Computer Vision for the OU Fish Visualization Project

Status Report 6-16-2006

Wesley Smith

Introduction

The OU Fish Visualization Project consists of two main systems: a computer vision system and a visualzation system. The computer visualization system's task is to identify and track fish for extended periods of time. It processes video data from cameras trained on fish tanks and outputs tracking data to the visualization system in real-time. The visualzation system then interprets the data, producing imagery across a series of monitors.

In order for the visualization to be expressive, it needs expressive data. The term expressive data desribes a particular relationship among data streams denoting a set of data streams with rich interconnections spatially, temporally, and critically. In the case of fish tracking, this means not just providing fish IDs with location over time, but also information describing the algorithm's process over time such as statistical parameters from various stages of the image processing unit. An initial set of data parameters includes number of fish being tracked, their IDs and position, the time of the data, how many blobs does the algorithm see, how similar is the current frame to the background, and the percentage of pixels in the image the algorithm see movement in. As the algorithm develops, more data streams can be added to reflect new stages in the process.

Computer Vision Problem and Approach

The problem of tracking fish over long periods of time is a twofold problem of detecting fish in an image and identifying individual fish from one frame to the next. The first problem is the most straightforward. Fish are always moving a detectable amount, even when they're sleeping so they can be identified based on a simple persistent motion tracking algorithm. The identification of individual fishes across frames is a much trickier problem with lots of special cases that need to be properly handled. For instance, if two fish move close to each other and then split apart, how can the system tell which fish has gone where or what happens when a fish hides and reappers a few minutes later? These kinds of issues can't be addressed by a generic algorithm. Instead, a part of the algorithm must be context aware and recognize that two fishs' trackpoints have essentially merged and will split shortly or that a fish has just hidden itself and will reappear from the same spot soon.

The persistent motion tracking algorithm for detect fish is basically a fancy background subtraction algorithm. The first stage of the process takes an incoming frame and subtracts it from a background image. This is then

