

VisionSketch:

Integrated Support for
Example-Centric Programming of
Image Processing Applications

Jun Kato^{1,2}, Takeo Igarashi¹

¹The University of Tokyo

²National Institute of Advanced Industrial Science and Technology

Cameras are ubiquitous

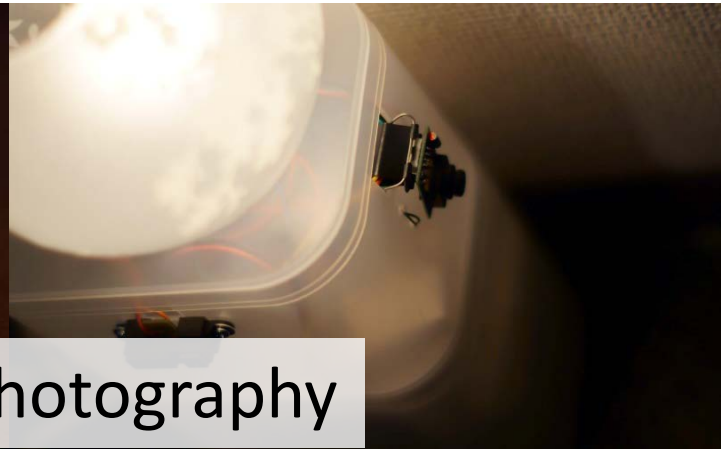
Surveillance camera image quoted from http://en.wikipedia.org/wiki/File:Three_Surveillance_cameras.jpg under CC BY-SA 3.0



Surveillance



Time-lapse photography



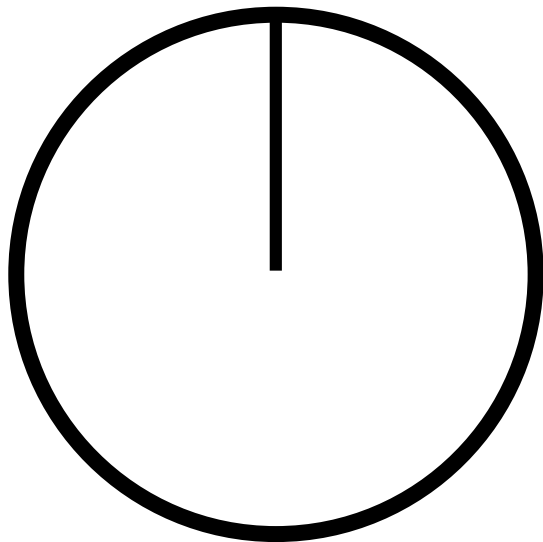
Interesting events
can be detected



Interesting information
can be extracted

Programming is needed

Monitoring 24h/7days?



Computers never get tired



Regarding the variety of desired information,
using **preset** programs is not enough.

Programming is difficult

- Programming in general is to create **abstract** logic
- IDEs are equipped with **textual** interfaces
- Tuning algorithms takes **long time** by iterative cycle of changing code and restarting the program

Programming should be easier

- ~~Programming in general is to create **abstract** logic~~

Start off by choosing a **concrete** example

- ~~IDEs are equipped with **textual** interfaces~~

Get **graphical** feedback with help of the example

- ~~Tuning algorithms takes **long time** by iterative cycle of changing code and restarting the program~~

