Feedback Controls in Droplet Microfluidics

by

Yuk Hei Wong

A thesis
presented to the University Of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, 2016

Author's Declaration

Abstract

The ability to stabilize and move individual droplets would allow scientists to perform micro-scale manipulation, and unlock the advantages of microfluidics. The challenge lies in the fact that droplet displacements are unstable, and that the system is multi-input-multi-output in nature. This dissertation begins with construction of a state-space model to represents fluid dynamics in a channel network. The model is validated with experimental data, and used to design LQR controllers. The controllers utilize feedback provided from computer vision, to actuate electro-pneumatic transducers appropriately in order to stabilize and control droplet movements. A significant portion of this report is dedicated to describing a custom computer program that was created for implementing the controllers. The program enables users to manipulate droplets in real time by interacting with an augmented video stream. A demonstration is provided in which droplets are generated, stored, merged and split repeatedly on-demand.

Acknowledgement

I would like to sincerely thank Professor Carolyn Ren for her supervision and support during my self-discovery journey in the last two years. I want to thank her for her tolerance, and especially the encouragement she gave me during my moments of doubt. I would also like to thank Professor Kaan Erkorkmaz for his teaching, his intuitive and insightful explanation, and his personal advice.

In addition, I would like to thank Professor Jan Huissoon for taking the time to read this thesis, and for joining my seminar committee.

For Mom and Dad,
and Vivien

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Vision without action is merely a dream.
Action without vision just passes the time.
Vision with action can change the world.

Joel A. Barker

Microfluidics is the study of fluid behaviour in micro scale. Fluid flow in micro scale has several unique properties. First and foremost, due to small Reynolds numbers, fluid flow is laminar and can be analysed and controlled with relative ease. On the other hand, due to large surface-to-volume ratios, thermal diffusion is rapid, allowing sample temperature to be uniformly adjusted without delay. In addition, electrokinetics phenomena induced by surface charges is prominent. Combining electrophoresis, thermal diffusion and electrokinetics leads to an optimum tool for performing numerous biochemical analysis such as PCR amplification, and the subsequent focusing and detection of DNA fragments [9].

Another advantage of microfluidics is the small sample volume involved. Typical microfluidics experiments consumes only nano-litres of chemical reagents. Compare with an equivalent macro-scale experiment, significant cost and sample preparation time is saved. Small sample sizes also allow reagents to be dispensed with high concentration accuracy, and results in precisive control over chemical reactions and synthesis [10].

One of the key research areas in microfluidics is lab-on-a-chip devices. This research field attempts to perform laboratory procedures and experiments within a micro scale device, in order to benefit from the use of microfluidics. Successful lab-on-a-chip applications range from isolating plasma from whole blood samples [11], to measuring reaction kinetics with nuclear magnetic resonance [12], to cell detection, sorting, and enrichment [13].

A recent development in lab-on-a-chip is droplet microfluidics, where fluid of two or more immiscible phases co-exist within the same device. Surface tension dictates that one of the phases will form discrete droplets, surrounded by the other continuous phase. Each droplets act as individual compartment in which chemical reactions can take place. Unwanted mass diffusion and contamination between samples are eliminated. Droplet microfluidics has proven to be particularly useful for bioassay [14] [15], drug screening [16], or acting as versatile chemical reactors [17].

In most applications, droplets are generated passively by maintaining a constant ratio between flow rates of the continuous phase and the discrete phase. This technique generates a sequence of droplets, whose length is dependent on the spacing between itself and other droplets. By designing intricate channel geometries, a train of droplets can be split; two trains of droplets can be merged. However, there is no mechanism yet to manipulate each individual droplets. Furthermore, the droplet generation, merging, and splitting geometries have narrow ranges of operation. Small changes in channel dimensions due to manufacturing defects will likely render them non-functional. For the same reason, combining several geometries to expand functionality is challenging as geometries will interfere with upstream and downstream components, resulting in their stable operating range being exceeded

An ideal droplet microfluidic device would be one that 1) can function despite of high manufacturing tolerances, and is insusceptible to environmental disturbance; 2) allows multiple geometries to be connected and perform without self interference; 3) can be designed following a general approach instead of requiring specialized knowledge; 4) eliminates the need of sample preparation.