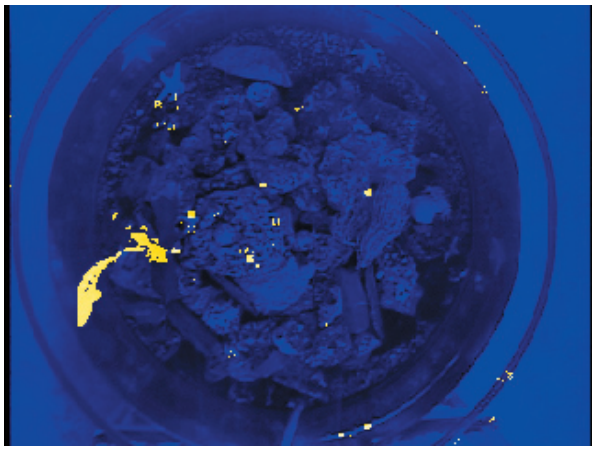
Basic Image Processing Schematic

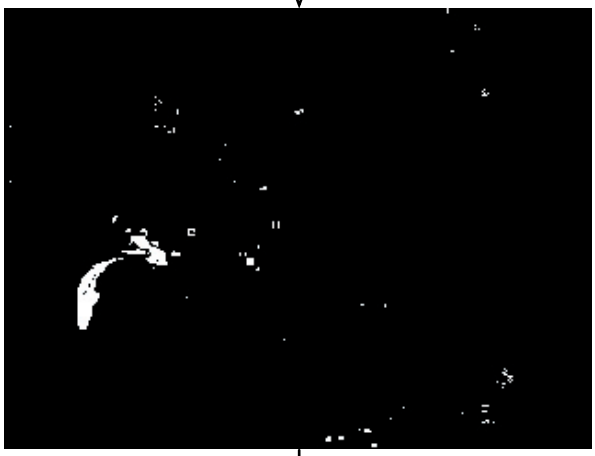
Raw Image

Background

Difference

Raw and Difference



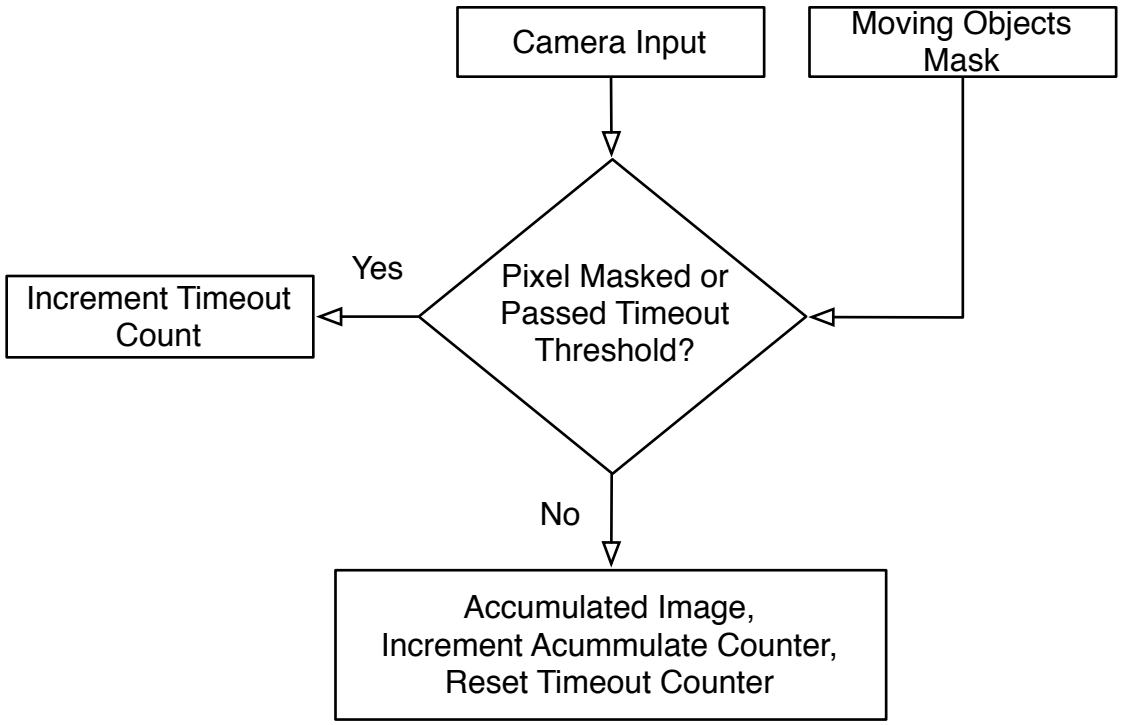


Blob Filter

thresholded and passed to a blob filter that keeps only shapes of a certain size. The result is a rough segmentation of fish from the video image. As a final step, the segmentation mask is fed back in to the system to update the background. In other words, it is used as a motion mask for preventing foreground objects from being added to the background image. When a new frames comes in to the system, the motion mask is used to pull out only background pixels and those pixels are accumulated into the background image.

The motion mask feedback is not flawless however and in some cases can result in "motion mask creep". This is where the motion mask prevents any new information from being added to the background making the background information more and more out of date. If the motion mask is identifying a fish, this is not a problem since identifying a fish is the goal, but if the motion mask is erroneously marking a location where some other type of movement is occurring such as water motion or changes in lighting, then the erroneous segmentation will actually be amplified as the segmented region grows.

Fortunately the fact that the fish are constantly moving can be used to prevent this kind of mask creep. If a pixel hasn't been updated for a given period of time because it has been constantly blocked by the motion mask, the motion mask is overridden and the pixel is updated anyway. If the pixel the next new pixel at this location



Adaptive Background Algorithm

is still different enough (meaning it's likely a fish), then it will still be detected as a motion pixel. When added to the system, this timeout feature vastly improved the segmentation results by reducing the number false positives.

