Embedded LS-PIV for Measuring Stream Flows

Ashish Dwivedi, Kyle Liang, Justin Nguyen, Jun Taguchi

***Abstract*—Many Internet of Things (IoT) applications, environmental monitoring for example, require deploying sensors or devices into the field. These applications require special attention to the energy budget because devices designed for this purpose should be battery powered and survive for years. To help the design of these energy-aware system, CommonSense has been developed as an extensive computing platform.**

United States Geological Survey (USGS) has been interested in monitoring streamflow by calculating the discharge amount. The flow analysis methodology is based on optical imagery and Large-scale particle image velocimetry (LS-PIV) algorithm. LS-PIV allows us to calculate the velocity of features in these images. However, low-memory implementations of this algorithm (utilizing direct cross-correlations) can be computationally intensive, $O(N^4)$, and cannot be handled by embedded devices that have a limitation on RAM and parallelism. As a result, LS-PIV is used as a post-processing method thus far. An embedded implementation of LS-PIV will enable deploying a real-time sensing system onto static platforms, i.e., bridges, or mobile platforms, i.e., drones.

In this paper, we target CommonSense as a computation platform and propose the optimized PIV algorithm with two operation modes: "Graded" PIV for the velocity field measurements and Discharge Measurement PIV for the discharge measurements. Both operation modes reduce the amount of computation by focusing on the critical part of the images to yield accurate results.

The estimated battery life of the proposed algorithm with two modes varies from six months to two years if the device with a 12,000 mAh LiPo battery reports the result ten times per hour, exceeding the current measurement rates of USGS stream gage stations. The velocity field measurement mode is more expensive than the discharge measurement mode because the number of image windows used for computation is large. The energy for computation is dominant in our system, so future iterations of the project would be focused on reducing the energy of computation.

***Index Terms*—Internet of Things, IoT, particle image velocimetry, PIV, Large-scale particle image velocimetry, LS-PIV, duty cycle, energy aware computation**

I. Introduction

The USGS measures stream flows since they are critical for long-term tracking and modeling/forecasting to ensure that federal water priorities and responsibilities can be met. Effective management of the Nation's rivers thus relies upon discharge data provided by USGS stream gages, but the gage network is sparse and has a relatively low update rate[1]. In addition, data collection during high flows can place personnel at risk. For all of these reasons, the agency seeks to develop innovative, non-contact methods for measuring river discharge. PIV is an algorithm that can be applied to this remote-sensing task and is actively being researched for these types of applications [1].

PIV is a widely used algorithm used for measuring stream flows using optical data, but can be computationally intensive, so it is performed as a post-processing step after data collection [1]–[3]. Currently no open-source systems exist which can perform PIV in real-time on the remotely deployed sensors which collect the data and performs PIV analysis to report the velocity or the discharge rate of the stream.

The team we collaborated with at the USGS using PIVLab [4], [5] to run the PIV algorithm on imagery captured by helicopters [2]. PIVLab runs on MATLAB, on desktops, which is obviously very far from the real time reporting of the desired data. The USGS wants to develop an embedded solution for the task which can provide real time data capturing, preprocessing on the data, PIV run on the images, sending back the data on the USGS centers. This project however, does not focus on the first two steps. We start with the preprocessed image and provide the velocity and discharge data using PIV.

II. Stakeholder Requirements

A. Expected Use Cases

There are two major use cases that our stakeholder, the USGS, is interested in applying PIV: imagery taken from a stable platform (like the side of a bridge) or imagery taken from a camera mounted to a mobile platform (such as a UAS, or helicopter). In the former, the images taken are very stable and do not need image stabilization or alignment [1], [3]. In the latter, the images need to go through an additional step of stabilization and/or alignment before they can be used for PIV algorithm.

B. System Requirements

The USGS want an embedded system that can perform PIV on-board at a reasonable rate and with a long lifetime for remote deployments. The system should be able to measure the discharge rate and the field velocity of the flow. More specifically, they expect our team to meet following expectation:

1) PIV implementation with decent accuracy compared to the baseline
2) Improvement in the current measurement rates which is 15 minutes[2]
3) Lifetime of the system should be > 3 months

The results from PIVLab [4], [5] in addition to ground-truth data measured via ADCP will serve as our baseline.

[1]https://www.usgs.gov/mission-areas/water-resources/science/usgs-streamgaging-network

[2]https://waterdata.usgs.gov/nwis/rt Accessed November 2020

III. Project Scope

Given the stakeholder's requirements, we define the possible use cases of embedded LS-PIV system: static platform and mobile platform cases. In this project, we focus on the the static platform case, which is an essential first step of realizing LS-PIV on in both use cases. We break down this objective and define the following three goals.

