Webcam as a HID

For fun, I made a doodle desk; any (neutral) surface will do. (Don't use a "wood grain" surface as this causes no end of confusion to the system)

1. Introduction

Summary. Uses webcam and paper cut outs as tangible UI controls.

Related projects

- WCam.
- Touchless – mainly looks for a point size and color.

Outline

- Introduction to the design, my goals, how the system is structured, the controls and actions
- Software Architecture. Structure of properties, events and actions. Video processing, architecture, analysis of visual objects. The audio processing
- Hardware Architecture and Power supply
- How to set up hardware and software build
- Performance profile
- How to get it do something useful
- How to calibrate
- Conclusion

Design goals.

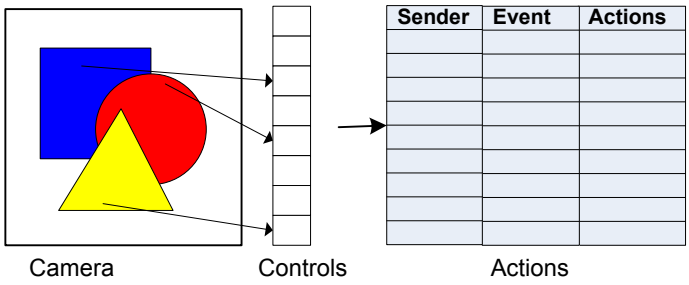
Interaction goals. Appropriate feedback. Immersive in phenomenon (esp. underlying phenomena)

Fun.

Overview of how the system work

Overall, the system maps a scene from the camera to an abstract representation of the controls and finally to actions.

Figure 1: System process overview

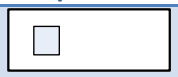


The controls map the scene to events that trigger the actions.

Types of visual controls a user can use / make.

Some examples of the controls:

Table 1: Visual controls

Name	Example	Description
Slider		A rectangle with a circle or rectangle inside can be used to control something from the range 0...1 by sliding the inner shape along the long axis. When the inner shape is obscured, the item is considered clicked, but the value is not known.
Knob		A circle containing a circle or rectangle can be used to control something from the range 0...1 by rotating the knob.
Pointer		A circle with a circle at the center is a mouse, and acts a pointer and select items.
Button		When the inner shape is obscured, the item is considered clicked, but the value is not known. When it is seen again, it is considered released.
Button box		Several buttons, each as part of a group

Actions that be taken

The system connects (loosely) the controls to the following actions:

- Play sound
- Stop sound
- Pause sound
- Skip ahead
- Skip back
- Move mouse pointer

Modules, Directories and Classes

External Parts, SDKs used in the application

- XP, Vista, Win 7, Windows CE
- Webcam (Logitech 9000) $100
- DirectX (Direct Sound), Direct Show.
- webcam.dll
- FFMPEG
- Written in C# (.NET) , since it's convenient to prototype with – there are a lot wrappers to a lot file formats, media types, and actions. And I can publish the steps along the way.

Table 2: Directory structure (directory per module)

Directory	Description
Audio	Objects related playing sound
Colors	Color space definition and transformations
COM	Code to wrap COM interfaces
Controls	
Decomposition	Analysis breaks down image into parts
DirectShow	DirectShow wrappers and helper proxies
Ffmpeg	
Image	Manipulates an image
Math	Math related helper objects
Structure	How the visual object relate to each other
Win32	Wrappers, structures and interfaces specific to windows

Table 3: Main Classes in application

Class	Description
AudioFile	A specific file wrapper
Control	Family of controls a user can use and links to actions.
DSIO	DirectShow IO abstract class
FlipVideo	Flips the video
DirectShowGraph	A wrapper around DirectShow's GraphBuilder, with some helper conventions
DirectShowNode	Most filter nodes, and have class for certain special filters
LightPaint	Figures out where the pen light is
Matrix	A mathematical family of types used to represent matrices and the many transforms.
Overlay	Overlays pen strokes onto a video stream
Vector	A mathematical family of types used to represent vectors, position and its derivatives.
VideoSource	Cameras, mainly. Derived from DirectShowNode
ZObject	The basis for object graphs in this project