The algorithm for identifying and tracking of the fish between frames uses results from the from the segmentation stage as its input. It takes the motion mask and determines shapes that are likely candidates for being fish. As the shapes move from one frome to the next, it marks the path the shapes are taking by giving unique IDs to them the persist over time. Although seemingly simple there are many cases where the tracking algorithm can break down, causing a tracked shape to be lost or confused with another shape. Scenarios where this might hap-

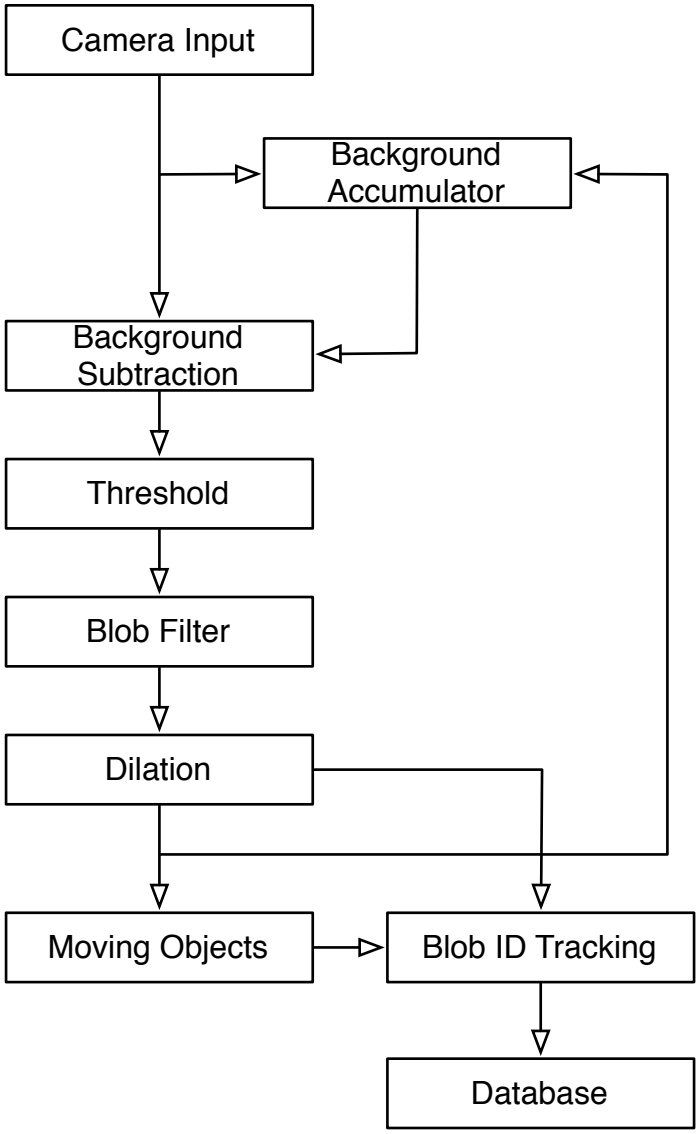


Image Processing Flow Diagram

pen include the case where two fish come together as indistinguishable shapes and then move apart. Identifying which of the two fish went in what direction can be difficult in this case. Also, if a fish hides or is acculded from the camera, the tracking system has to be able to detect this and be aware enough to wait for the fish to reappear in a logical location for retagging the shape with the old ID so that the data points representing the fish's location over time are consistent. Information such has far a fish can move between frames in any given direction can also be used to properly pick out which shapes are actually fish by giving a likelihood estimate based on the previous frame's results.