Update the program and get **immediate** feedback



Example-Centric Programming

Integrated Support for Example-Centric Programming

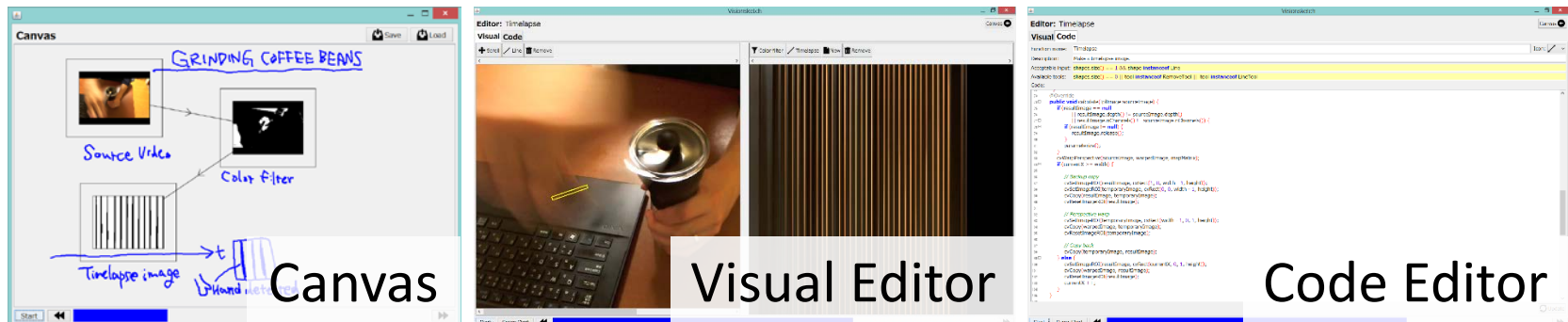
Demo

Target apps: image processing applications (fixed viewpoint)

Goal: address difficulties of current mainstream IDEs by allowing programmers to...

- Start off by choosing a **concrete** example
- Get **graphical** feedback with help of the example
- Update the program and get **immediate** feedback

Method: provide three interlinked interfaces

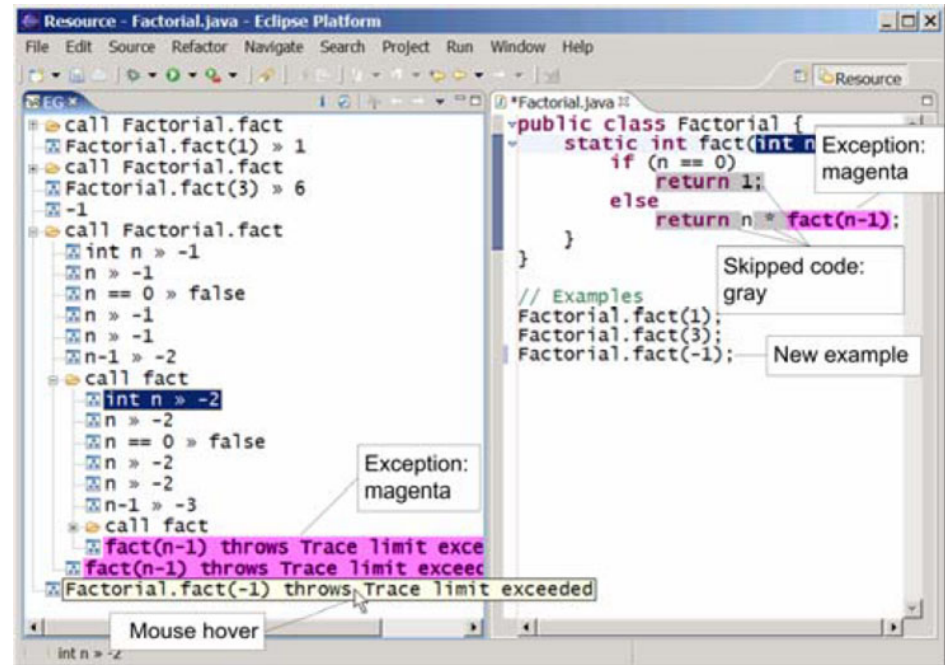


Subtext

[Edwards, OOPSLA Onward '04]