Example figures:

- Movie showing mouse movement, clicks, drawing (split screen)
- Movie showing drawing and mouse movement (Split screen)
- OCR and scanned text (before and after)
- Movie showing button like controls
- Movie showing frame key in

2. How the system works, system's structuring mechanisms and things are represented

An outline of the section:

- Structuring mechanisms
- Properties
- Event distribution and handling
- Actions

Structuring mechanism

The system is structured using object graphs as the basis for behaviour. This graph is directed, and represents context and scope. Properties are not retrieved from a specific object – they are effect of the graph. The same is true for events and event handling. Events are distributed in the graph in a structured way, seldom does an object need to explicitly subscribe. Objects themselves are often proxies for framework behaviour or actually implementations of specific behaviour.

Properties are access by name, type, neither or both. Properties are not explicitly classed as essential, intrinsic, or incidental – that is, we don't track whether the property is a key to the identity of an item.

Links to these are primarily thru sending events (which may trigger other events) and fetching properties.

Most of the system relies on ZObject as a base class, which implements the object graph and the basic operations on the graph. (This class also restricts its portion of the graph to a directed acyclic graph.)

There are also Helper objects that represent state and operations: matrices, vectors, and their methods (or delegates to them)

Properties

When a property value is to be fetched, the system searches the object graph in the following order, taking the first answer it finds:

- Checks the item first to see if it can respond
- Checks the linked visual object (if a control)
- Checks the children
- Checks the ancestors (but not their children)

The system can find items by name, type or both name and type.

- The name is a string. If name isn't specified, it is a wild card.
- The type is a System.Type reference. If type isn't specified, it is a wild card. Prefer to use the closest to base that distinguishes type.

Table 4: Property names and types

Property Name	String Name	Type	Description
Angle	angle	double	The angle of rotation
kAudioSink	sink(audio)	object	
kAudioSource	source(audio)		Used to find the audio source of a network
kAudioSource2	source(audio,url)		
Bounds	bounds	Size	
Center	center	Point	
Color	color	Color	
Control	control	Project . Control	
Frame	frame	Size	
kLooping	looping		
kNumberOfChannels	channels		
kNumberOfSamples	length(samples)		In a sound, the number of samples in the audio file
kSampleRate	samples/second		The number of samples per second
kSampleWindow	samples/window		The preferred buffer size, in terms of samples. Multiply by number of channels and size of sample.
kind	kind	Type	
Radius	radius	double	
kVolume	volume		
state	state	ZState	Unknown, pressed, released
visual	visual	VisualObject	

Events

Events are sent using a neighborly distribution, without an event pipeline/queue. Events indicate what has changed, not how it changed, what drove the change nor what it changed to. Two event handlers are called: one (willProcess) to at the start of the event processing, and the other (didProcess) at the end.

1. willProcessEvent is raised
2. Sends will event to children (each child receives a willProcessParentEvent being raised. This is sent in Depth first order: to the senders first child (A), then to A's first child (and so on), then rest of A's children in a similar way, then the rest of the sender's children in a similar way.
3. Sends will event to parent. Each successive ancestor receives a willProcessChildEvent call. This is raised from the most distance first, to the immediate parent last.
4. OnEvent is raised
5. Sends did event to children

6. Sends did event to parent. This is raised from the most distance first, to the immediate parent last.
7. didProcessEvent is raised

Note: processing is stopped if anyone cancels the event. Event handlers should not retain the event structure.