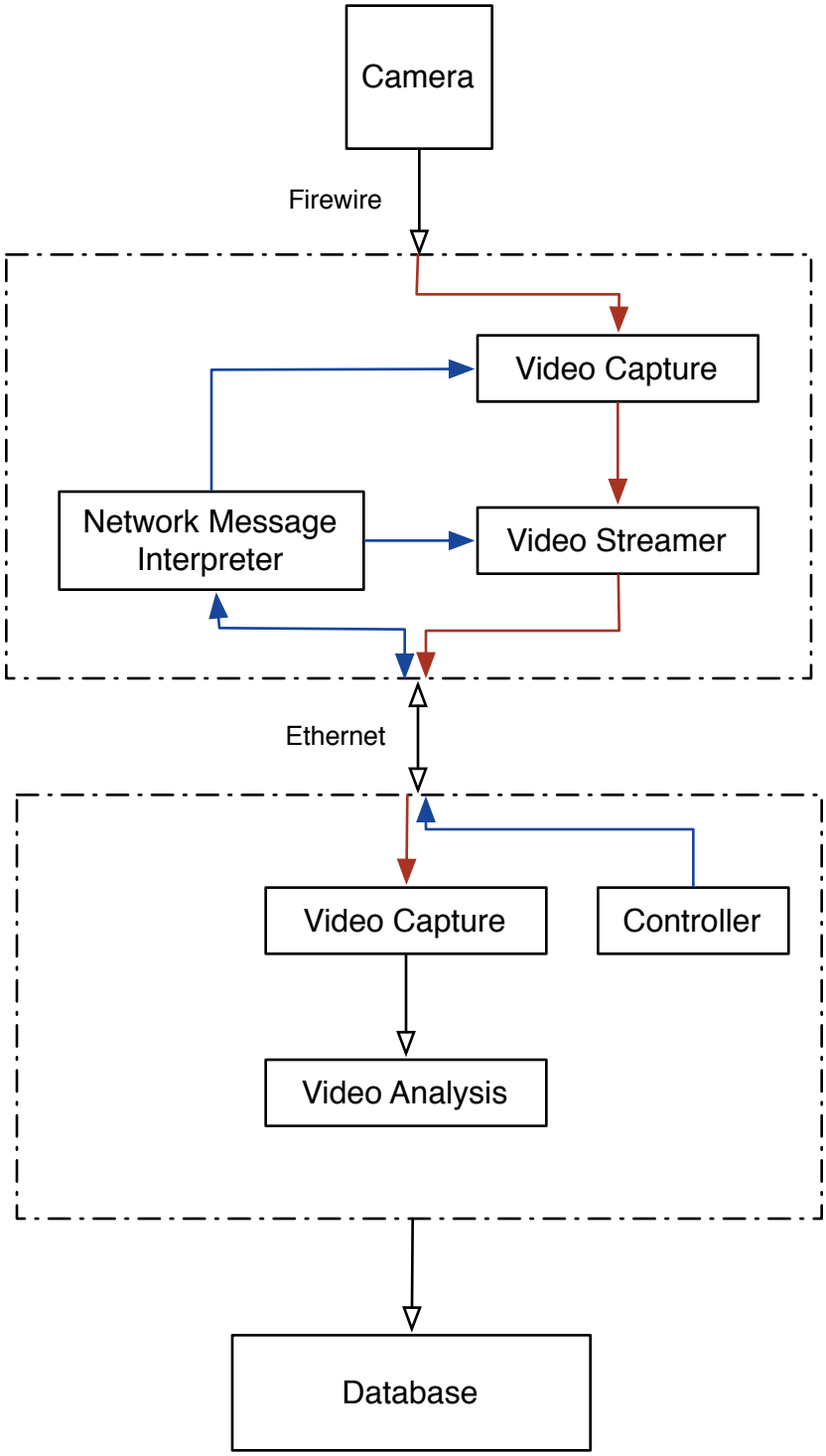
Currently, only a basic ID and point tracking algorithm is in place. The scenarios described above as needing special attention have not been addressed and will likely cause some errors or inconsistencies in the tracking data. The current state of the tracking system can however provide feedback to the segmentation algorithm, improving results on that end. The shapes marked as fish can be used as a probability mask for filtering the background from the incoming frame in addition to the stages that are already in place, providing a sanity check for those results.