The first three aspired features can be achieved by introducing feedback controls into chip operations. First, the robustness of a feedback systems would overcome effects of minor manufacturing defects. Meanwhile, its disturbance rejection ability counteracts environmental influence, compensating to guarantee performance. Furthermore, a feedback controller can apprehend the coupling effects when connecting multiple geometries, such that each of them remain functional under feedback controls. Better still, once a control law is formulated, it can be applied to other chips.

Although feedback is essential in controls theory, introducing sensors into microfluidic devices is challenging. Electrodes can be embedded into chips to provide sensing [18], but only at the expense of increased chip complexity, extra source of error, and limited design freedom. This in effect defeats the benefits of feedback.

Fortunately, droplets have visible boundaries, and they provide a unique opportunity for exploitation. Using computer vision, fluid flow rates that would otherwise be invisible can be observed through droplet movements. In control terms, the loop can be closed by using vision information as feedback.

Another critical component for feedback controls is actuation, which defines a mean for controllers to take action upon the system. Common actuators used in lab-on-a-chip experiment are the pressure pumps and the syringe pumps. They are external equipments that connect to chips to deliver precise flow rates or pressures at the chip inlets. To prevent interference between different parts within a network, pneumatic valves [19] can be fabricated onto chips. These internal actuators allow manipulation of individual fluid volumes. Yet again, it comes at the expense of chip vulnerability and the ease of design and manufacturing.

A feedback controller would embrace instead of trying to eliminate interference between different parts of a chip. It does so by capturing the coupling effects in a mathematical model, and calculating appropriate external actuations that will compensate for internal interferences. In this fashion, de-coupling can be achieved without the need of on-chip pneumatic valves.

Henceforth, the subject of this thesis is on applying controls theory in droplet microfluidics. The introduction of controls theory leads to an unexpected advantage. Since the controller can command fluid to flow at arbitrary rates at any given moment, processes that were previously unstable is now possible. This implies that droplet lengths can be adjusted independent of droplet spacing, and individual droplets can be moved, merged, split, and sorted at any time according to user's request.

The proposed method is rather new in the field of lab-on-a-chip. In the initial phrase, the proposed method will be slow compare to prevalent techniques in droplet microfluidics. Compare with existing open-loop techniques which can generate tens to hundreds of droplets in a second, the proposed closed-loop method would take seconds to generate one droplet.

As the proposed method matures, improvements in hardware and controller should lead to speed recovery. However, regardless of speed, being able to manipulate each individual droplets independently should bring benefits that far out-weight the through-put deficiency.

It is important to note that the proposed method is entirely software-based. The core enabling component is a computer program, whereas the required hardware consist simply of a camera and a pressure pump, both are prerequisites for operating lab-on-a-chip devices, and are commonly found in microfluidics laboratories. No additional exotic equipment nor exclusive chip designs nor special sample preparation is needed.

This thesis is laid out as follow. Prior art of controls theory being applied to microfluidics is summarized in Chapter 2. The methodology and hardware set up is mentioned in Chapter 3. Chapter 4 develops a general modelling approach to describe fluid dynamics in a channel network. Chapter 5 validates the model with experimental data. Chapter 6 describes the design of feedback controllers, while Chapter 7 contains details regarding the computer program used in implementation. Chapter 8 demonstrates the results, while recommendations for future research is noted in Chapter 9.

