Visionsketch: Gesture-based Language for End-user Computer Vision Programming

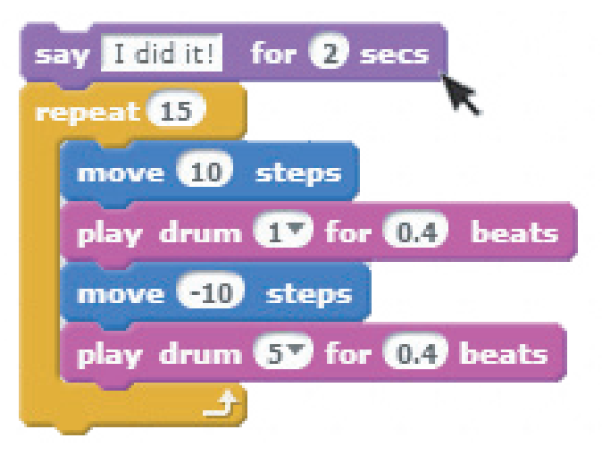
Jun Kato <i@junkato.jp>

The University of Tokyo

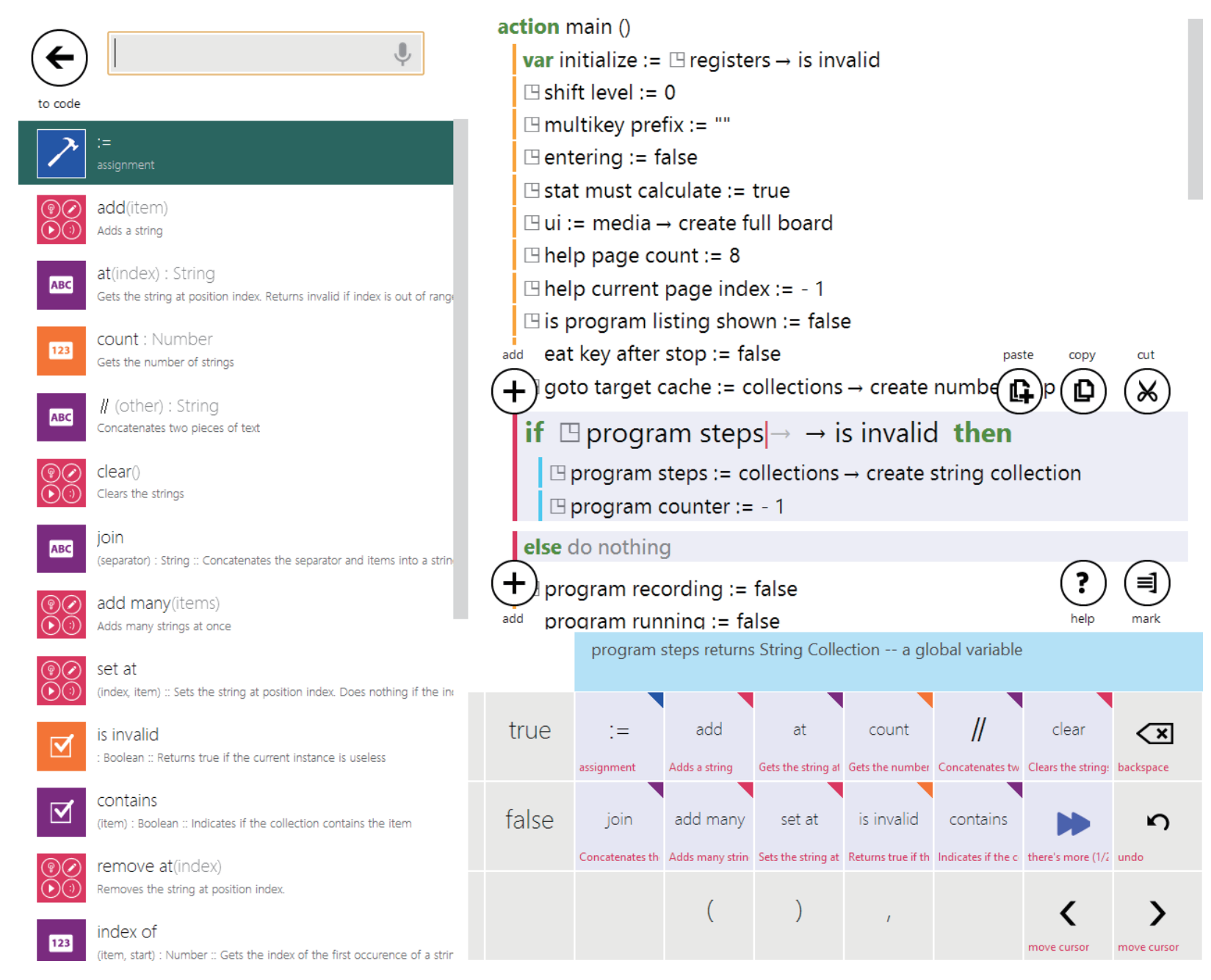
Problem and Motivation

Programming Language for Interactive Surfaces

An interactive touch surface is getting more and more popular as the primary input source for computers. Meanwhile, there is an increasing demand on the use of complex data types e.g. images. While textual or visual (symbolic) programming languages cannot handle such data nicely, I thought a new language on the surface can construct image processing programs through intuitive direct manipulation.