The following kinds of events

Table 5: Kinds of events used in the system

Event	Description
FileEnd	We have reach the end of a file
FrameAcquire	An event sent before and after the receipt or grabbing of a frame
FrameAnalyze	An event sent before and after the scanning a frame
ObjectAdded	An object was added to the list of objects being tracked
ObjectLost	An item that was previously visible is no longer visible
ObjectFound	An item that was previously visible, then lost, is now visible again
BufferReady	
Moved	The items position has changed
SizeChanged	
OrientationChanged	
Pressed	The button was pressed
Released	
ValueChanged	
Start	An event used to start a long running action, like playing a sound
Stop	
Pause	

The event argument structure is includes:

- The kind of event (See above),
- Whether or not the event was cancelled, and
- An optional link to an image frame (if any).

The event arguments are not part of the object graph, although it was considered. (It was dropped for performance reasons)

Actions

Actions are not directly triggered. Some nodes in the graph corresponding to action respond to events. They are triggered by the events sent by controls. Some actions connect directly to the controls so that they get the right control.

Long running actions – like playing a song – are broken down into smaller actions like start, stop and pause. Each of the actions send an event that they are happening.

The actions include (expand table from section 1):

Action
Play audio
stop
Pause
Resume
Skip ahead
Skip back
Scratch sound
Move mouse pointer
Button down
Button up

Controls

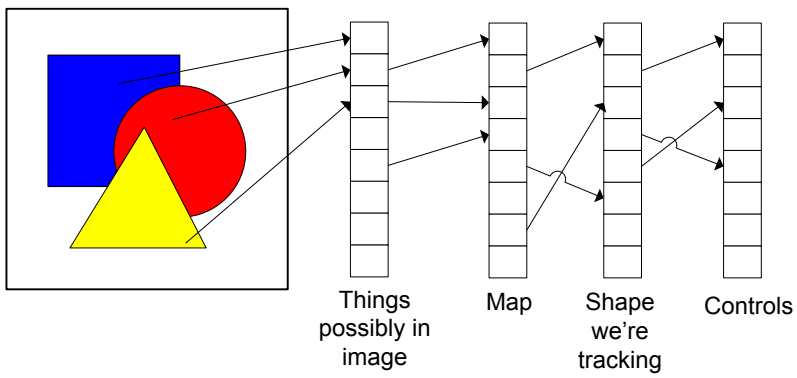
Include some event handler API's similar to System.Windows.Forms.Controls events:

- OnGotFocus,
- OnLostFocus,
- OnMove.

3. Composition of visual objects into controls

The following diagram describes the overall process of mapping a video frame to the controls the system knows about. We will look into more about the controls and how shapes map to a control structure. Then we'll look at the kinds of shapes that can be recognized, and how we determine the shapes, position, orientation, and change.

Figure 2: Image to control processing



A control is composed of shapes, their properties and how they go together. There are two kinds of shapes: rectangles and circles. The nesting (parent-child relationship) to identifying what kind of control it is. Properties relevant to composition include such things as orientation and position, visible.

Controls are identified by their object graph of shapes. A shape is mapped to a control. If it doesn't map to any the system, sees if it is new, looking for patterns in control see if it should add it.

How visual objects go together

Containing relationship (aka parent-child relationship).

Table 6: Structural composition of common controls

Control	Primary Shape	Child	Description
Button	Either	None	
Mouse	Circle	Either	The child must be centered in the circle
Slider	Rectangle	either	
Knob	Circle	either	The child is off center

A control can be contained within another control. For instance, a button is contained within a slider.

Assigning left/right/index

Atomic Shape analysis

Determining what shape an item is. We must answer the following kinds of questions for each shape:

- How we know it is that shape
- How we know it's center (in more precision than the initial estimate)
- How we know its size
- How we know its angle of orientation

Kinds of shape: circle, rectangle

Table 7: Shape analysis

Circle	
How we know it is a circle	edge distance to center has low variance
How we find corners	n/a
How we find center	Key points on edge are used with determinant method to triangulate
How we find size	
How we find orientation	n/a
Rectangle	
How we know it is a rectangle	It isn't a circle
How we find corners	Points at the Min X, max X Point furthest along axis (8 points), quadrant around center
How we find center	Intersection between corners
How we find size	Based on corners
How we find orientation	law of cosines (Arccos) between corners its size and where the corners actually are to find for theta orientation
Triangle	
How we know it is a triangle	?

