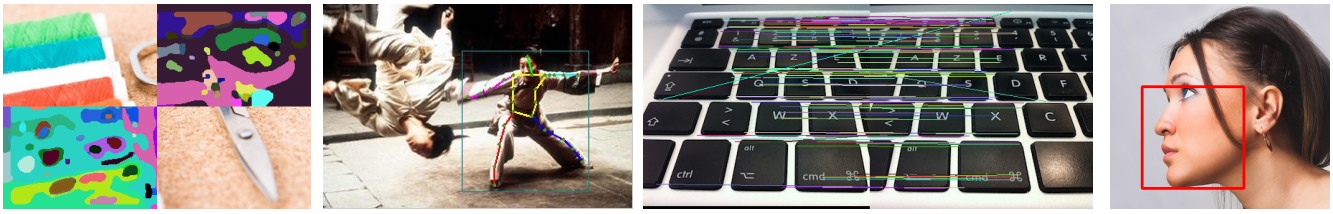
OpenVL: A Developer-Level Abstraction of Computer Vision

Gregor Miller and Sidney Fels*
Human Communication Technologies Laboratory
University of British Columbia, Vancouver, Canada



(a) Segmentation (b) Pose Detection (c) Point Correspondence (d) Face Detection

OpenVL provides an abstraction at the task-level over complex computer vision algorithms. The abstraction is based on segments*, distinct regions of the image defined by the developer through properties such as colour or texture (a) which are apparent in the image. Operations may then be performed on segments, such as pose detection (b), point correspondence (c) or face detection (d). For correspondence, a description of how segments vary between images is given; for detection, a template is defined for which groups of segments are matched.*

Computer vision is a complex field which can be challenging for those outside its research community to apply in the real world. Existing vision frameworks and libraries present APIs to users as lists of specific computer vision techniques (such as detection using Haar cascades, tracking with particle filters, etc.). Application of these methods under real-world conditions requires deep algorithmic knowledge in order to (1) select the appropriate algorithm for the current problem and (2) to provide appropriate parameters to this algorithm to achieve a reasonable result. Requiring this level of expertise is a significant barrier to the widespread adoption of computer vision. We propose a task-level abstraction called *OpenVL* which hides the details of algorithms behind a powerful interface flexible enough to provide solutions to a wide variety of vision problems. The target is to let general developers easily include vision methods in their applications without requiring a steep and long learning curve. OpenVL requires developers to have enough knowledge of a task to accurately describe it using our API. The developer's description is analyzed and used to invoke the appropriate algorithm(s), customize the parameters and provide a solution in the specified format. The OpenVL description model covers many tasks, such as segmentation, matting, correspondence, image registration, detection, optical flow and tracking. The description is made up of three parts: the first is a model of the contents of the image; the second is a description of the process (through a simple language model [Miller and Fels 2013]); the third is a detailed description of the individual components of the process.

The image contents are modelled as *segments*: regions within the image which are distinct from their surroundings. The definition of distinct is provided by the developer through *properties* such as colour, texture, intensity and blur. These can be chosen based on knowledge of the problem: e.g. to separate coloured balls or balloons we would choose *colour* as a the property, but to differentiate carpet and thread we may choose *texture*, as shown in Figure (a). If the chosen problem is segmentation, we use the model to segment the image based on the properties. Otherwise, we use this information as the first component of the problem description, and move on to the next step. Once segmentation has been defined for an image, we can apply operations using segments to solve other problems. For example, we can define a matching operation on segments which uses a developer-provided set of *variances* which describe how the segments vary between images (position, colour, intensity, size, etc.). If the matching (or image registration) problem is defined by the developer, the variances are used to select the correct algorithm: e.g. if the segments' intensity varies, an intensity-invariant method would be chosen by OpenVL. The operations may be sequenced together to describe higher-level tasks, e.g. image registration is a segmentation, correspondence and then global optimization to find a transform. The sequenced operations form the second component of the description, and the details (such as the variances) form the third and final component. When these are all in place, OpenVL interprets the description and executes a hidden method to produce the result.

The abstraction is designed to be easy to use, at a level above specific vision algorithms. This leads to more effective methods of acceleration: like OpenGL, vendors can provide their own implementations of OpenVL which compete based on power, precision and performance. For example, we can define the speed of operation in segments-per-second and detections-per-second, and use a standardized set of images and problems to evaluate quality. The quality-to-speed ratio would give some indication of the effectiveness of the implementation. Our reference implementation is CPU-based, and provides solutions for colour/texture/intensity/size segmentation, chroma-key matting, strong sparse correspondence, 2D image registration and front/profile face detection. A combined CPU/GPU version is also available for segmentation problems, and demonstrates the capacity of OpenVL to support hardware acceleration. We are continuously working on adding new descriptions and expanding the reference implementation for new problems. More information and development libraries are available from `http://www.openvl.org`.

*e-mail: {gregor,ssfels}@ece.ubc.ca

References

MILLER, G., AND FELS, S. 2013. OpenVL: A task-based abstraction for developer-friendly computer vision. In *Proceedings of the 13th IEEE Workshop on the Applications of Computer Vision (WACV)*, WVM'13, IEEE, 288–295.