Scratch (Symbolic)



```
action main ()
  var initialize := registers → is invalid
  shift level := 0
  multikey prefix := ""
  entering := false
  stat must calculate := true
  ui := media → create full board
  help page count := 8
  help current page index := - 1
  is program listing shown := false
  eat key after stop := false
  goto target cache := collections → create numbe
  if program steps → is invalid then
    program steps := collections → create string collection
    program counter := - 1
  else do nothing
  program recording := false
  program running := false
```

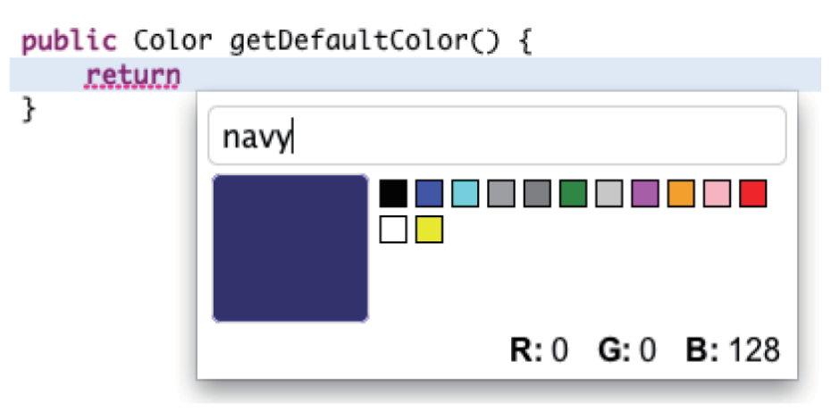
program steps returns String Collection -- a global variable

TouchDevelop (Textual)

Background and Related Work

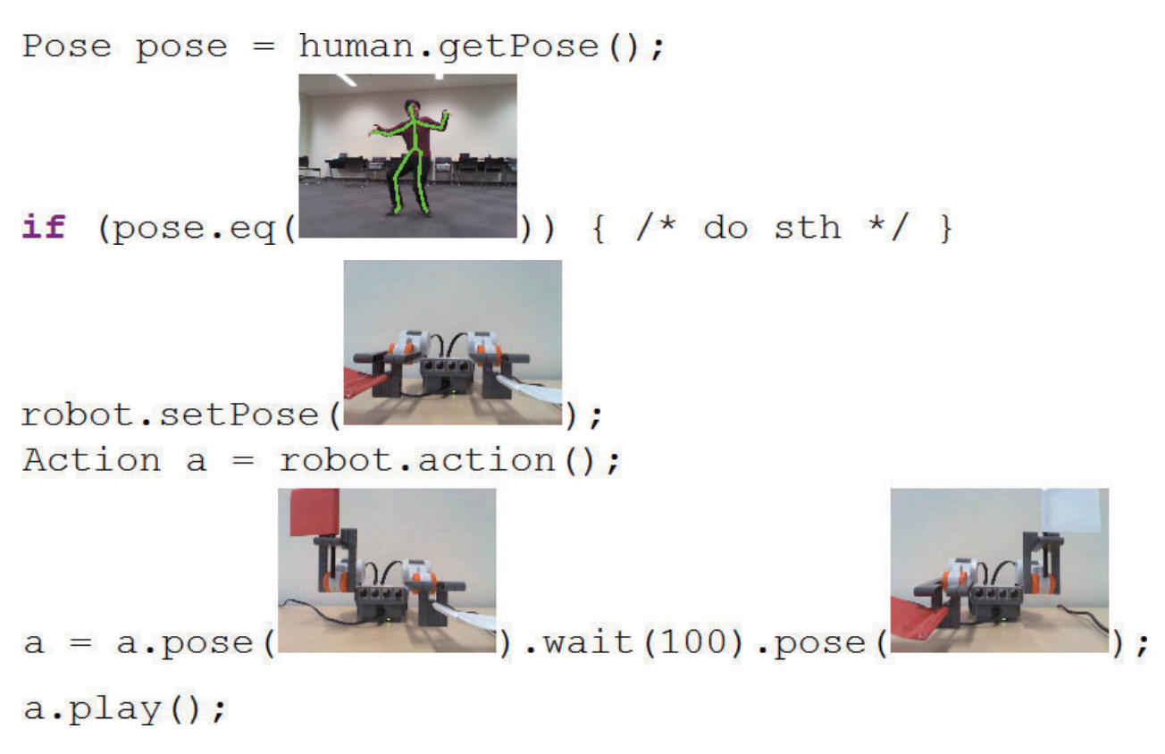
Programming with Visual Representations

Integrating visual representations into development environments has been successfully enhanced the programming experience. I made a new language and its environment from scratch rather than integration. They allow constructing domain-specific programs by direct manipulation just like Morphic (graphical user interface) and Excel (spreadsheet calculation).



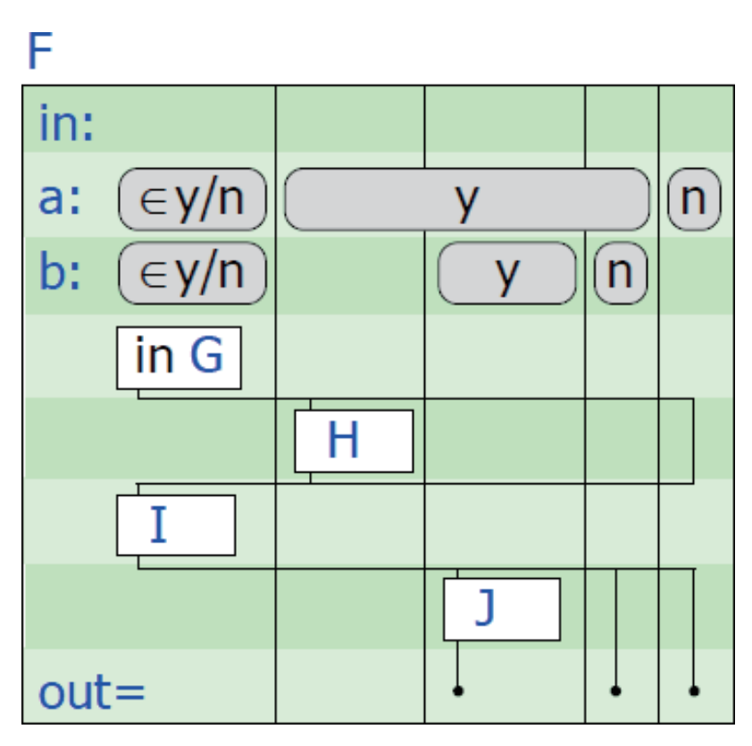
```
public Color getDefaultColor() {
    return
}
```