The first goal is to create an embedded implementation of the PIV algorithm targeted to run on real-time embedded systems. As mentioned previously, LS-PIV can be computationally intensive, and there are no versions targeting real time use. Thus, we targeted CommonSense as a platform, and developed the PIV algorithm which can be deployed to a static platform such as bridges. The accuracy of the result is the metric to evaluate our implementation.

The second, and most challenging, goal is to optimize the algorithm and software structure for minimal energy consumption. The device should survive for years. In this sense, computation time is also critical; as computation time decreases, we can save more energy.

The third goal is defining the energy budget and software interface to the CommonSense board and designing the final product. Our study focuses on core algorithm development for implementing PIV and subsequent work on the implementation of a real-time LS-PIV system can build on the algorithm described here.

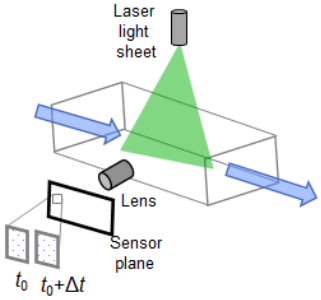
We proposed two operation modes: discharge measurement mode and velocity field measurement mode to achieve these goals. These modes will be discussed in section V.

A. Alaskan Dataset

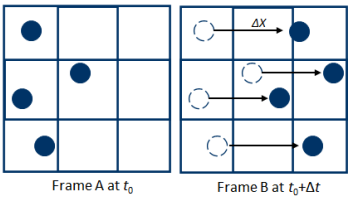
The image data set we used for analysis are of size 1580x2880. These were taken periodically at a 1Hz frequency aerially from a helicopter of the Tanana River and have a 15cm/pixel conversion rate [2]. The dataset is comprised of raw image sequences and image sequences that have been aligned, geo-referenced, and preprocessed to maximize contrast of the sediment flow used for PIV [1].

IV. PIV Algorithm

Particle Image Velocimetry (PIV) is a mature and essential velocity measurement technique in fluid dynamics. PIV is used extensively in academic laboratories as well as industrial settings. PIV was originally developed for use in laboratory plumes where the flow can be seeded with particles and illuminated with a laser light sheet [6], [7]. Velocity is estimated by comparing successive images and identifying the displacement that maximizes correlation between frames A and B. As with any velocity, the PIV algorithm essentially measures the distance traveled by the fluid, $\triangle\mathbf{x}$ over a specific time period $\triangle t$, and uses the well know formula of $\mathbf{v} = \triangle\mathbf{x}/\triangle t$ [6]–[8]. To track the displacement, $\triangle\mathbf{x}$, the PIV algorithm tracks specific points in the fluid. To get the points, a PIV setup typically requires a camera setup vertically over the flow of the river [1], [3]. Depending on the illumination in the flow there may or may not be any additional light source. In our case, since we are measuring the flow and discharge of the river, the dataset we are using described in section III-A we have ambient sunlight and hence do not need any additional light source. A typical PIV setup is as shown in Figure 1.



(a) Laboratory PIV setup with laser and camera



(b) Two successive frames used to measure the velocity of the fluid

Fig. 1: Typical PIV setup and measurement

PIV algorithm starts with picking a pair of sequential images for performing a 2D cross-correlation. Optionally the resulting cross-correlation matrices can be aggregated (summed) in a technique known as ensemble correlation [6], [7], [9]. The peak search and subpixel peak search is applied to the final correlation matrix to yield velocity vectors. They are described in detail below:

1) *2D Cross Correlation:* 2D cross correlation between two images essentially takes a sliding window dot product of two images (where one image is fixed and other is shifted horizontally and vertically before taking the dot product). It should be noted that both the images are essentially pictures of the same frame taken at different times. The output of the sliding window dot product is another 2D matrix, we call that CCF which computed as shown in equation 1 presented in [6]. Here $f(m,n)$ and $g(m,n)$ represent the first and the second subsampled images, respectively, M and N represent the number of rows and columns in the images. $\bar{f}$ and $\bar{g}$ represent the mean image intensity of the interrogation window.

$$C(i,j) = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}[f(m,n)-\bar{f}][g(m+i,n+j)-\bar{g}]}{M \times N} \tag{1}$$

The $CCF(i,j)$ essentially captures the cross correlation between the image1 and shifted image2. If the image1

and shifted image2 are similar, the cross correlation value will be higher. Thus, a higher value of CCF depicts how much has image1 moved horizontally as well as vertically before image2 is captured. The maximum value of i and j are called the overlap value and size of images over which the correlation is performed is called the *interrogation area*. The interrogation area size is specified in length units of a side length. The cross-correlation can also be realized in the Fourier domain which is the preferred approach as explained in Section V-B.