How we find corners	Not clear. Based on rectangle
How we find center	Halfway along an axis. or Circle determinant?
How we find size	Based on corners
How we find orientation	Law of cosines to tee?

Tracking visual objects

The system tries to keep track of visual objects by shape, color, size and last position.

Principles: shapes are persistent, size, shape doesn't change, color doesn't change. So we go and look for similar shapes; we use a ranking system to calculate perceptual distance, and choose the least far way item.

If you are going to take them off the desk, they need to be unique. Otherwise how would the system know which control it is?

Mapping possible shapes found in an image to the ones we are tracking

Besides looking for an item with the same key characteristics, then choose item least distance away.

4. Image processing

This section includes a discussion of:

- Simplifying assumptions
- Direct Show
- Image processing steps
- Camera stabilization
- Camera correction
- How we map hypothesized items to a tracked item.

Simplifying assumptions

The system is built to work with a 2D surface. This allows me to make many simplifications:

- There will be no depth or perspective issues. No shadow or parallax.
- Items don't change scale.
- The affine angle of rotation is (essentially)the same of for the x and y axis
- There is little or no affine shear in the how an object moves across the surface
- A visual object's color doesn't change
- Color doesn't really matter
- A visual object doesn't change shape

Development Environment

My development environment does not quite match the deployment environment. I'm running it as a C# with DirectShow and NAudio.dll on Windows 7. I'm deploying on Windows CE, with a WebCam.dll and some bits to replace the missing pieces in NAudio.dll.

Direct Show

DirectShow was used on a Windows PC to prototype the visual acquisition. First a quick overview of DirectShow. Software built using DirectShow employs component (object) graphs to do its work, and create the overall behaviour. Some components are the ones I definitely want – video camera– some add features, and a few are needed to connect it all. When the graph is built and run, DirectShow goes thru and makes sure each node agrees on the exact media formats that will be provided. I enjoy object graphs as a design structuring technique, yet I found DirectShow a challenge to master.

In a sense, you lay out the basic schematic of the system – or at least the DirectShow portion – then create code to do the fine details and add functionality. Prototyping a graph (schematic) can be done with GraphEdit (or Monograph EditStudio). These tools have a large catalog of the pieces, making it easy to try different ideas out. Then you can layout the pieces and see if they go together (I found that some pieces just do not) and test it.

Video formats, in order of preference:

- I420 – one buffer for gray scale (luminance), and two for U and V. The other two are half the height and half the width.
- YUY2 – Y0, U0, Y1, V0, Y2, U2, Y3, V2, Y4, U4, Y5, V4: reflects the Bayer pattern
- RGB24 – blue, red, green

Why prefer these? If the hardware can convert it to gray scale for me, why not use it? It's probably faster, more accurate overall. But I need to prefer the kind that the Windows CE environment will have.

Windows CE and WebCam.dll

To understand how we get the camera's video frames, we need to explain Windows CE. There isn't WebCam support integrated in DirectShow on Windows CE. Instead you use a layered approach to construct the appropriate decoder:

- A driver called WebCam.dll (http://cewebcam.codeplex.com/) to get each frame,
- A modules that wraps the IOCTL's used to retrieve the frames
- A MJPEG decoder, based on a modified JPEG image decoder, that decodes each frame into a 3 buffers: one for Y, Cb and Cr. The latter two are half the width of Y (each element represents two pixels)

Table 8: Performance profile, characteristics / properties

Property	Value
Frame rate	30fps
Frame encoding	MJPEG
Frame size	352x288
Frame decode time	20ms[1]
Analysis time	15ms (full), 3ms for unchanged frame

[1] full color decode is 25ms – initially was over 100ms; I expected a much faster decode time.