Active code completion helping complex data construction



```
Pose pose = human.getPose();
if (pose.eq( )) { /* do sth */ }
robot.setPose( );
Action a = robot.action();
a = a.pose( ).wait(100).pose( );
a.play();
```

Inline photos (Picode) representing posture data



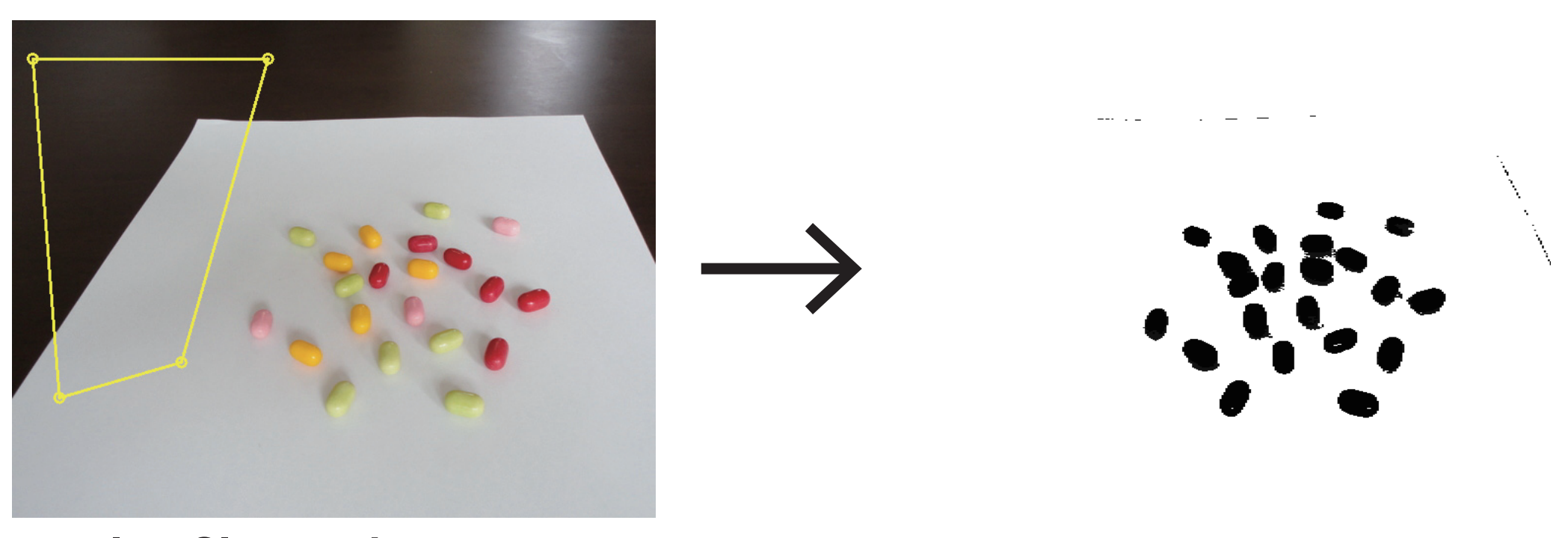
Schematic Tables (Subtext) representing logic

Approach and Uniqueness

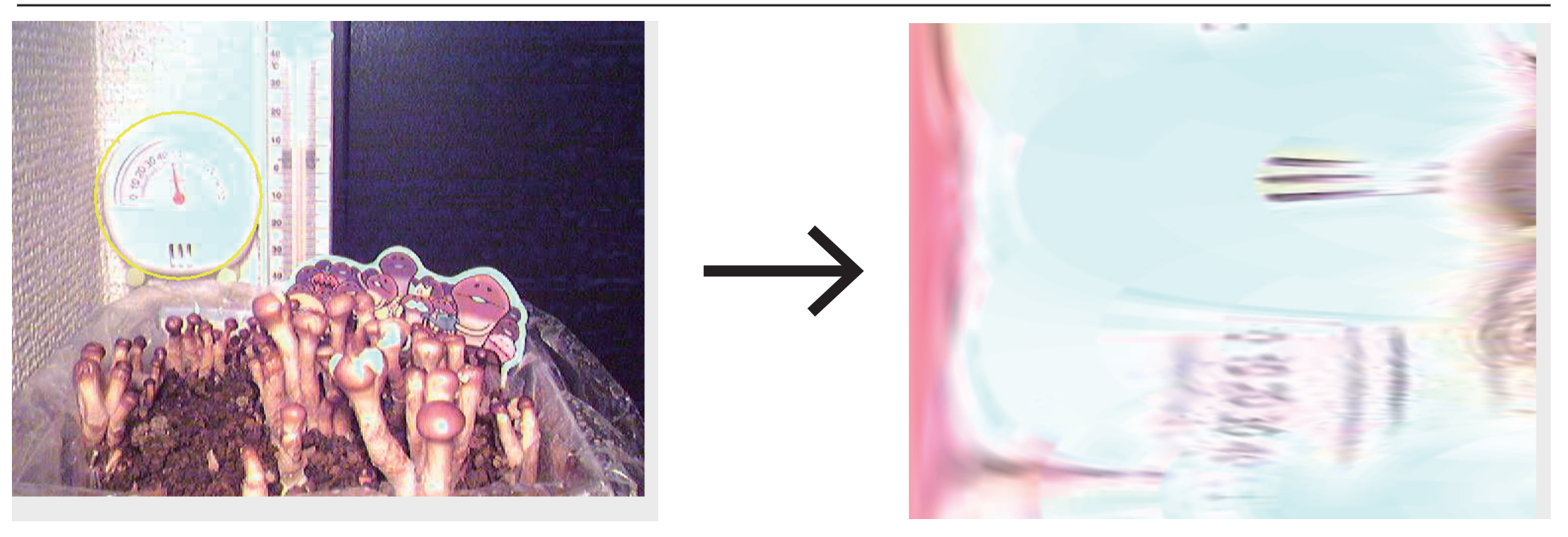
Gesture-based Language and its Integrated Development Environment