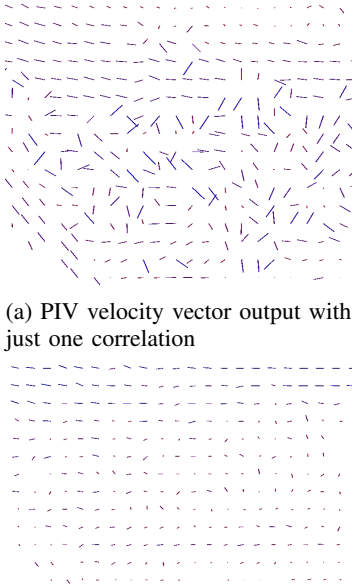
2) *Ensemble Correlation:* Although this is generally optional in many laboratory PIV applications, given the noise in our dataset due to using moving sediment as our seeding particles [1], [2], it is a requirement in our application. To suppress noise, cross correlation (step 1), is performed over multiple pair of images the sum is taken for all the CCF matrices before moving to the next stage. Ensemble correlation can be expressed in equation 2.

$$ensemble\_CCF = \sum_{i=1}^{ensemble\_size} CCF_i \qquad (2)$$

In our experiments, we have chosen $ensemble\_size = 16$, which is statistically shown to be the minimal number of ensembles needed provide a strong peak in the output correlation matrix [7]. This means we are operating on 32 unique pairs of images, finding 16 unique pairs of CCF, and taking their sum. To clearly demonstrate the need for ensemble, an example PIV output is shown with or without the ensemble correlation in Figure 2.

3) *Peak search:* Once we have the ensemble CCF (or single pair if we don't ensemble), we find the peaks of the correlation. Since, CCF values depict how closely image1 and shifted image2 resemble, peaks in the CCF tell us i and j coordinates with which image2 must be shifted to get the highest correlation. That (i, j) pair, gives us one of the possible displacement, $\triangle\mathbf{x}$, of the fluid in the given time duration.

4) *Sub-pixel Peak Search:* The quality of PIV results are dependent on getting rid of various types of error sources. One such error is - under resolved optical sampling of the particle and discretization of particle positions in to integer pixel locations. Sub-pixel curve-fitting suppresses this noise using a gaussian (parabolic is another option) curve to interpolate the displacement vector in the subpixel level [6]. More discussion and optimization of this step in not explored in this project. Interested readers can refer to [6], [10]. This step finally produces the velocity vectors that we can visualize in a GUI.



(a) PIV velocity vector output with just one correlation

(b) PIV velocity vector output with 16 correlation and ensemble averaging

Fig. 2: Two PIV outputs - one without ensemble and one with 16 ensemble average. This noise in the 2a clearly shows the necessity of the ensemble averaging

V. Embedded PIV Implementation

Generally PIV is used in controlled laboratory environments, but a modern application of PIV is the so called Large-Scale PIV (LS-PIV). This variant of PIV focuses on imagery of large bodies of water usually taken from an UAS [3] or a helicopter [1], [2] rather than small, usually microscopic particles, imaged in a laboratory setting [6], [7]. An additional constraint for applying LS-PIV that has not been adequately covered by the literature is its implementation on embedded systems using cameras or optics that would not be feasible for a long-term, remote deployment. Generally PIV is applied as a post-processing step after imagery has been collected [1], [3], but there is a need for a PIV algorithm which can be applied in real time. This would enable possibilities such as dynamic tweaking of PIV parameters to best suit a particular measurement, or allow for remote deployment of LS-PIV sensors which are resource constrained in battery life, network bandwidth, and/or processing power.

We will introduce our implementation which describes optimizations to the PIV algorithm to enable the use of this algorithm on resource constrained embedded-systems. We begin by outlining the hardware constraints of the CommonSense platform on which we developed our algorithm. The CommonSense "stack" includes accurate power measurement circuits, dubbed *PowerDue*, for analyzing power and energy usage of our implementation. CommonSense utilizes an Atmel SAMD51, a very common 32-bit microcontroller widely used in embedded applications. We will then analyze the PIV algorithm and detail where possible reductions in computational complexity, memory complexity, or software architecture can be included to make the PIV algorithm feasible for low-power edge-compute devices. Finally we will show the ac-

TABLE I: PIV Memory Utilization

Interrogation Area (px)	Direct XCorr	FFT XCorr
64x64	32KiB	64KiB
128x128	64KiB	256KiB

curacy impact of these optimizations and outline how various optimization parameters can be tuned for the each specific application or deployment.

A. Hardware Limitations

The ATSAMD51 processor on the CommonSense platform is a 32-bit ARM Cortex-M4 microcontroller which runs at 120MHz; it includes a floating point unit and has variants which contain up to 256KB of SRAM. Unlike our baseline provided by the MATLAB PIVLab package [4], [5] which runs on the desktop (like all other PIV implementations), our algorithm does not have the luxury of parallel processing or a large-register vectorized instruction set. The constraint of the hardware imposes the requirement that the PIV algorithm must be tractable for real-time scenarios by maintaining a linear complexity, as only single PIV windows can be computed at a time.