Integrated support for **example-centric programming**

- allows to write incomplete code
- as well as concrete test code



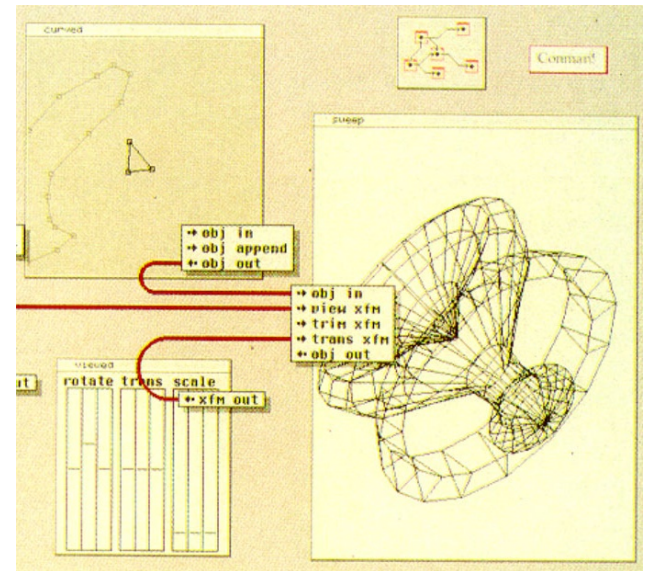
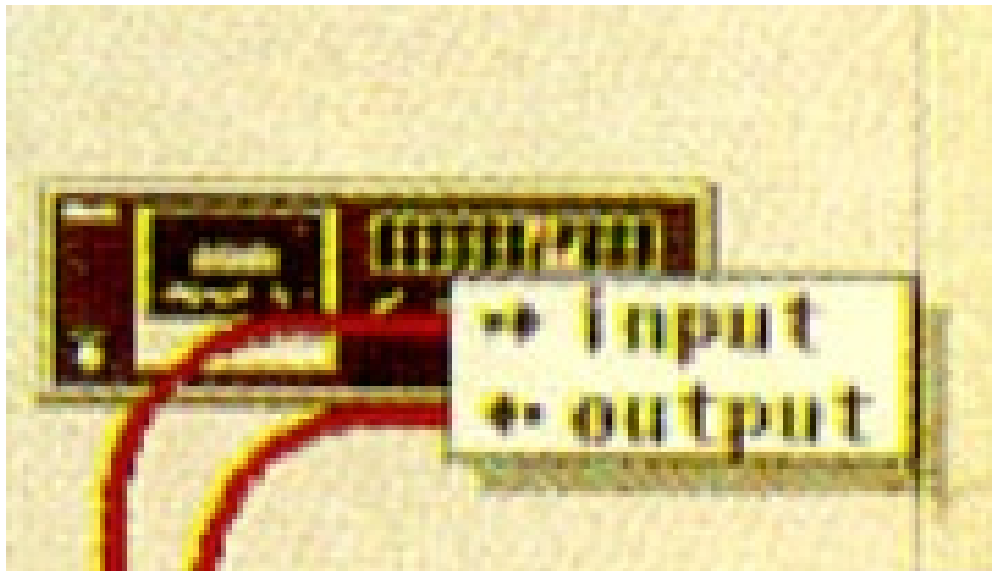
Designed for simple character-based applications

➡ No graphical representations

ConMan

[Haeberli, SIGGRAPH '88]