The smaller frames size(and rate) was chosen to allow decoding and processing to happen fast enough. The features that in the device that will have to be updated:

Table 9: Camera sensor properties

Code	Name	Default	Desired
2	Auto-Exposure Mode	8	0
3	Auto-Exposure Priority	1	0
4	Exposure Time (Abs)	166	1
18	Brightness	128	
19	Contrast	32	
21	Saturation	32	
22	Sharpness	224	
24	White Balance Temp	4000	
26	Backlight Compensation	1	
27	Gain	0	
28	Power Line Frequency	2	
30	White Balance Temp-Auto	1	0

Updates to the driver

lstrcpy and lstrcat had to be updated to "more secure" versions.

There were several spots where a return was missing, and a dodgy check for invalid parameters.

Need to send a note to the author.

Memory was made to reduce the usage.

Check that the memory copies are not a problem for me.

Potential bugs in the driver

The device has some buffering quirks. PddVclas.cpp, line 135 (`pdd_DeviceAttach`)

```
	pPDD->strStreamDefault.dwNumBuffs = NUMDEFBUFFS;
```

PddVclas.h line 70:

```
#define NUMDEFBUFFS 2
```

Okay, WebCam.cpp line 757 (`mdd_SetVideoFormat`)

```
	// Verify that we at least have three buffers
	if (vsData.dwNumBuffs < 3)
		vsData.dwNumBuffs = 3;
```

Shouldn't the default satisfy this constraint?

pddVclas.cpp DoVendorTransfer does a memory copy. This could be done better, but isn't necessary since it isn't used for the video frames.

line 335 seems unnecessary

Count # memcopies, count bytes per transform, count number of frames. Then rate the change

Software Steps

1. The program (and it's constituent's) only allocate memory at the start of allocation.
2. A channel to the camera driver is opened, and the camera is configured for the property settings.
3. The frame rates and sizes are examined to find one suitable for our processing capacity
4. The video frame channel is opened with the selected image size.
5. The software loops in a process
    a. get new frame
    b. decode the frame. Most of the time is spent in the get_bits_2() procedure, rolling the bits around.
    c. Process the frame (described in the next section)

This was prototyped on the PC and develop of the key algorithms were ported stepwise from a PC.

Image processing steps

Steps in image processing:

1. Motion check – look for change in image. This reduces the time per frame in the other portions, keeping the overall frame rate higher.
2. ~~Correct image brightness~~
3. ~~Temp (k) convert image with color correction~~
4. ~~Correct image orientation (warp / sheer)~~
5. ~~Clean image == remove too bright of spots~~
6. Do background identification and removal
7. Separate the image into blobs. Algorithms: flood fill,
8. ~~Do derivatives. Each object gets its own histogram if pixel A and pixel B~~
9. ~~Analyze histograms to find orientation and origin~~
10. ~~Finding edges (to distinguish between circle and rectangle)~~
11. Finding corners, outlying points and estimated center of region
12. Determine the shape
13. Map objects to previous object (past)
14. Report changes: item lost, item observed, moved, rotated
15. Map to a "known" item.
16. Map changes to action. Things can be mapped to: move window, move mouse, button press, pause music, mouse click

The buffer layout is special. The stride of the internal buffers are a power of 2; the extra bytes (between the width and stride) are ignored. References to a pixel are done with an index, rather than a pointer or x-y coordinate. Instead of converting an index to coordinates with:

```
y = Idx / Width;
x= Idx – Idx*Width;
```

we use:

```
y = Idx >> a;
x =Idx & b;
```

This allows a speed up of several ms per frame (about 7ms).

Background Removal

Optimization: could use a stored image – if one is known.

Thresholding method. This is done with a histogram of the image and removing anything that is brighter than 7% of the pixels. This band is found by identifying the histogram of background brightness and find the region that is either known to be interesting or known to be uninteresting.

Finding Regions / Blobs

A blob is linked list of pixels. As scary as that sounds it really is faster for the rest of the analysis.