Current System

The system that is currently in place for testing the fish tracking algorithms uses MaxMSP/Jitter for image capture and streaming as well as analysis and data storage. Installed in the MSI REEF aquarium center is a ceiling mounted and a networked computer setup for streaming the video to a remote machine for processing. A Jitter patch running on the capture machine allows remote control of the capture settings such as framerate, color or grayscale image, size of the image, and IP address to stream the video to. A complementary patch has been built that connects to the machine and sets these parameters. In addition, an SSH server has been setup on the machine for remote login and administration of the machine. The machine can be reset and the software updated through the SSH mechanism. When the machine starts up, it automatically logs in and launches the capture and streaming video patch. If this patch is replaced with a new version, the next time the computer is booted up, the new vesion will be launched and the changes will take effect.

On the processing end, a video receiving patch grabs the video stream from the network and runs it through a tracking and segmentation algorithm, generating data in realtime for the visualizations. The video image is

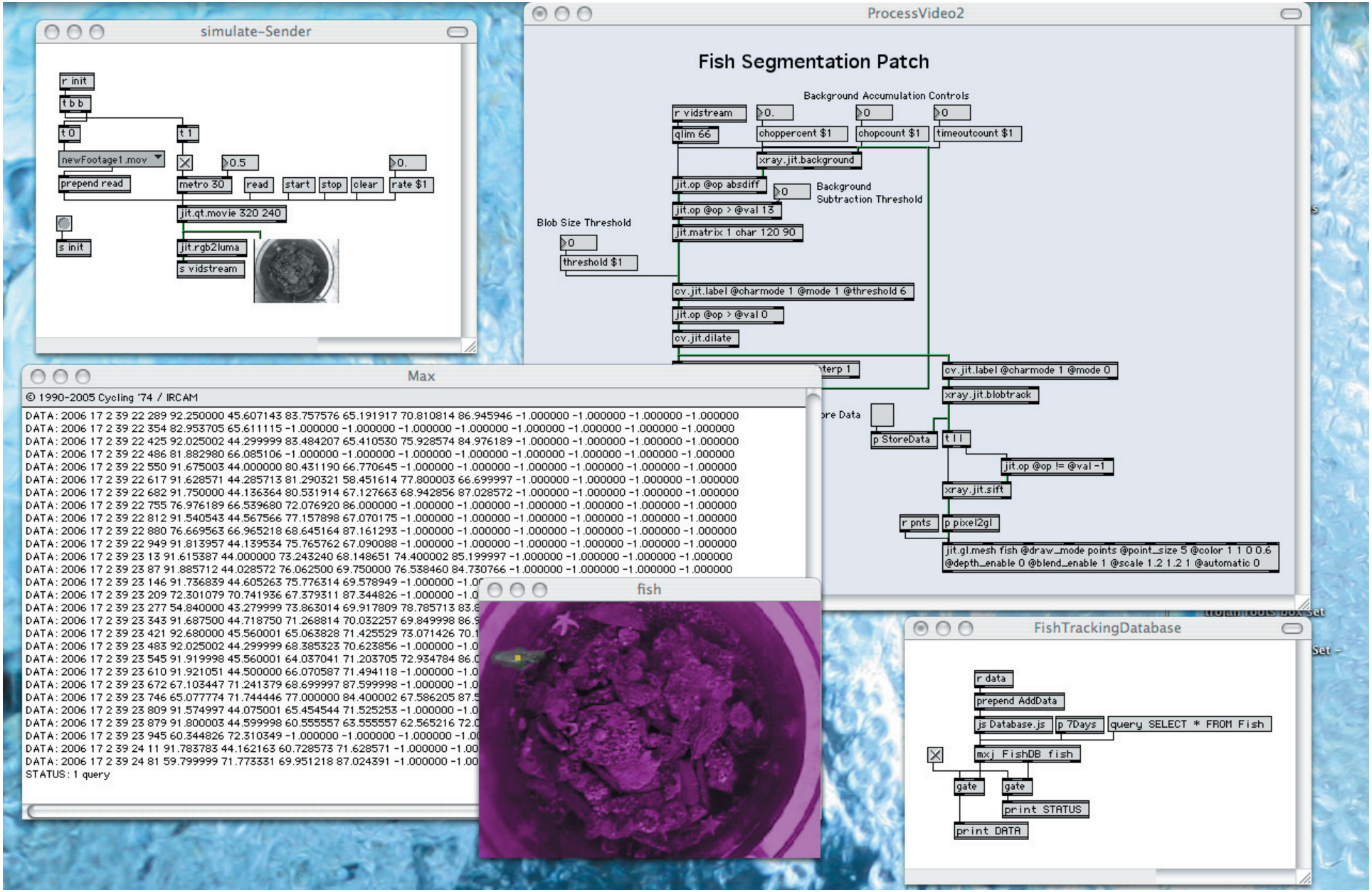
passed through the computer vision system described above and the results are stored in a database (or streamed to the visualization system when that's running). The database is persistent across sessions, so old data can be queried if needed and piped into the visualization system.