VPL, **casual program execution** using recorded data



Designed for tuning parameters of CG rendering

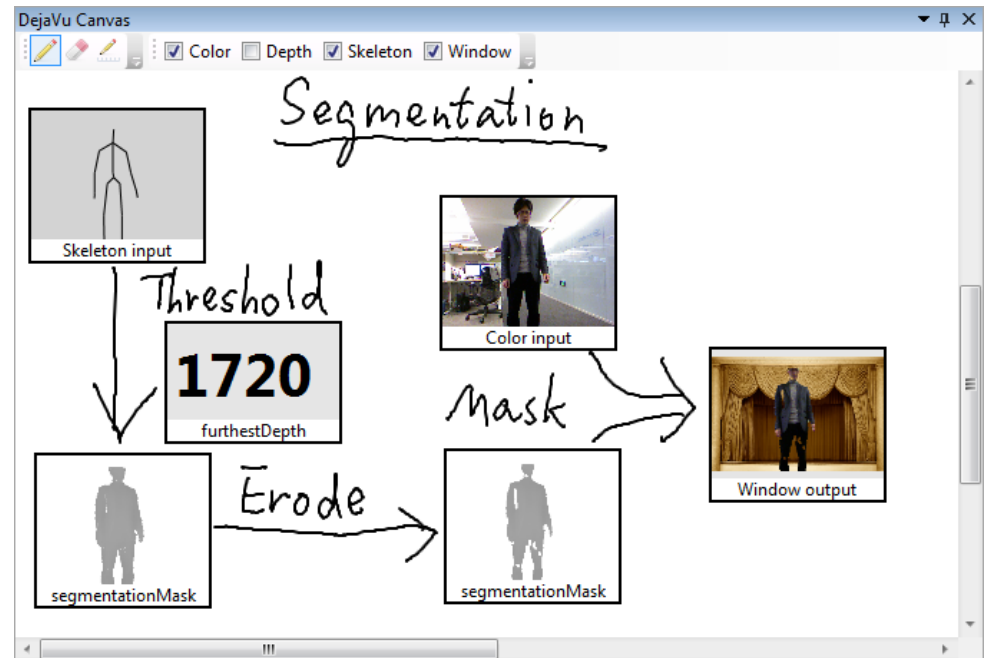
➡ Mere visualization, no graphical editing

DejaVu

[Kato et al., UIST '12]

Canvas interface for visualizing contents of any variables

- during execution
- after the execution



Designed for record & replay of program executions

➡ No support for direct manipulation of graphics

Preliminary user study

Purpose: • To collect user feedback and investigate applications and limitations

Participants: • **5** male programmers with **professional programming experience**, aged 23-36

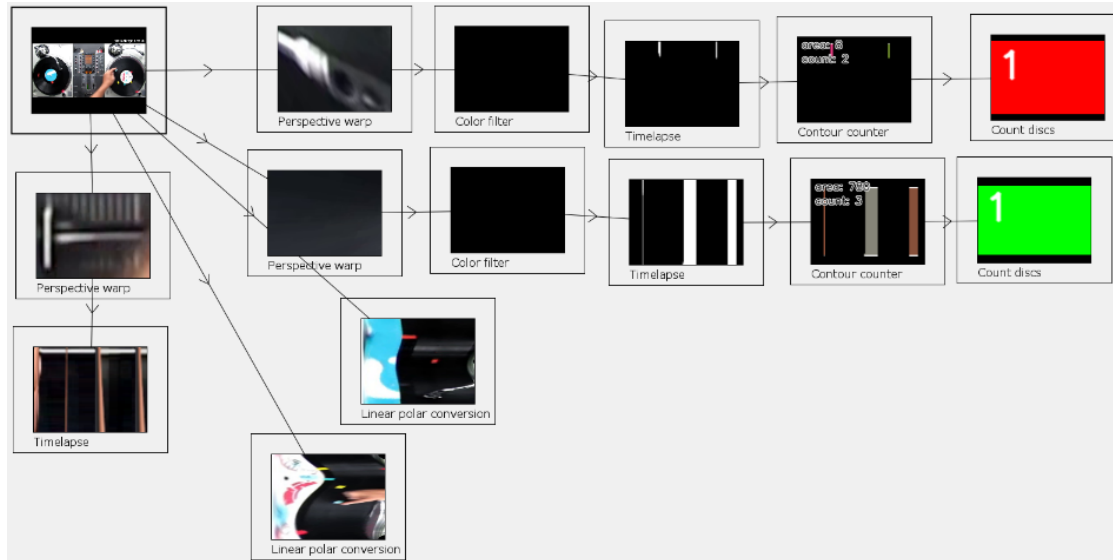
• **4** of them **have used OpenCV** for static image processing