With these constraints in mind, an analysis of the raw memory utilization of the PIV algorithm for a single interrogation area would yield the memory utilizations described in Table I. Note that the direct cross-correlation memory utilization is computed with storage for two input window buffers, the output overwritng on of those buffers in addition to using 32-bit integers. An actual realization could reduce memory usage if the cross-correlation was properly bounded allowing for a smaller integer type to be used. The FFT cross-correlation is computed with storage for four window buffers. Four windows are required since no direct real-to-real 2D Fourier transforms exist and a complex intermediate Fourier representation is required. Two buffers are used per interrogation area to store the real and imaginary values. The assumption that the output overwrites one of the input buffers is held in addition to using 32-bit floats.

Legleiter and Kinzel have done a parameter optimization for the PIVLab implementation of the PIV algorithm for the LS-PIV application using helicopter imagery and sediment seeding rather than manually introduced tracer particles and have found that across many frame rates, an interrogation area of less than 50 pixels gives heavily biased results [2]. Interrogation areas greater than 50 pixels (taken at a sufficient measurement frequency) yield similar results regardless of interrogation area. Additionally larger interrogation areas being preferred up to 100 pixels where the improvements in result are not as noticeable. As will be discussed in the V-B, the FFT cross-correlation is the only approach tractable for our application which constrains our interrogation area to be a power of two. Taking a 1Hz imaging rate, the 64x64 pixel interrogation area is the smallest window that must be supported to yield acceptable results. It is also important to note the use of KiB units; using the direct cross-correlation, interrogation areas greater than 128x128 can fit in memory while FFT cross-correlation approach would not fit at the same interrogation area in to 265KB of RAM (especially with other application code). Due to these constraints, utilizing the much preferred 128x128 pixel interrogation area is not feasible on our microcontroller without some modifications to the implementation such as:

- Using Q15 (16-bit) fixed-point floats or 16-bit half precision floats: may affect accuracy
- Split Radix FFT: may also affect accuracy
- Store parts of the interrogation in off-board memory and swap where needed: slow

For our application and using a microcontroller which has DMA, the last option may still be feasible as only a few tens of kilobytes would need to be stored outside of RAM.

B. Complexity Evaluation

Our optimization techniques take advantage of several application specific assumptions in the use cases described previously, each of which includes several caveats for how the algorithm could be optimized.

Firstly, for the static sensor configuration (if this system is deployed on the side of a bridge, for example), the position of the camera relative to the body of water is known. In addition the areas of interesting flows will be known before hand (or learned dynamically) and will generally not change over time. These facts will allow us to only perform PIV for a cross section of the image a few interrogation areas wide. Secondly, we know that velocity fields near the bank will be much more turbulent than at the center of the channel, so we can optimize our PIV to perform more ensemble near the banks and fewer at the center without greatly changing the result. Lastly, for discharge measurements, only a single highly accurate velocity vector is needed at the deepest part of the channel; this allows us to have a second measurement mode that focuses in a smaller area and can yield a more accurate result with more ensembles.

On the CommonSense platform, we started with an implementation of PIV by Benjamin Pelc[3] which was written in C++ and utilizes the direct (spatial) cross-correlation. For a single 64x64 pixel interrogation area, this operation took about 13 minutes and 30 seconds. This is not feasible for the real time applications that embedded PIV would be typically deployed for; additionally the long run time would severely reduce battery life. This is especially true if ensembles or multiple interrogation areas are desired, which is always the case. This result does match our $O(N^4)$ complexity for the direct cross-correlation.

We modified Pelc's imeplementation to use the FFT cross-correlation yielded much more tractable results, taking 0.2 seconds to run PIV for one 64x64 pixel interrogation area. Even with this much improved result, it is still not feasible to perform this window-by-window algorithm on large areas of the image. As a comparison, the vectorized FFT

[3]https://github.com/benjaminpelc/pivelocimetry

cross-correlation implementation that is used by PIVLab in MATLAB, takes 1.23 seconds for an entire image worth of interrogation areas.

C. Graded PIV

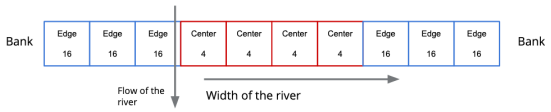


Fig. 3: Illustration of the Graded PIV method. Center interrogation areas are showing in red, while edge interrogation areas are showing in blue. The numbers in each interrogation area is the size of the ensemble which may be changed. The ratio of center to edge interrogation areas is also variable.