Remote Capture System Schematic

The database is an embedded SQL database called HSQLDB. It is loaded in an mxj (a Max JAVA object) upon instantiation. It's data is stored across a small set of files on disc which are automaticlly read when the database patch is loaded. Due to current limitations of the mxj HSQLDB implementation, the TIME and TIMESTAMP SQL fields are not used to store the time a data point is recorded. Instead, separate fields for Year, Date, Hour, Minute, and Millisecond provide the time a datapoint was recorded. Currently, up to 5 simultaneous points are recorded at any given time although this can easily be changed if more or less data points are needed.

In any given frame of analysis, one or more might not be seen by the system, resulting in less data points. In this case, a (-1, -1) is given for the (x, y) location of the fish in the image. Otherwise, coordinates are given in pixels. Currently, this range is [0, 120) for x and [0, 90) for y. To query the database, standard SQL commands are used to acquire the desired data.


simulate-Sender
r init
t b b
t 0
t 1
newFootage1.mov
0.5
0.
prepend read
metro 30
read
start
stop
clear
rate $1
jit.qt.movie 320 240
s init
jit.rgb2luma
s vidstream
ProcessVideo2
Fish Segmentation Patch
Background Accumulation Controls
r vidstream
0.
0
0
qlim 66
choppercent $1
chopcount $1
timeoutcount $1
xray.jit.background
jit.op @op absdiff
0
Background
Subtraction Threshold
jit.op @op > @val 13
Blob Size Threshold
jit.matrix 1 char 120 90
0
threshold $1
cv.jit.label @charmode 1 @mode 1 @threshold 6
jit.op @op > @val 0
cv.jit.dilate
cv.jit.label @charmode 1 @mode 0
xray.jit.blobtrack
p StoreData
t l l
jit.op @op != @val -1
xray.jit.sift
r pnts
p pixel2gl
jit.gl.mesh fish @draw_mode points @point_size 5 @color 1 1 0 0.6
@depth_enable 0 @blend_enable 1 @scale 1.2 1.2 1 @automatic 0
Max
© 1990-2005 Cycling '74 / IRCAM
DATA: 2006 17 2 39 22 289 92.250000 45.607143 83.757576 65.191917 70.810814 86.945946 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 354 82.953705 65.611115 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 425 92.025002 44.299999 83.484207 65.410530 75.928574 84.976189 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 486 81.882980 66.085106 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 550 91.675003 44.000000 80.431190 66.770645 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 617 91.628571 44.285713 81.290321 58.451614 77.800003 66.699997 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 682 91.750000 44.136364 80.531914 67.127663 68.942856 87.028572 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 755 76.976189 66.539680 72.076920 86.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 812 91.540543 44.567566 77.157898 67.070175 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 880 76.669563 66.965218 68.645164 87.161293 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 22 949 91.813957 44.139534 75.765762 67.090088 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 23 13 91.615387 44.000000 73.243240 68.148651 74.400002 85.199997 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 23 87 91.885712 44.028572 76.062500 69.750000 76.538460 84.730766 -1.000000 -1.000000 -1.000000 -1.000000
DATA: 2006 17 2 39 23 146 91.736839 44.605263 75.776314 69.578949 -1.000000 -1.0
DATA: 2006 17 2 39 23 209 72.301079 70.741936 67.379311 87.344826 -1.000000 -1.0