Linear scan (left to right in each row, and top to bottom).

1. If the pixel is a background, it is skipped
2. If the pixel to the left wasn't background, assign it's region id to this pixel
3. If the pixel above wasn't background, assign it's region id to this pixel
    a. If both had region id's, and they are different, make a note that the two id's are equivalent.
4. If both were background, assign a new id to this pixel
5. Count the number of pixels for each region

After the scan is done, go thru each of the regions and merge all of the different-but-equivalent ids to use a single id. Then throw out the regions that don't have enough pixels.

I also compact the regions id's, since the size of other tables depends on this, and it can consume time to manage them.

Finding the corners of item

- The min and max Y of the regions pixels, no matter the X
- The min and max X of the regions pixels, no matter the Y
- Nominal Upper left: The point whose X and Y, such that there is no other point (P) whose P.X < X and P.Y < Y
- Nominal Lower right: The point whose X and Y, such that there is no other point (P) whose P.X > X and P.Y > Y
- Nominal Upper right: whose X and Y, such that there is no other point (P) whose P.X > X and P.Y < Y
- Nominal Lower left : whose X and Y, such that there is no other point (P) whose P.X < X and P.Y > Y

Nominal? That is assuming an non-rotated rectangle. Rotating makes the corners in different places. There is a check that it may have been rotated and then corrects for that.

Finding these points is a linear scan thru the whole image (!). It isn't perfect.

Optimization; make the image width a power of two (big stride!)

Optimization: make the region a chain of pointers

5. Audio bits

Uses DirectSound (part of DirectX?) to implement:

- Playback
- Audio loops
- Audio effects

Parameters:

- Sample rate: the number of samples per second
- Number of channels (usually 2)
- Sample size: usually 16bits (2 bytes)

These go together to determine the size (in bytes):

$$Size = Sample\ Rate\ \times Number\ of\ Channels\ \times Sample\ Size\ \times Duration$$

The preferred buffer size is a playback duration plus a read ahead. The duration is a value in the range 10ms to 100ms

Buffer size (the amount to read ahead + 1 section)

#sections = BufferSize / Sections size = L / Section Duration

= 1 if perfect

perfect is L < max length to be perfect (may be looping vs single shot)

Event to skip ahead, skip back

Sample Rate correction (matching file to output rate)

May not be necessary. Right now I just change the DirectSound settings at the start and end.

Skip ahead

Parameter how much to skip.

1. Set DirectAudio speaker index ahead to skip over our read ahead buffer
2. Set file index ahead
3. Optional: do we fill in with a scratch sound? Yes: it gives a natural response to the action, either a fixed scratch sound, or sampling.

Skip back

Parameter how much to skip back

1. Set DirectSound index ahead, to skip over remaining sound
2. Set File Index back

It isn't clear how to make the response sound.

Accelerated Playback

The audio speed up
is two factors: number samples on, number of samples skip. The playback rate is

$$Playback\ Rate = \frac{On + Off}{On}$$

To avoid shifting the frequency, we need a minimum number of on samples: SampleRate * 15ms or so. So

$$Off\ = Minimum\ On\ \times\ (Rate - 1)$$

There is no slowing the music down below 1

Playing backward – skip and do rate.

Disk angle makes rate exponential

6. Hardware profile

Size	4"x6"x1"
Power	5volts, 800mA operating + 500mA per USB. A LDO regulator (eg MCP1827s) down regulates the batteries to a constant voltage, and provides clean shutdown when the voltage is too low.
Audio	
Camera	Logitech QuickCam 9000

Characteristics of Camera

Camera selection

Latency	How responsive the system is, or if it will feel sluggish
Focal length	How close the camera has to be to the desk. Limits the size of the desk area.
Aspect Ratio	4:3
Exposure control	Low-light and image correction. If the camera doesn't support it, you'll have to work on the lighting by hand.