I propose Visionsketch, a gesture-based language where each code element is constructed through gestures on an image/video. With Visionsketch, the programmer can construct an image processing program by drawing lines and shapes instead of typing.

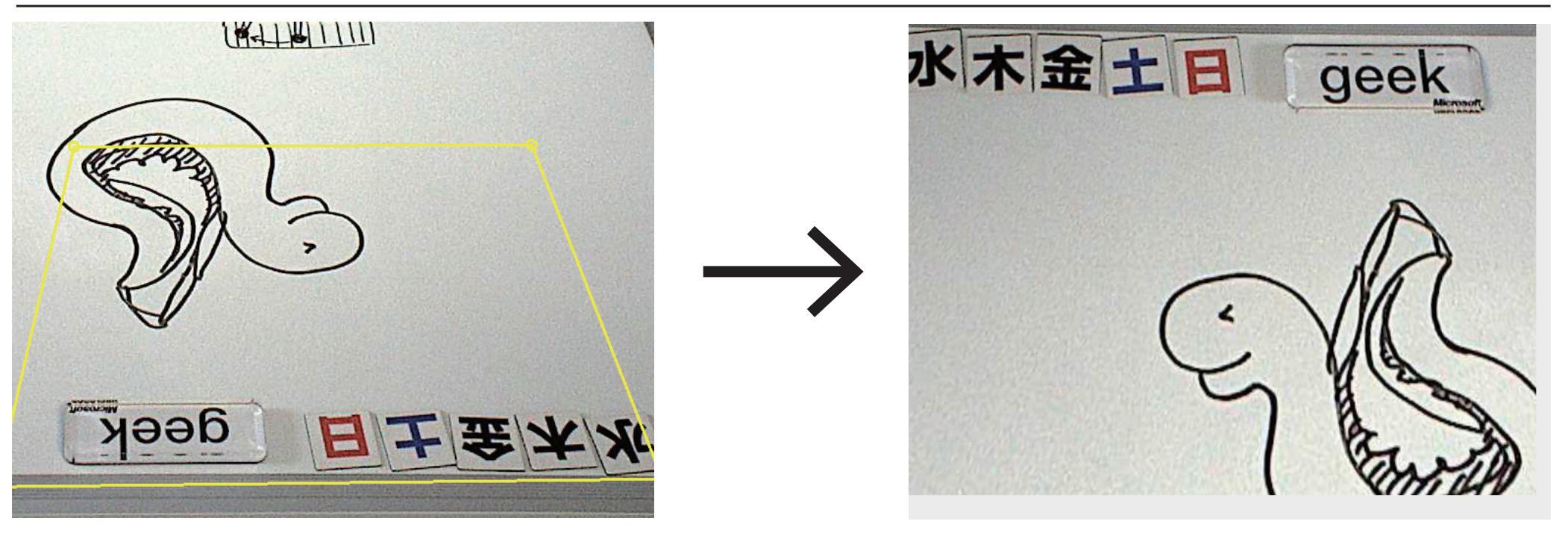
Language primitives



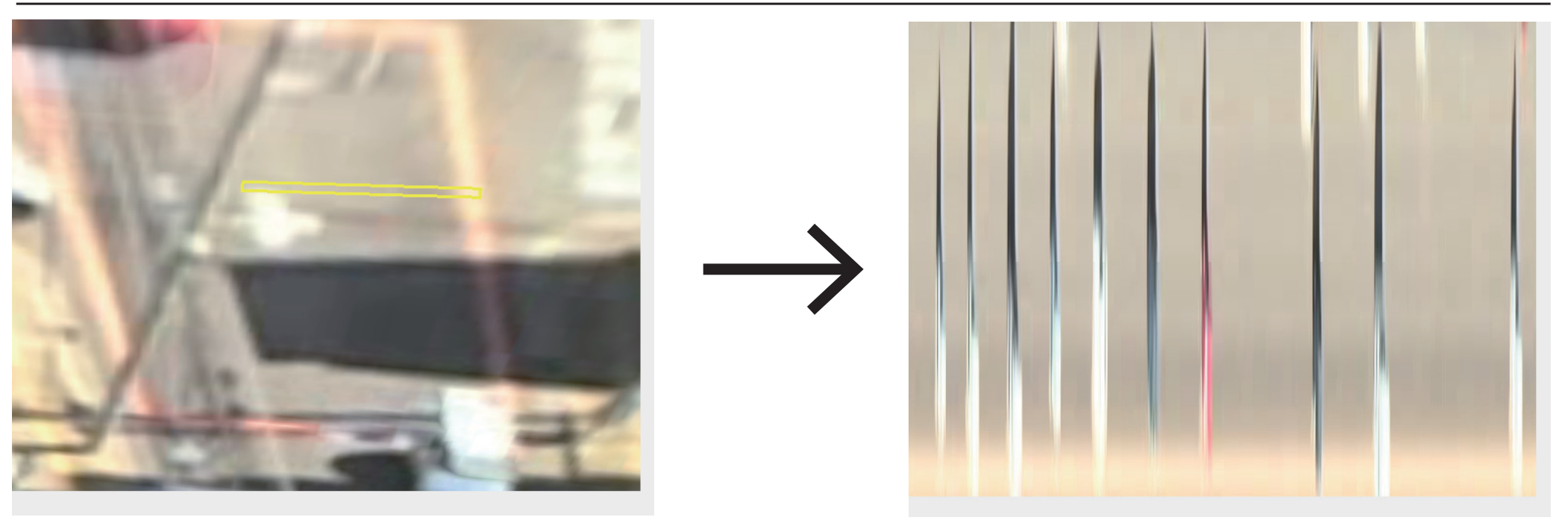
Color filter / shapes: any / out: image



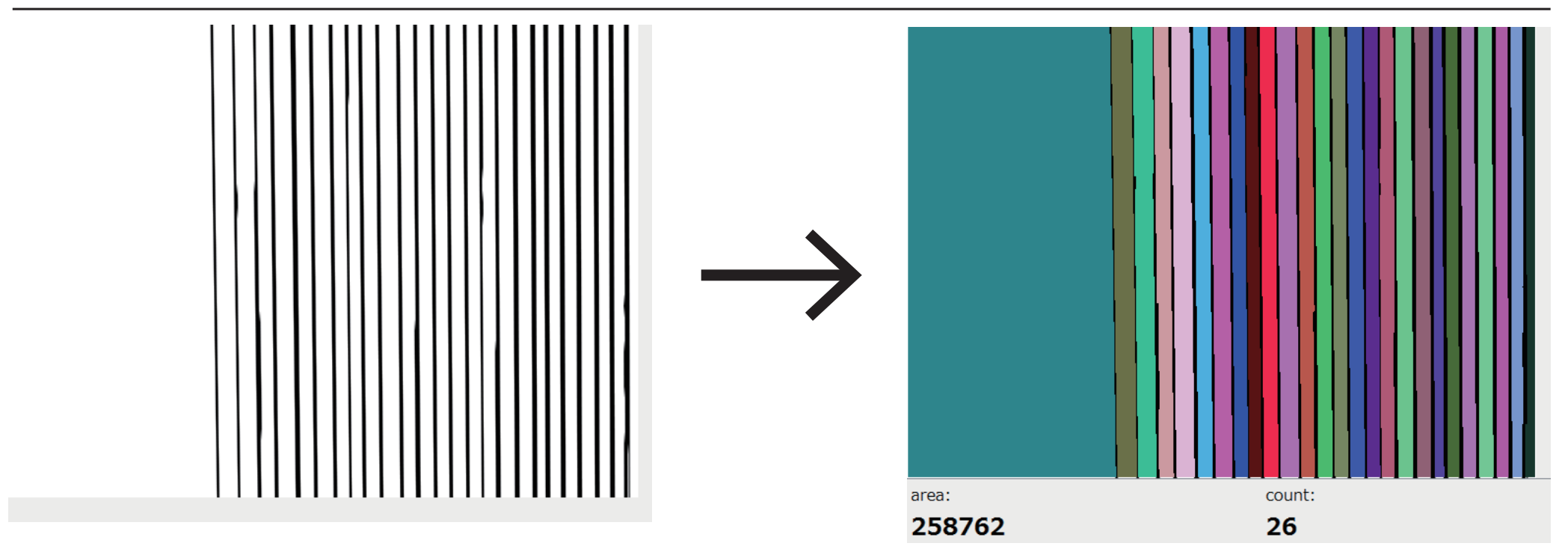
Linear-polar conversion / shapes: a circle / out: image