Procedures: • Pre-study questionnaire

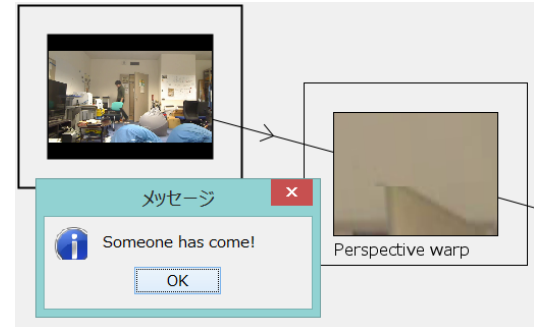
• **Work on a video** (selected based on their interest) to create their own app

• Post-study questionnaire and interview

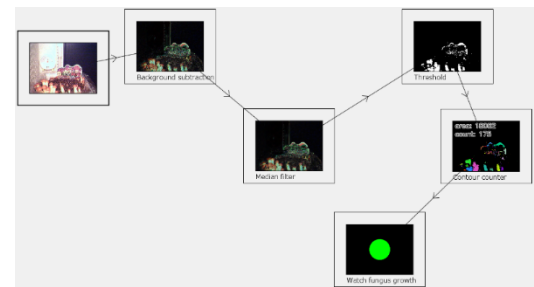
Example applications




Disc Jockey Analyzer  



Door Watcher 



Good-for-eating
Sensor 

Observations from the study

IDE + user code = application

- Normally: toolkit + user code = application
- Suitable for **prototyping** (programmer = user)

Many simple components > a few complex code

- When computational cost doesn't matter...
- Preference for graphical operations over coding

Improvements on code editor needed

- Criticism on not providing graphical feedbacks
- **Combination with past work** (e.g. DejaVu) desired

Limitations and Future Work

Technical limitations

- Images are assumed to be captured from static viewpoints **but the system can be extended to handle dynamic viewpoints.**
- Graphs with loops are not supported **but can be supported.**

Intrinsic limitation

- Example-centric approach cannot be applied to building apps with **real-time feedback loops.**

Integrated Support for Example-Centric Programming

Proposed and evaluated design of **VisionSketch IDE** with three interlinked interfaces to aid **example-centric programming of image processing apps**.



Open-source distribution for Windows & Mac OS X
<http://junkato.jp/visionsketch/>

Canvas

Visual Editor

Code Editor

For sketching
program overview

For drawing shapes to
choose & tune algorithms

For writing code to
implement algorithms

Appendix

Canvas

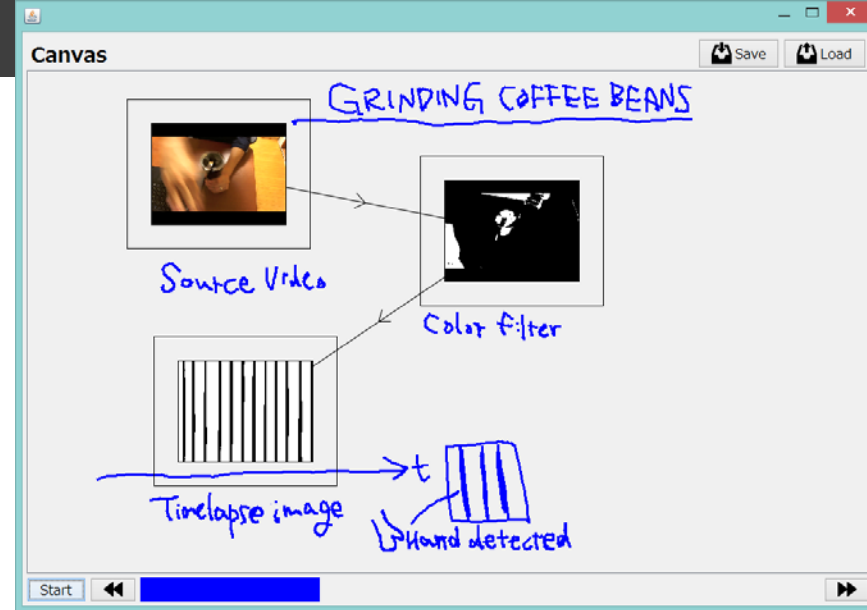
Graphical user interface for
graphical overview