From the assumptions listed previously we present a "graded" version of PIV which distributes the density of ensemble measurements to areas with more turbulent flows as illustrated in Figure 3. This method reduces the number of PIV computations required by reducing the total number of ensembles. Since we are working under the assumption for a bridge mounted system with both banks in view, our graded PIV method yields three optimization parameters, the ratio of "center" interrogation areas to "edge" interrogation areas, the number of images to ensemble for the edge interrogation areas, and the number of images to ensemble for the center interrogation areas. Since we also know that the velocity field vectors should transition smoothly from window to window, we can further smooth our result (or interpolate bad results) by applying a Gaussian kernel or any kernel smoothing function.

D. Discharge Measurement PIV

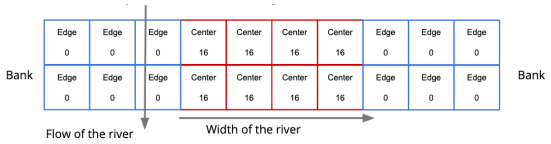


Fig. 4: Illustration of the PIV algorithm optimized for discharge measurements. Note that the size and location of the focused area is variable in addition to the size of the ensemble. In this realization all interrogation areas are considered part of the "center".

We also implement a version of PIV which is primarily aimed at providing single source, high accuracy, measurements which are suited for measuring discharge of the stream. This measurement mode focuses on a configurable number of interrogation areas in a specific portion of the frame. Since the interrogation area size and step size is already specified globally for the PIV algorithm, the parameters for this optimization technique is the number of interrogation areas wide to perform the surface velocity measurement for measuring discharge and the number of ensembles to use. For this measurement mode, it is expected that the number of measurements to ensemble is quite high to ensure a high accuracy result. Compared to the graded PIV technique described previously, if the size of the measured area is kept small in relation to the length of the channel, fewer PIV computations are required, reducing the time to yield a measurement and also reducing the power required. Also, compared to other surface flow measurement methods for discharge measurement, the USGS uses radar measurements [1], which like our discharge measurement mode, takes measurements at a single spot. An advantage of our optical PIV method over radar is that is is possible to change the location of measurement on the fly, possibly in an automatic fashion.

E. Rectangular Interrogation Area

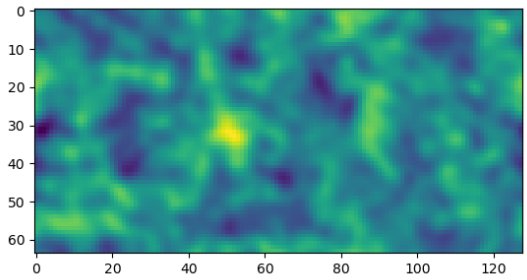


Fig. 5: Rectangular PIV cross-correlation matrix of centered about window 2610 from the Alaska Dataset [2] that is 128x64 pixels.

A final optimization that we have implemented in our PIV algorithm is the ability to use non-square interrogation areas. This can be visualized in Figure 5 which matches the baseline in Figure 6. For the dataset we were analyzing, we found little variance between the result using a non-rectangular interrogation area. For our application, the use of rectangular windows can offer two improvements.

If the interrogation area is elongated in the direction of stream flow, the additional data in the direction of movement would allow for more particles to "track" which can yield more accurate results. If the interrogation area is elongated in the direction perpendicular to stream flow, the total number of interrogation areas can be reduced by increasing the step size between adjacent interrogation areas. This reduction would reduce computation time and increase battery life.

F. Accuracy Evaluation

To test the accuracy of our implementation, we compared the results of the cross-correlation and final displacement vector to that of the PIVLab implementation.

As seen in Figures 6 and 9, it is clear that these optimizations do not effect the FFT cross-correlation in any way.

As seen in Figures 8 and 9, it is clear that our implementation matches that of the baseline provided by PIVLab for the Alaska Dataset [2].

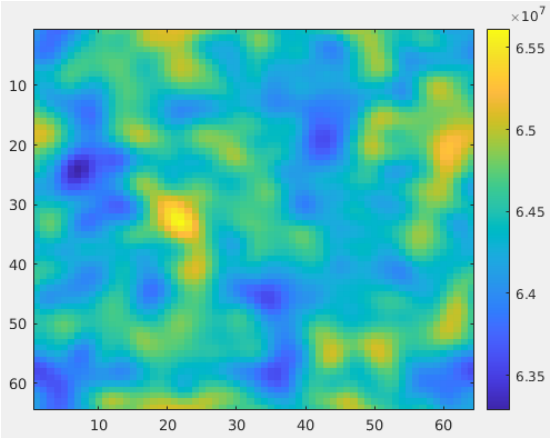


Fig. 6: Cross correlation matrix of window 2610 from Alaska Dataset [2] from PIVLab.

