#IP #gigatrack designed for abitrary nnumber of cameras, each patricle has diameter/brightness and velocity

flow from https://turbulence.pha.jhu.edu/

make sure flow matches up with stuff form JHU

fun asides

f stop

resolving power is decreased, larger f-stop means more diffraction ## computational power vs cost very low computational demand, so not really a worrt ## can run on lyceum maybe can probably just run on my PC, dont need to worry about RAM VRAM might be an issue for STB, we'll see # JHU scripts pull data from database and convert into a form that can be used as an input input parameters in the input area L drive for stuff, can use it but its slow kinda ignore them, can just have the input data

calibration scripts

TCF calib(2)

have to define the setup (where are cameras, what can they see, apeture) uses JHU data to create a long strip of cameras

need to do stuff for chromatic aberration, colour spill saves stuff for synthetic image generation ### OTF Optical Transfer Function is how a point source of light appears to the camera plots show the focus of the point source of light at different depths from the focal plane

stored as a look up table discretisation for integration of brightness, fairly obvious

cross contamination

there is already a bit projects geometric centre onto the pixels in code, it is just an identity matrix bayer filter is tranfer matrix multiplied by the red, green and blue layers

think of each channel as a matrix transfer function is identity matrix multiply the channel with the transfer function to get the output with identity matrix as transfer funtion, this is a 1-1 mapping

not implemented too well only implements one camera model, no space for chromatic aberration might be easy to add aberration

adjacency matrix

checking if cameras have overlapping fields of view speeds up PTV later

plot_syn_calib

plots the calibration, along with particle density in the volume good way to review the configuration ive just made shows where the cameras can see

synthetic image generation (RGB folder)

rgb_sig_init

writes everything to a calibration file give it a camera model give it all the particles

rgb_sig

reads stuff from [[#rgb_sig_init]] and does the generation similar to mandelbrot-adjuster vs mandelbrot-renderer

gives stuff without going through a bayer filter

processing

proc_rgblpt

similar to the way [[#synthetic image generation]] works, makes a calibration file for low density stuff uses stuff similar to star tracking, assumes that local areas stay mostly the same

batch_rgblpt

actually does the lpt is a wrapper that calls **rgblpt**, makes everything (embarrasingly) parallel takes less time than the image generation assume that the acceleration is mostly small saves stuff that looks bad ### path estimation do a taylor expansion of the path Δt is very short, so this is a reasonable assumption know typical displaceemnt from characteristic velocity and Δt typical acceleration is $a' \times \delta t^2$ because Δt is small, it is assumed that acceleration is negligible ## plot_{} ### plot_snapsho plots the channel flow form a certain number can show the ones that look bad crom [[#batch_rgblpt]]

STB

rgbipr files as opposed to rgblpt try to reconstruct entire position and velocity thing at once STB does it in timesteps

big least squares fit, position is

tcf_mean

finds the means and the variance of the velocity at different points between the wall not going to vary streamwise or perpendicular because the flow is tiled, only varies vertically from the wall good for chekcing for ghost particles, if its in the reconstruction but not in the reference its a ghost particle # camera calibration can look at checkerboard, has imformation on spacing of camera mostly to be aware of for now, will be more relevant for the experimental stuff ## tcf_generate_synthetic_calibration_images generates images of the target for a camera array

qrcode_calibration

searches through the images for the qr code finds the coordinate of the corners dewards the QR code from the views ## multicamera_calibration takes in the files from [[#qrcode_calibration]] and does the calibration figures out the position and rotations between the cameras uses standard stuff in matlab easy to do bwtween two cameras, needs more work to get it between 3 or more

test_

basically like unit tests

todos

git repo will be updates with EVERYTHING i will get access to the NAS

wrapping up

PHD students

will join me in january they will be doing other things, will be useful to be able to talk to them january gets scary

test data

will be on the NAS already has some data on the colour spill demosaicing done in matlab dont bother with the demosaiced data, just use the raw files to find the transfer function

better to discuss in meetings

can send prep docs before meetings

less text on slides remember that the audience knows nothing