Chapter 2

Literature Review

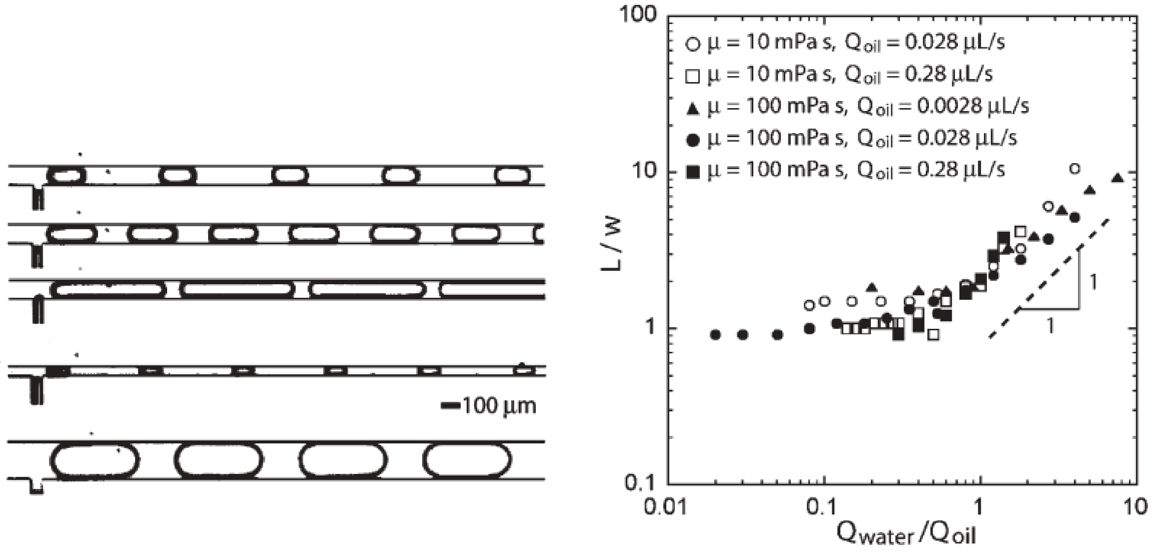


Figure 2.1: Garstecki droplet formation[1]

The prevalent droplet formation technique[1] is shown in figure 2.1. Constant flow rates of the dispersed and continuous phases are supplied to generate a train of droplets. The resultant droplet sizes and spacings depend on the steady state flow rate ratios, fluid properties, and channel dimensions. Non dimensional empirical relations were extracted from experimental data, allowing researchers to target a certain droplet size, and design microfludic chips accordingly.

To merge two in-coming droplets, special geometries are fabricated into chips, such as the pilers [2] shown in figure 2.2. The first arriving droplet is shorter than the piler array. This traps the droplet temporarily by allowing the continuous phase to escape between pilers and reducing pressure behind the droplet. The second droplet would arrive and merge with the first. The extended droplet would then block all the pilers, raising back pressure to push the droplet away.

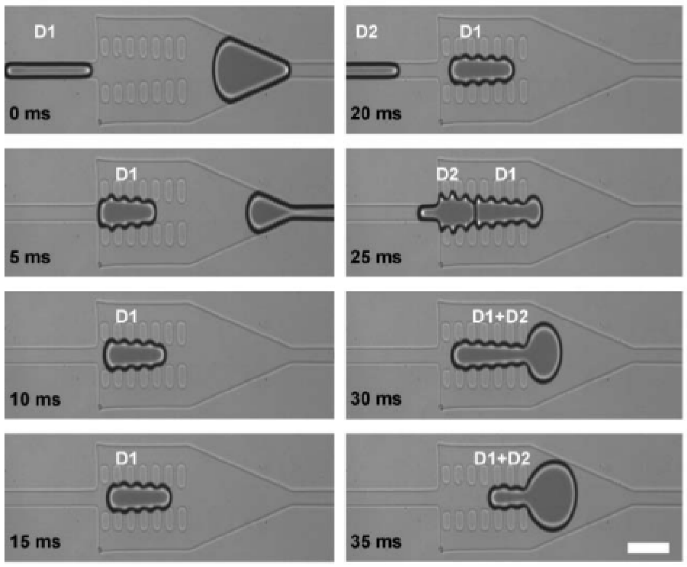


Figure 2.2: Niu et al. droplet merging[2]

Droplets might be stored to carry out prolonged chemical reactions and observations. Huebner[3] had designed traps for capturing droplets as shown in figure 2.3. The droplets could be released by reversing the flow inside the microfluidic chip.