DATA: 2006 17 2 39 23 277 54.840000 43.279999 73.863014 69.917809 78.785713 83.8
DATA: 2006 17 2 39 23 343 91.687500 44.718750 71.268814 70.032257 69.849998 86.9
DATA: 2006 17 2 39 23 421 92.680000 45.560001 65.063828 71.425529 73.071426 70.1
DATA: 2006 17 2 39 23 483 92.025002 44.299999 68.385323 70.623856 -1.000000 -1.0
DATA: 2006 17 2 39 23 545 91.919998 45.560001 64.037041 71.203705 72.934784 86.0
DATA: 2006 17 2 39 23 610 91.921051 44.500000 66.070587 71.494118 -1.000000 -1.0
DATA: 2006 17 2 39 23 672 67.103447 71.241379 68.699997 87.599998 -1.000000 -1.0
DATA: 2006 17 2 39 23 746 65.077774 71.744446 77.000000 84.400002 67.586205 87.5
DATA: 2006 17 2 39 23 809 91.574997 44.075001 65.454544 71.525253 -1.000000 -1.0
DATA: 2006 17 2 39 23 879 91.800003 44.599998 60.555557 63.555557 62.565216 72.0
DATA: 2006 17 2 39 23 945 60.344826 72.310349 -1.000000 -1.000000 -1.000000 -1.0
DATA: 2006 17 2 39 24 11 91.783783 44.162163 60.728573 71.628571 -1.000000 -1.0
DATA: 2006 17 2 39 24 81 59.799999 71.773331 69.951218 87.024391 -1.000000 -1.0
STATUS: 1 query
fish
FishTrackingDatabase
r data
prepend AddData
js Database.js
p 7Days
query SELECT * FROM Fish
mxj FishDB fish
gate
gate
print STATUS
print DATA

The previous image is a screen grab of the system components running on recorded data. A simluation patch has been built that simulates the way the streaming data interacts with the image processing system for offline work. The main workhorse is the image processing patch (blue background). It segments fish from the image and outputs up to 5 track points depending on the how many fish it sees. For visual feedback, a display window shows the raw video image with the fish highlighted and a yellow trackpoint indicating the location of the fish.

The third patch in the screenshot contains the database. If a record switch is turned on, data will flow from the image processing system and be recorded into the data base for later retrieval. Values are stored with the current time of day up to millisecond exxuracy for proper sequencing of data upon retrieval at a later time.

Future Work

There are many improvemnts that can be made to the system from augmenting the algorithms for new scanarios to making the algorithm more adaptable and self-adjusting. The current algorithm uses a series of threshold values to process the video image. As an improvement, these values could be made adaptive based on the state of the system. For instance, if the image changes drstically because of new lighting conditions, the background subtraction threshold could be adjusted to compensate. If not enought blobs pass through the blob filter, the size threshold could be lowered so that more blobs are seen. Ideally, these automatic adjustments would be tied to measurements taken on the system so that the tracking results remain consistent throughout varying image conditions.

Another possible improvment might involve using a stereo tracking system to identify where fish are depth-wise in the tank. A stereo tracker would not only allow fish to be tracked in the Z-direction, but also provide redundancy in the tracking algorithm by providing verification. It remains to be seen however whether it is worth the expense and complexity of handling two simultaneous video streams. A stereo systemw ill require twice as much input bandwidth and potentially twice as much processing power which may overburden the machines.