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.

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 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.

References