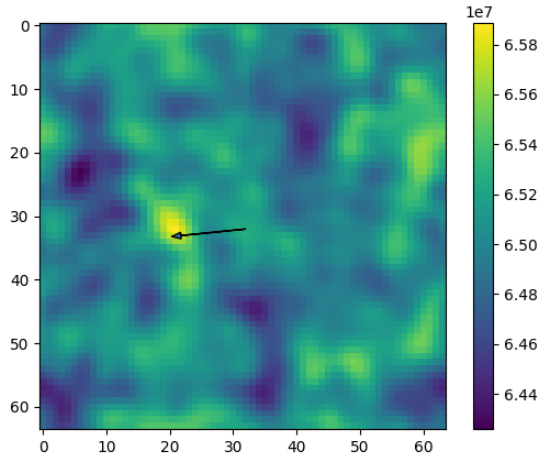


Fig. 7: Our implementation of cross-correlation matrix for interrogation area 2610 between images 1 and 2 of the Alaska dataset [2]. Magnitude: 10.273 (PIXELS/s = 1.54m/s) Direction: (-1, 0.1007). Note the change in sign, the MATLAB output is reoriented 180° such that North is up, opposite of our implementation

G. Application Optimization Parameters

From the two measurement methods described above to reduce the computation time (and energy consumption) of the PIV algorithm, we end up with a total of seven optimization parameters: one for the ratio of measurements between the two types, four from the graded PIV approach to retrieve a velocity field, and two for the discharge measurement approach. This is visualized by Figure 10. The optimization for rectangular interrogation areas are omitted here as they should be applied after the optimizations listed in Figure 10 are tuned to the specific application.

VI. Application Energy Usage

We have directly tested the power consumption of running PIV on the CommonSense boards through PowerDue as shown

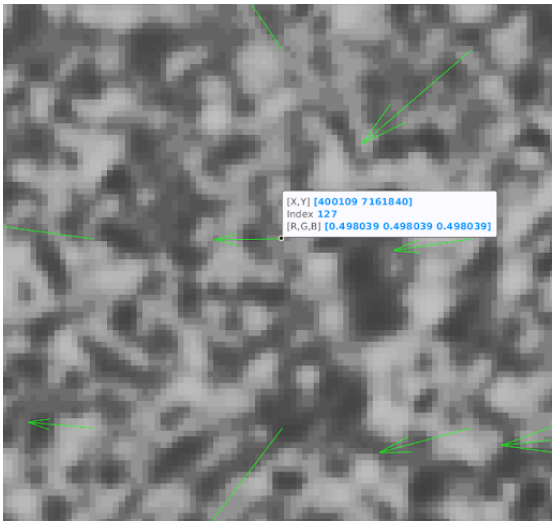


Fig. 8: PIVLab displacement vector plotted on the interrogation area from image 1 of the Alaska dataset [2] Magnitude 1.6834(m/s) Direction: (-1, -0.0108).

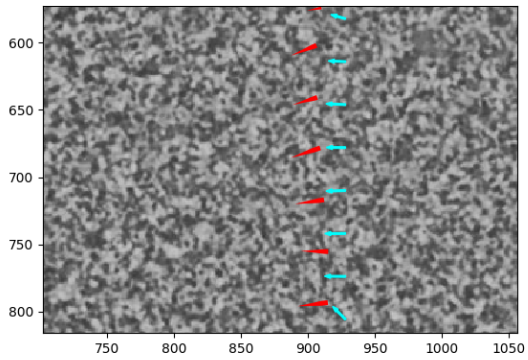


Fig. 9: While our implementation matches that of the PIVLab implementation, we include a comparison between our result (cyan) and the ground-truth values (red) measured via ADCP.

in Figure 11. The current consumption is roughly 50 mA and takes 200 ms to execute a PIV analysis between a pair of 64x64 pixel sized images. A power analysis with the device using 1580x64 image sizes sampled 8 times per hour (12 hours a day) yields a rating of 0.53 years for a 12,000 mAh battery. The main contributor to power loss stems from the fact that a single comparison of the algorithm takes 200 ms, and the algorithm is repeatedly run for ensembling effects. With these parameters, we run the PIV algorithm 224 times, which leads to the device being on for almost 10% of the hour. Leaving the processor on contributes to 99.6% of the device's power consumption, as seen in Figure 12. In comparison, the LoRa radio takes less than 1% of the total energy from the battery. The power analysis does not take into account camera usage from sampling.

To remedy our low lifespan problem, the device can use environment-specific knowledge to swap between modes to

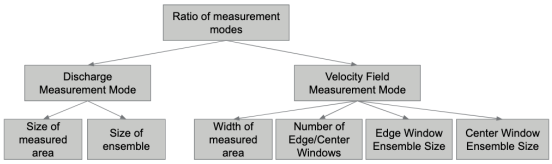


Fig. 10: Break down of the optimization parameters by measurement type. In this figure, "windows" are referring to interrogation areas.

