Gesture recognition

Ondřej Novák, František Ondřej, Jan Ondřej

Requirements

Build a simple recognizer of yes/no gestures for Wintel platform. Emphasize the robustness of the recognizer, provide confidence information if possible.

Support turn on/off feature of the recognizer.

Build the system on top of Intel OpenCV toolkit

Integrate the recognizer using Visionary framework (IBM layer on top of OpenCV), encode the recognized gestures as XML messages

Gesture recognition support in OpenCV

There are function in OpenCV designed to generate motion template images that can be used to rapidly determine where a motion occurred, how it occurred, and in which direction it occurred which can by used for arms (legs) motions recognition e.g. arm-wavig. It also allows to define motion regions produced by movement of object of interest so it is not nececery to calculate the motion orientation for whole image.
There is also support for static gesture recognition in OpenCV which can locate hand position and define orientation (right or left) in image and create hand mask image. This can be potentially used to recognize various hand gestures, though we have no information how robust this recognition is. Because this method needs two cameras, we decided to base our recognizer on motion template functions.

Solution

Our gesture recognition method can be divided into several logical parts - motion tracing, motion comparing and gesture comparing.

Motion tracing
For motions tracing and preserving information about them we design class Motion. For every motion in image we save when motion begin and when ended. We also save motion center position and motion orientation in every frame. In this part we especially rely on OpenCv capabilities. In the concrete we use `cvUpdateMotionHistory, cvCalcMotionGradient, cvSegmentMotion` and `cvCalcGlobalOrientatio` functions. When a motion ended we compare it with motion templates to find if the motion is one of those we are interested in.

Motion comparing
Motion template (class MotionTemplate) is ideal motion and when we compared it with capture motion as a result we provide information how confident we are that they are same. First we compare time duration. Then overall motion direction. Both information have to be close to motion template values. If this test pass, we compare course of movement. To the effect we use motion orientation which we stored by motion tracing. All recognized motion are keep for further use. Templates are store in XML file and are loaded at program startup so it should be easy to add new motion templates.

Gesture comparing
As well as we use motion templates to compare with motions we have gesture templates to define gestures. Also gesture template are defined in XML file. One gesture may consists of one or more motions. Of course there are some restriction. Motions have to concur in defined time interval and consequential motion have to occurs near position where previous ended. Recognized gesture probability is set as average probability of all motions creating gesture. When gesture is recognized XML message is send to server.

Conclusion

We managed to build simple gesture recognizer based on OpenCv toolkit and integrated it into Visionary framework. As a yes gesture we mark up and down hand motions no matter which hand is used. As a no gesture we mark left to right and right to left movement or opposite right to left and then left to right. We also build in turn on/off feature and also time-limited recognition, which make possible to save computer resources. Due to storage motion and gesture templates in XML file it allows to change or add gestures without modifying source codes.

Appendix

Creating a new motion and gesture template

Motion template

To create new motion template it is necessary to add to the *gesture.xml* file a new *Motion* element. The *Motion* element is a descendant of the element *Motions*.

Motion has a *name* attribute, which describes the motion and is used in the gesture template. *minDuration* and *maxDuration* attributes specify the motion minimal and maximal duration in seconds, other motions are ignored. Attributes *startOverlap* and *endOverlap* determine the range of the motion in degrees, where the probability is not yet decreased. Moreover the Motion element has one *ShiftVector* and several *Gradient* descendants. *ShiftVector* defines the direction vector of the motion and has *x* and *y* attributes. The sequence of *Gradient* elements determine the gradient intervals, which are specified with *size* attribute.

The motion corresponded to the template must have at least one gradient in each interval, otherwise the probability is decreased. The *startOverlap* can be the same as the first gradient in the template or can vary.

Gesture template

Gesture template is defined with a *Gesture* element, which is a descendant of the element *Gestures*. The *name* attribute is the name of the gesture, it is one of the output parameter of the program and is sent to the server. The *timeDelta* and *posDelta* attributes determine the characteristics for connection of two motions in the gesture. Motions, of which the gesture consist, are specified with a sequence of *MotionName* elements. The *MotionName* has one attribute *name*, it must correspond to the name of the motion template defined in the *gesture.xml* file.

Configuration file

This section describes parameters in gesture.xml file. This file contains setting for motion recognition, motion templates and gesture templates.

Settings

Element *Settings* has several attributes for functions, which process the input image.

MhiDuration – maximal duration of motion track in seconds

maxTDelta – upper time threshold.

minTDelta – lower time threshold.

apertureSize – size of aperture used to calculate derivatives. Value should be odd (3, 5, 7).

segThresh – segmentation threshold; recommended to be equal to the interval between motion history "steps" or greater.

minMotionHeight – segments, which height in pixels is lower, are removed.

minMotionWidth – segments, which width in pixels is lower, are removed.

minWidthHeight – segments, which sum of width and height in pixels is lower, are removed.

minMotionLength – motions with less then *minMotionLength* gradients are removed.

correspondingMotionDistance – a maximal distance in pixels to assign a gradient to a motion.

Motion templates

Element *Motions* consists of one or more *Motion* elements. *Motion* element contains one *ShiftVector* element and a sequence of a *Gradient* element. *Gradient*

Motion

name – description of the motion.

minDuration – the minimal motion duration in seconds.

maxDuration – the maximal motion duration in seconds.

startOverlap – defines how much more can be the angle of the first gradient overlapped. (startOverlap = firstAnlge + overlap)

endOverlap – defines how much more can be the angle of the last gradient overlapped.

ShiftVector

x, y – the vector of the motion direction. (up – (0,1), left – (-1, 0), ...)

Gradient

size – defines the angle of the motion gradient from 0 to ~360 degrees.

Gesture templates

Element *Gestures* consists of one or more *Gesture* elements. Gesture element has several attributes and contains a sequence of motions defined with *MotionName* elements.

Gesture

Name – name of the gesture. This text is send to the server.

timeDelta – time in seconds. It is the maximal time between the termination of the motion and beginning of the next motion in the gesture.

posDelta – it is the maximal distance in pixels, in which must begin the next motion in gesture.

MotionName

Name– name of a motion template. Text in this parameter is compared with a motion template name, so it must have the same size of letters and the same length.