Perspective warp / shapes: a rectangle / out: image



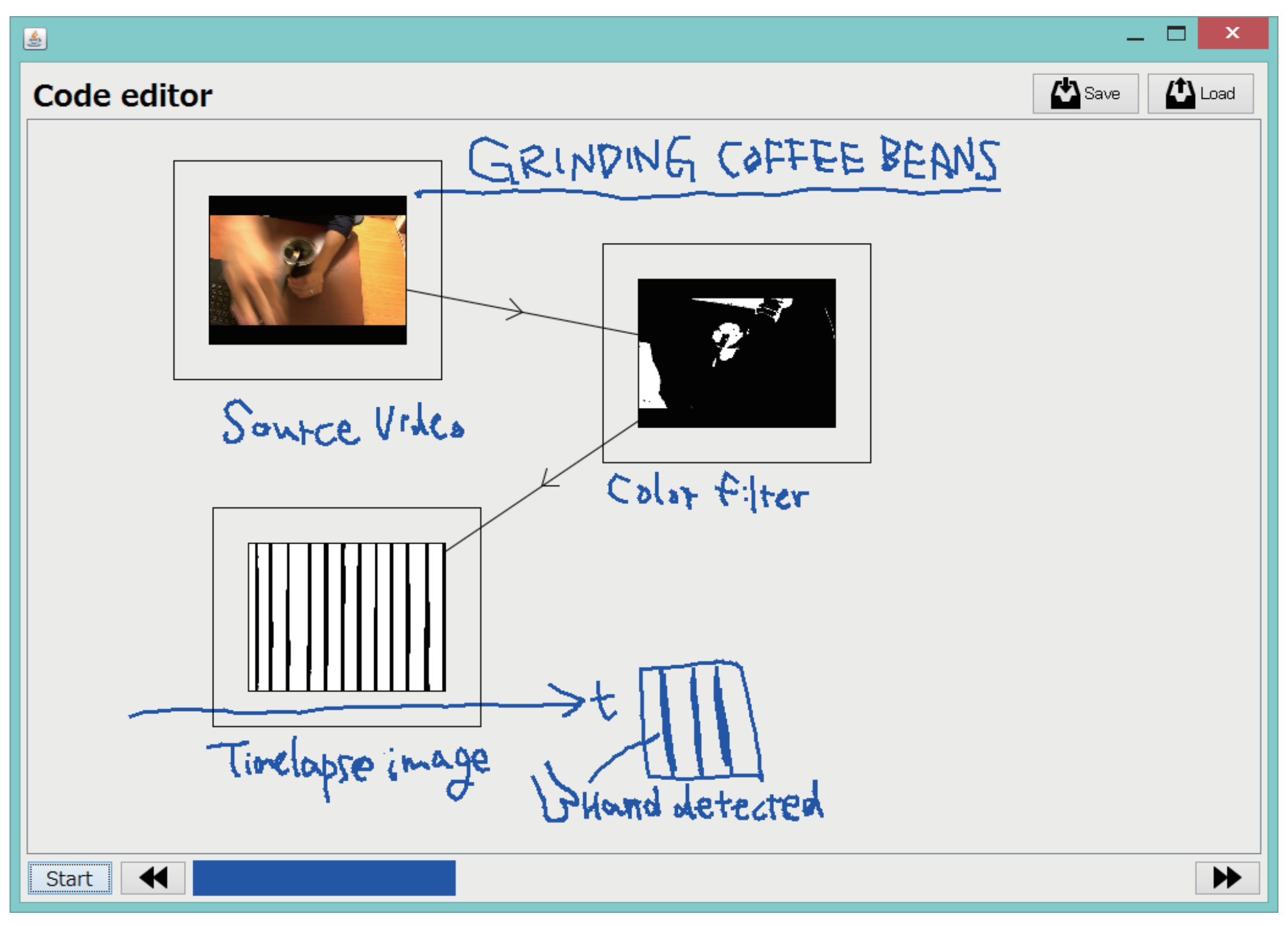
Timelapse conversion / shapes: a rectangle / out: image



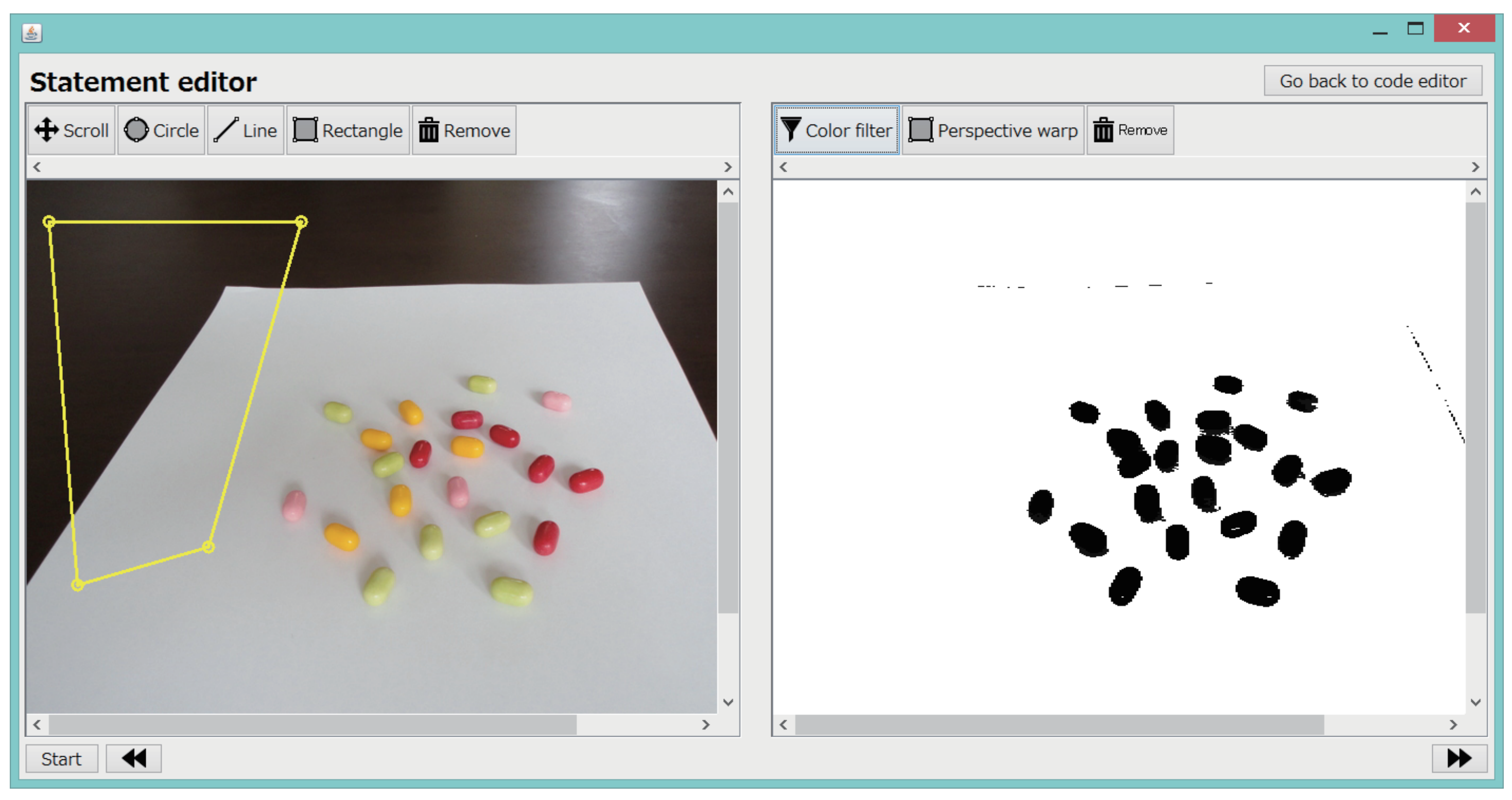
Contour counting / shapes: none / out: image, region count

