Getoor, S. Schmidler, and N. Short, "The Collage/Khoros Link: Planning for Image Processing Tasks," in *Integrated Planning Applications: Papers from the 1995 AAAI Spring Symposium*, 1995, pp. 67–76.
[5] A. Renouf, R. Clouard, and M. Revenu, "How to Formulate Image Processing Applications?" in *International Conference on Computer Vision Systems*, 2007, pp. 10–20.
[6] C. Spampinato, Y. H. Chen-Burger, G. Nadarajan, and R. B. Fisher, "Detecting, Tracking and Counting Fish in Low Quality Unconstrained Underwater Videos," in *3rd International Conference on Computer Vision Theory and Applications (VISAPP'08)*, 2008, pp. 514–519.
[7] A. Faro, D. Giordano, and C. Spampinato, "Evaluation of the Traffic Parameters in a Metropolitan Area by Fusing Visual Perceptions and CNN Processing of Webcam Images," *IEEE Transactions on Neural Networks*, vol. 19, no. 6, pp. 1108–1129, 2008.
[8] G. R. Bradski, "Computer Vision Face Tracking For Use in a Perceptual User Interface," *Intel Technology Journal*, vol. 2, no. 2, pp. 12–21, 1998.
[9] C. Spampinato, D. Giordano, R. Di Salvo, Y.-H. J. Chen-Burger, R. B. Fisher, and G. Nadarajan, "Automatic Fish Classification for Underwater Species Behavior Understanding," in *Proceedings of ARTEMIS '10*. New York, NY, USA: ACM, 2010, pp. 45–50. [Online]. Available: http://doi.acm.org/10.1145/1877868.1877881
[10] M. Uschold and M. King, "Towards a Methodology for Building Ontologies," in *IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.
[11] G. Nadarajan, Y. H. Chen-Burger, and R. B. Fisher, "A Knowledge-Based Planner for Processing Unconstrained Underwater Videos," in *IJCAI'09 Workshop on Learning Structural Knowledge From Observations (STRUCK'09)*, 2009.
[12] G. Nadarajan, "Semantics and Planning Based Workflow Composition and Execution for Video Processing," Ph.D. dissertation, University of Edinburgh, 2010.
[13] G. Nadarajan, Y. H. Chen-Burger, and R. B. Fisher, "SWAV: Semantics-based Workflows for Automatic Video Analysis," in *Special Session on Intelligent Workflow, Cloud Computing and Systems (KES-AMSTA'11)*, 2011.
[14] D. C. Howell, *Statistical Methods for Psychology*, 6th ed. Belmont, CA, 2007.