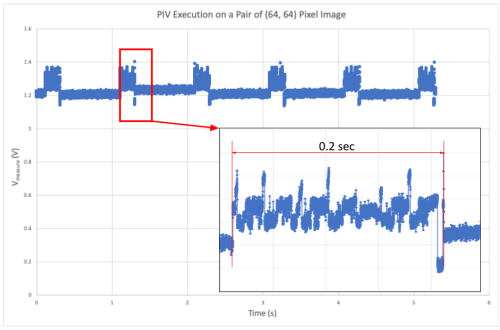


Fig. 11: Power measurement of running PIV on a pair of 64x64 pixel sized images.

optimize usage of energy. The contributing factor for power consumption is the time spent by the processor being awake. As the processor speed can only be marginally changed, power consumption relies on reducing the number of PIV iterations required for analysis. The optimization modes described take advantage of using less interrogation areas as a trade off between power consumption and the velocity result accuracy. Knowing the physical attributes of the system allows for the device to swap between optimization modes. Once deployed, the device can develop a history of predicted values over time. Overall, large streams do not change rapidly in direction and magnitude over an 8 minute time period. Thus, the device can use a smaller number of ensembles, interrogation area, and interrogation area sizes to calculate the velocity vectors. The discharge measurement mode takes advantage of this and utilizes stream depth to calculate the discharge rate of the stream. If the device detects major changes within the stream velocity, it can revert back to a velocity measurement mode

			% ON hour	mAh
Processor	Computing (mA)	50.808	10.07%	5.1181642
Contribution to Power	standby (uA)	18	0.00%	0
99.55%	hibernate(uA)	4	89.93%	0.00359
Radio	Receive Mode (mA)	10.8	0.01%	0.0012
Contribution to Power	Transmit Mode (mA)	29	0.08%	0.0218080
0.45%	Sleep Mode (uA)	0.2	99.91%	0.0001998
			Total (mAh)	5.1449620
			Years of Life	0.5325066

Fig. 12: Holistic power analysis of running PIV on a pair of 64x64 sampled 8 times/hour.

in which it computes PIV over a larger portion of the image. The optimization parameters that we can vary in Figure 10 will depend on the surrounding environment the device is installed.

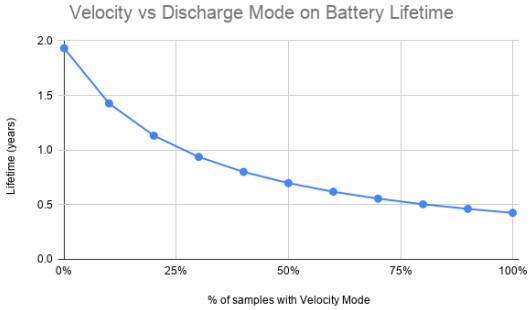


Fig. 13: 10 samples/hour on image size 1580x64. Varying between discharge and velocity mode significantly increases battery life by 4x.

Varying the device mode can greatly extend the lifetime of the device. Figure 13 describes a scenario where the device sends 10 samples per hour for a 1580x64 pixel sized image, we have selected the velocity mode to take 8 and 4 center and edge interrogation areas respectively with ensemble size 8 and 12 respectively. The discharge mode takes a 128x128 image with ensemble size 8 with a step size of 1 interrogation area. We then varied the amount of samples taken for each mode per hour. As expected, we get a greater lifespan of the battery when we use less interrogation areas to compute the velocity vectors. The lifespan of the device almost quintuples by reducing the number of interrogation areas from 224 to 48.

For ensemble sizes, 16 ensembles would reduce noise by 90% [7]. The ensemble sizes chosen were design parameters that we varied to balance time spent vs. accuracy lost. We determined that 8 and 4 are good ensemble sizes for the Alaskan data set as center interrogation areas have less noise and variance compared to edge windows. We used a 128x128 pixel image size for the discharge mode as the Alaska data we used to test had a 15cm/pixel conversion rate giving us a 19.2x19.2 m^2 interrogation area [2]. This conversion rate is larger than most bridge-mounted scenarios as these Alaskan images were taken aerially from a helicopter. The pixel/m conversion would likely decrease in a bridge-mounted case where the camera is much closer to the water. The number of interrogation areas needed for good velocity measurements would need to increase to account for smaller camera FOVs.

VII. Reflection and Future Work

Through this project, we've learned that in real-life IoT applications, space complexity (which is usually ignored in modern programming) is a pressing issue. A 64x64 interrogation area size is the bare minimum to achieve good results and is the limit we can put on the CommonSense board. Time complexity is also a concern, but those are often immediately identified. We designed trade offs solutions in both spatial and

temporal domains: using less interrogation areas and having more hardware support.