(Other primitives can be easily implemented as Java classes.)

Code editor

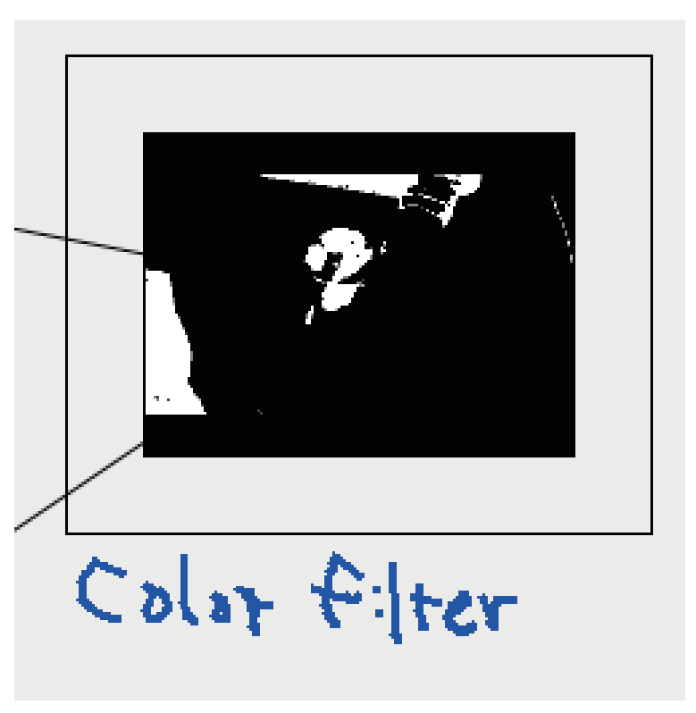


Statement editor



While Visionsketch IDE is constructed from scratch, it borrows some important concepts from modern text-based IDEs.

Comments in code

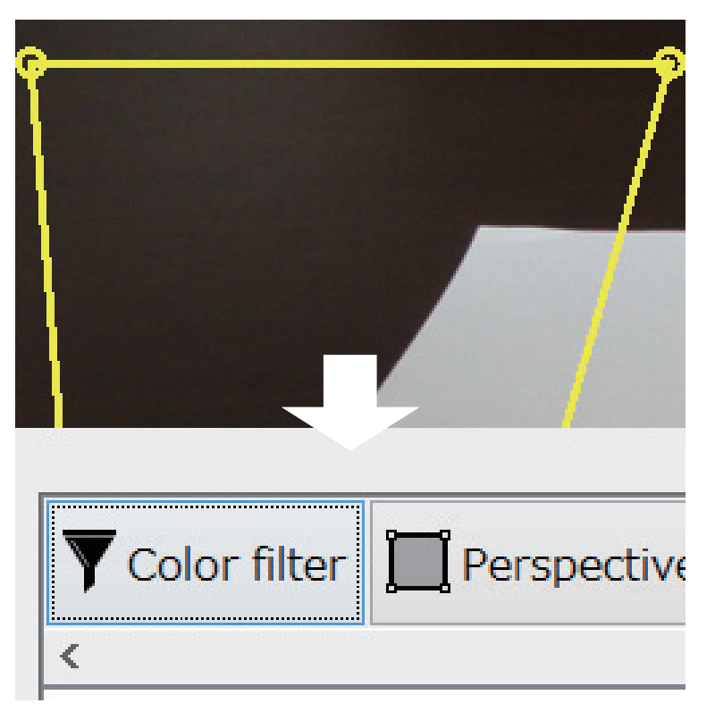


The code can be annotated by free-hand drawings, which are often more expressive than mere text comments.

Traditional text comment:

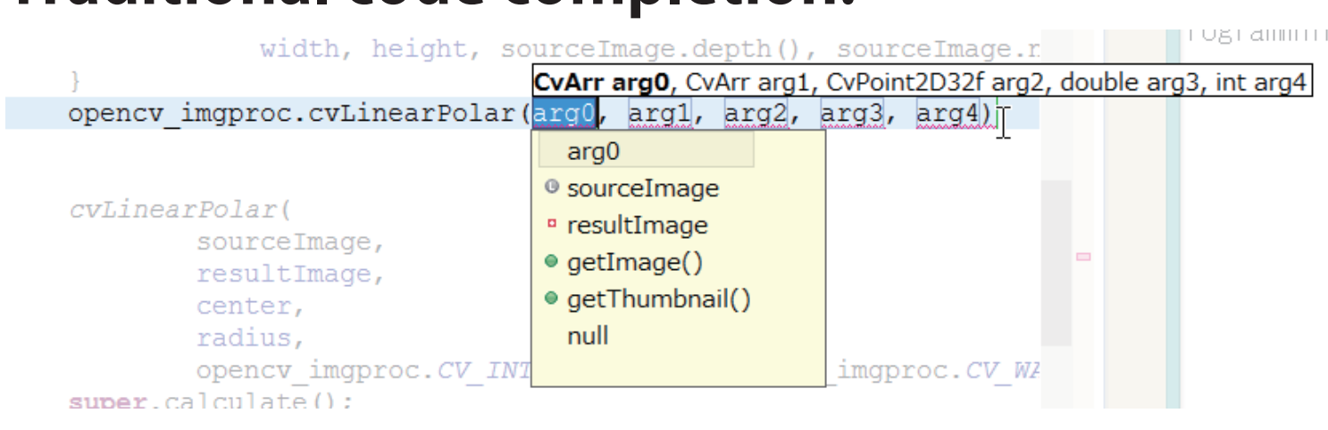
```
// Applying color filter
doSomeTasks();
```