Start off by choosing a **concrete** example

- Program execution casually controlled by the **slider**

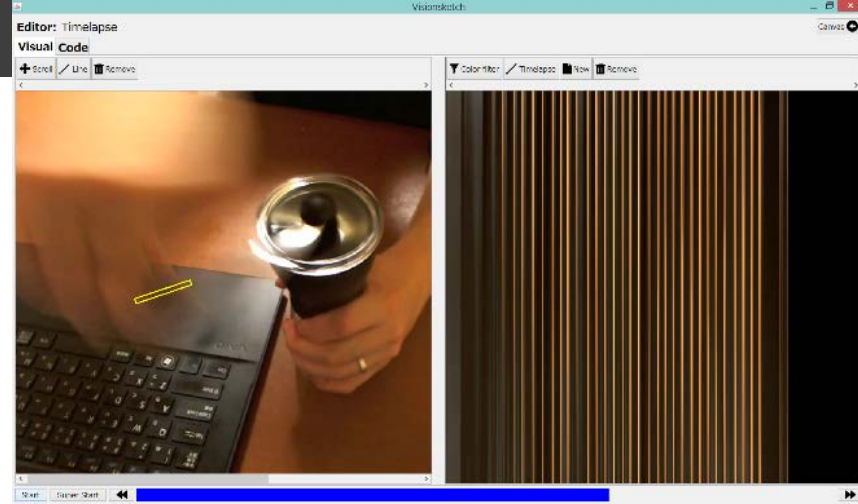
Get **graphical** feedback with help of the example

- Typical visual programming language but **with graphical representations for all components**
- **Freeform comments** sketched on the canvas



Visual Editor

Graphical user interface for **choosing** image processing component and **tuning** its parameters



Start off by choosing a **concrete** example

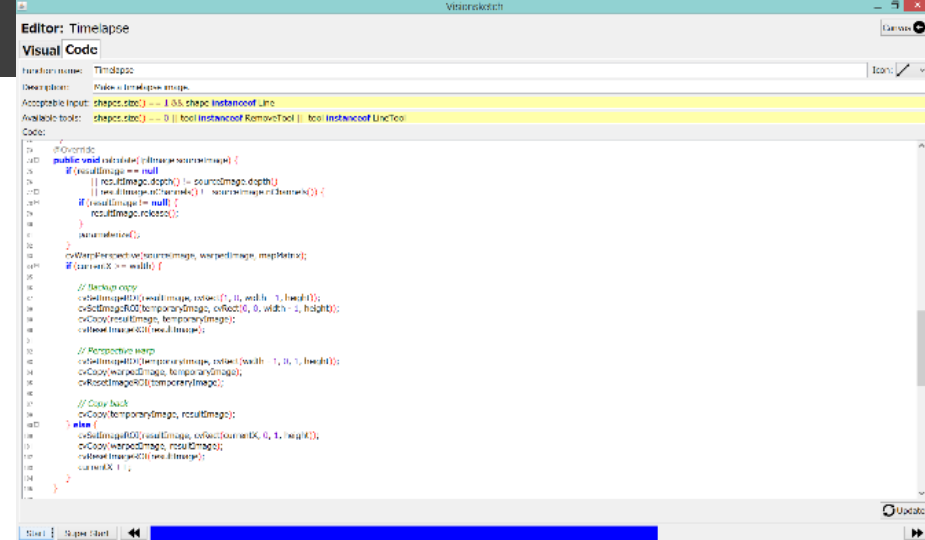
- **Parameter-based code completion** (drawing shapes narrows down the list of applicable components)

Update the program and get **immediate** feedback

- **Interactive graphical feedback** of processing results

Code Editor

Text-based code editor for editing and creating image processing components



Update the program and get **immediate** feedback

- **Seamless switch** between text and visual interface
- Selective updates of corresponding components **without restarting the whole program**