The CommonSense architecture we had planned to use for the PIV algorithm has both limitations in speed and space. Ideally the hardware running the PIV algorithm have more RAM or utilize hardware components that are dedicated for FFT operations. In terms of space, the CommonSense board has a limit of 250MB of RAM. We could not compute the FFT for interrogation areas of 128x128 pixels. 64x64 pixels is the minimum for the PIV algorithm's accuracy to be acceptable, so it would be desirable to run 128x128 sized FFTs. Padding the 64x64 for accurate FFTs is also impossible as it requires the dimensions of the interrogation area to double. Computation time correlates to power spent. Our power analysis indicates that the processor is the bottleneck of the power consumption, responsible for 99.6% of the power consumption. Having a faster computation for the cross correlation will save us more energy. As an FPGA can easily finish computation much faster than our CPU can, and we believe that the startup delay and energy of running an FPGA is much smaller than the time spent computing on the CPU. By Amdahl's Law, we can do little to improve the system through other means either than reducing the amount of time the processor spends on computing the PIV algorithm.

For future iterations of the project, the PIV algorithm would be mutated to account for more dynamic settings. The PIV algorithm is dependent on the quality of the image capture, and it requires ample sunlight for good images. This limits capturing optical images to daytime. One could circumvent this by using an infrared camera, but using infrared images tend to have a lower image quality than optical images. More preprocessing would be necessary to achieve good results on infrared images. Ideally, this algorithm would run atop of a drone hovering over a stream, so many steps of preprocessing would be required to achieve this. The incoming images would need stabilization and geological information of the location/angle of the camera to accurately calculate the velocity vectors of the stream. The CommonSense architecture also requires drivers to be written for the QSPI interface with its flash memory.

ACKNOWLEDGMENT

We would like to thank Carl Legleiter, Paul Kinzel, and Mathieu Marineau from the USGS for this opportunity and help. We would also like to thank Bob Iannucci, Eve Hu, and Reese Grimsley for their help and support on the CommonSense platform.

REFERENCES

[1] P. Kinzel and C. Legleiter, "suas-based remote sensing of river discharge using thermal particle image velocimetry and bathymetric lidar," *Remote Sensing*, vol. 11, p. 2317, 10 2019.

[2] C. Legleiter and P. Kinzel, "Inferring surface flow velocities in sediment-laden alaskan rivers from optical image sequences acquired from a helicopter," *Remote Sensing*, vol. 12, p. 1282, 04 2020.

[3] M. Detert and V. Weitbrecht, "A low-cost airborne velocimetry system: Proof of concept," *Journal of Hydraulic Research*, vol. 53, 08 2015.

[4] W. Thielicke and E. J. Stamhuis, "Pivlab – towards user-friendly, affordable and accurate digital particle image velocimetry in matlab," *Journal of Open Research Software*, vol. 2, 10 2014.

[5] W. Thielicke, "The flapping flight of birds: Analysis and application," Ph.D. dissertation, University of Groningen, 2014.

[6] D. Dabiri, "Cross-correlation digital particle image velocimetry – a review," 2007.

[7] E. Delnoij, J. Westerweel, N. Deen, J. Kuipers, and W. van Swaaij, "Ensemble correlation piv applied to bubble plumes rising in a bubble column," *Chemical Engineering Science*, vol. 54, no. 21, pp. 5159 – 5171, 1999. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000925099900233X

[8] J. Santiago, S. Wereley, C. Meinhart, D. Beebe, and R. Adrian, "A particle image velocimetry system for microfluidics. exp fluid," *Experiments in Fluids*, vol. 25, pp. 316–319, 09 1998.

[9] C. Willert, "Adaptive piv processing based on ensemble correlation," 07 2008.

[10] B. Triggs, "Optimal filters for subpixel interpolation and matching."

APPENDIX

Planned schedule and actual end date of our project is as follows:

ID	task	plan	actual
1	Use MATLAB Autocoder to embedded C	11/4	11/4
1.1	Evaluate task 1	11/16	11/16
2	Reimplement MATLAB code in embedded C	11/4	11/6
2.1	Complexity analysis of algorithms	11/9	11/9
3	Evaluate performance and energy differences between Autocoded and embedded code	11/16	11/16
4	Optimize V1 algorithm complexity	11/23	11/23
5	Analyze algorithm accuracy with 1D cross correlation and non-uniform windows	-	11/24
6	Analyze algorithm accuracy and SNR performanc ewith(out) multipass DFT	-	11/24
7	Analyze Tradeoff the direct vs Fourier cross correlation	-	11/24
8	Evaluate performance and energy of optimized algorithm on CommonSense	12/7	11/28
9	Rework V1 algorithm accuracy and/or performance into V2	12/4	12/4
10	Analyze V2 algorithm accuracy, performance, and energy	12/11	12/11