Code completion



Text-based code completion usually provides a list of methods and a template of parameters according to the variable type. In my case, parameters are specified first, and a list of image processings is constructed according to the input and parameter types.

Traditional code completion:



```
        width, height, sourceImage.depth(), sourceImage.r
}
opencv_imgproc.cvLinearPolar(arg0, arg1, arg2, arg3, arg4)

cvLinearPolar(
        sourceImage,
        resultImage,
        center,
        radius,
        opencv_imgproc.CV_INT          _imgproc.CV_WA
super.calculate();
```

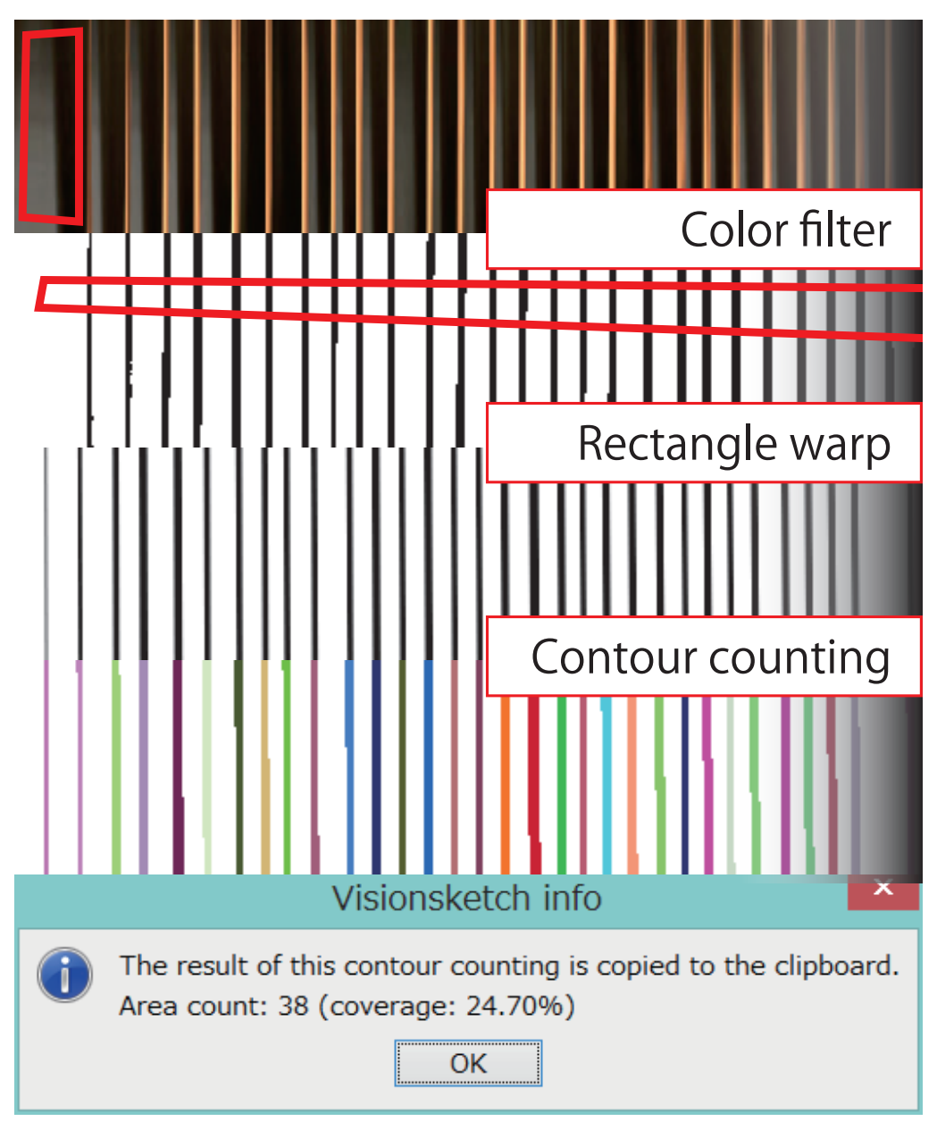
CvArr arg0, CvArr arg1, CvPoint2D32f arg2, double arg3, int arg4

arg0
sourceImage
resultImage
getImage()
getThumbnail()
null

Results and Contributions

Melting the Boundary between PL and UI

Since Visionsketch does not need traditional text-based programming, I expected that it can be used by a non-programmer and conducted a preliminary user study with her. The participant could construct a program to count the number of rotations of the coffee beans grinder. While Visionsketch does not cover all the programming language features (e.g. there' s no "if"), it is feasible for the practical use, melting the boundary between Programming Language and User Interface.



The result of this contour counting is copied to the clipboard.
Area count: 38 (coverage: 24.70%)

More information available at
http://junkato.jp/visionsketch/