DejaVu:
Integrated Support for Developing Interactive Camera-based Programs

JUN KATO$^{1,2}$, SEAN MCDIRMD$^1$, XIANG CAO$^1$
MICROSOFT RESEARCH ASIA$^1$
IGARASHI LAB., THE UNIVERSITY OF TOKYO$^2$
BACKGROUND

Camera-based programs are getting popular with affordable hardware and useful software libraries.
PROBLEMS

Various visual data
Continuous processing
Non-reproducible input
PROBLEMS

Various visual data

• Camera input
• Intermediate results
• Window output…

Current IDEs:

• Textual value only

Input

Color input
Skeleton input

Image processing pipeline

Erosion operation

Window output

Output
PROBLEMS

Continuous processing

• Frame-by-frame nature
• Temporal visualization desired

Current IDEs:

• Breakpoint is problematic
• Custom visualization needed
PROBLEMS

Non-reproducible input

• Iterative test process
• Same action impossible
• Environmental noise

Current IDEs:

• Just go back and forth between the camera and the PC…
CONTRIBUTION

Enhancement to an IDE for general development of the interactive camera-based program.
RELATED WORK: PROTOTYPING TOOLS

**Exemplar, RePlay**: record & replay of sensor input

[Hartmann et al., CHI’07]  
[Newman et al., UIST’10]

**Crayon, Eyepatch**: external tools for computer vision

[Fails et al., CHI’03]  
[Maynes et al., UIST’07]
RELATED WORK: GESTALT

Programming for machine learning [Patel et al., UIST’10]
DEJAVU INTERFACE
DEJAVU INTERFACE: CANVAS

- Visualizing various visual data
- Showing continuous processing results
- Sketching capability
DEJAVU INTERFACE: TIMELINE

- Recording & replaying to handle **non-reproducible input**
- Synchronized with canvas
DEMOnstration

 KinectDress: virtual dressing room application

 Go close to start
 Move to try-on
 Swipe to change clothing
IMPLEMENTATION

Extension to SharpDevelop IDE

Thin-wrapper of Kinect for Windows SDK

Silently inserting code to track variables

Example) Original: double a = func();
Rewritten: double a = (Double)(IDE.track(func(), 17));
CONCLUSION

- **DejaVu** provides enhanced **integrated support** for interactive camera-based programs.
  - **Canvas**: visualization of current situation + sketch
  - **Timeline**: visualization in temporal fashion + replay
- We received positive feedback from target users.
APPENDIX
EXECUTION MODES

Live execution
Replay
Refresh
EXECUTION MODES

Live execution
Replay
Refresh

Live input
DejaVu API
User program

1720
furtherDepth
segmentationMask

Window output

Recorded session
Recorded input
Variable contents
Output
EXECUTION MODES

- Live execution
- Replay
- Refresh
EXECUTION MODES

Live execution
Replay
Refresh

Recorded session

Recorded input
Variable contents
Output

Live input
DejaVu API
User program

1720
furtherDepth
segmentationMask

Window output
EXECUTION MODES

Live execution
Replay
Refresh

Recorded session

Recorded input

Variable contents

Output

DejaVu API

User program

Live input

Window output
USER FEEDBACK

3 experienced programmers
1 hour trial + informal interview

Positive comments and suggestions

• Simulating and manipulating input
• Visualizing generic arrays
• Composite visualization
EXAMPLE APPLICATION
(KINECTDRESS)

Go close to start

Swipe to change cloths

Move to see fitting

Camera captures frame

Process frame data

Distance between user and camera < threshold \( D_t \)

Segment user’s image based on depth data

Based on the most recent \( n \) frames, recognize whether there is a swipe gesture?

Switch to next virtual suit in the list

Update virtual suit’s position and size

Render application graphics
The diagram illustrates the flow of data through a system:

1. **Live input**: A device, possibly a Kinect, provides live input.
2. **DejaVu API**: This component processes the live input.
3. **User program**: Outputs are directed to a user program, which processes the data further.
4. **Recorded input**: Recorded data is also processed through the DejaVu API.
5. **Variable contents**: This appears to be a variable or a parameter, possibly `furthestDepth`.
6. **Output**: The final output is visualized as a window output.

The diagram connects these elements to show how data flows from input to output.