Lens, focal length, pixel count are important, but not as much. Color response: Different sources will have narrower or wider cooler gamuts, color spaces, different illuminants, sensitivity, exposure time, etc. This makes for some trying to figure it out.

Windows CE build configuration

Debug:

- IDE storage driver
- Hive registry
- R6040 Ethernet driver
- Realtek-8100 Ethernet
- Compact flash
- 1st serial port
- SPI flash controller
- USB 2.0 controller driver
- USB audio device
- USB mass storage
- Headless
- 256MB Ram
- CAB File Installer/Uninstaller
- .NET Compact Framework 3.5
- CoreCon
- AutoLaunch
- RegFlushApp
- IMGEBOOT
- KITL (no IMGNOKITL)

all else off

The release version has less

other

Faster boot; config.sys autoexec.bat setting

how to remove map and pdb files from release build

windows setting

service packs, installation order, how to install on 7, virtual machine.

parts

Camera Device

Reference frame

Reference dFrame

Background (est) – image of background (estimated). This is built up over time

mask – indication of whether pixel is known or unknown

Camera Correction

Stabilization. Not implemented

Handling the shape of the desk as seen in the camera. Calibration / de-warping

Option is to either dewarp the whole image, and process that… or accept the distortion and apply the change to the few key points later. Currently a preference for that.

No color correction yet

Orientation. Correct skew/angle based on calibration (possibly pre-correct image). Use simple background (flat color, no texture, wood). Employ key points on display (desk) to recalibrate screen. Check ensemble of visual objects to see if there is a consistent movement between dp, theta, using that to recalibrate

Color and detail. Use remaping of RGB to correct range (formula/lookup). HDR algorithm. Cycle exposure with HDR to increase dynamic range. Oversampling window.

Size can only be increased

Angle can be updated. First match by rotation 45deg (pi/4) then if delta > threshold

We don't need scaling, perspective effect, shear, or different thetaxy angles

Optimization. use an image difference to figure out which areas have changed (and scan only those)

Resolution boost. For items, zoom in on corners and analyze for finer resolution

Convert trichromate to scalar (e.g. color temperature)

Estimating camera / desk orientation

- look at each shape's corners in original. Use size, orientation + M to map to ideal
- Use a subset of these points to calculate new M0. This will create interesting artificial frequency injection into system, so will need a filter.

Matching color

Graph of the visualization

Diagram of the memory format.

Flow of mapping pixels to a thing and the semantics

Each mapping involves one or more steps.

What we can update (after mapping to items()

Position can be set (if delta > threshold)

7. Implementation

Parameters aren't specific to source (too much). They try to be in the units and range of one of the buffer representations. Some work to get source to buffer in a consistent fashion. Conversion parameters.

The different sources will have narrower or wider color gamut (With respect to each other), different illuminants, sensitivity, exposure time, etc. This makes for some fun in try to figure it out.

Configure similarity thresholds

Relevant algorithms

Kinds of objects in the ensemble. Audio source. Audio sink. Something that knows how to handle knobs/sliders.

Example of a slider controlling volume.

Example of mouse.

Processing Steps

1. Get image frame from video stream
2. Process the image, analyze into changes in controls
3. Update state of controls and take action
4. Queue new work

Image / frame stabilization

Not implemented yet

1. Do HP edge filter
2. Select key points to find offsets with respect to reference frame (convolution)
3. Compute Mframe for shaking (represented as transform)
4. Multiple M0 (the transform to flat rep coefs)
5. Use these to build flat, stabilized image

Parts

What kinds of sliders, button, point. Sliders come in any number of variations – button in box, knob like item. Circle or stroke in knob . Those can be circle or rectangle. Pointers are better as circle (I believe it is more intuitive to tell where the center is)

Configuring what the controls do, go the panel, select configure controls. Grid of controls and what they do. Find the control you are inherited in and then change the action it is associated with. Then close the window and move on.

Non-affine transform

Grabbing an image of an item

Scanning for a QR code

QR could mean sound url, sound label, NPC characteristics

Persons VCard