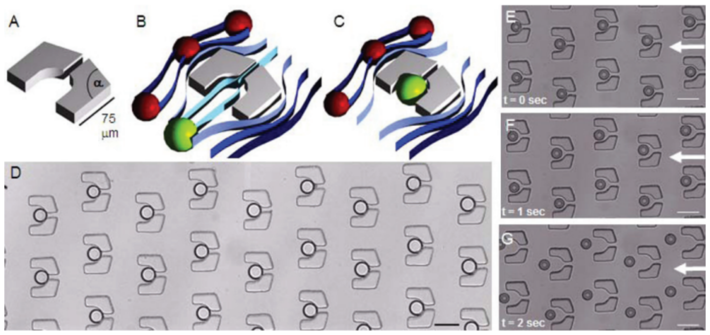


Figure 2.3: Huebner et al. droplet trapping[3]

2.1 Controls in Microfluidics

In this section, previous work that involves microfluidics and controls theory is reviewed.

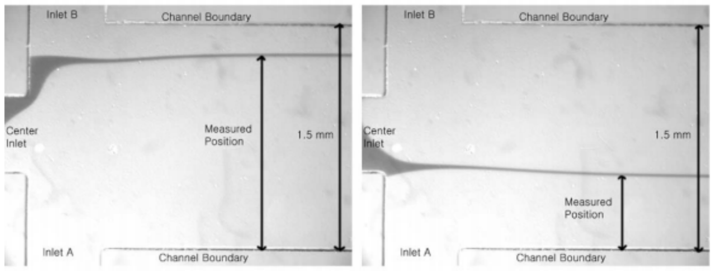


Figure 2.4: Kim et al. interface control[4]

Kuczenski[20] and Kim[4] had developed a mechanical device for actuating inlet pressures in a two-phase flow device. Using feedback provided by pressure sensors, they were able to accurately manipulate the interface location of a focused fluid stream, as shown in figure 2.4. Their focus is on regulating the laminar interface in a specific geometry, instead of droplets in an arbitrary channel network.

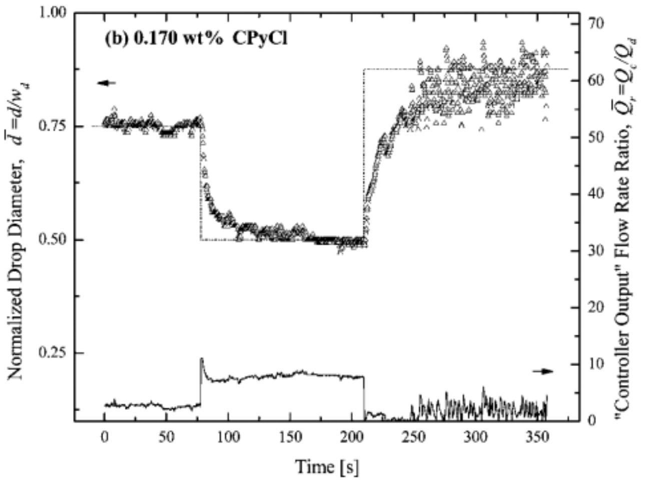


Figure 2.5: Miller et al. controlled generation[5]

In Miller's work[5], droplet generation is regulated using a PI controller. They measured droplet diameters with image processing, and controled flow rate ratios by actuating syringe pumps. Drolets were generated within stable regimes, and they managed relatively slow closed-loop response as shown in figure 2.5, in which a desired droplet size was achieved in $20 - 50s$ depending on fluid properties. Zeng[21] has proposed similar techniques, but their study only provided simulation results and lacked experimental validation. Even though these studies employed feedback controllers, they were concerned with continuous droplet generation, and did not attempt to manipulate individual droplet movements.

By embedding electrodes into the microfluidic chip, Niu[6] was able to sense droplet presences through capacitive measurements. In addition, they were able to deflect droplets into a targeted channel by differentiating dielectric properties of different reagents. This method only works for a specific set of fluids and channel geometry, and requires embedded electrods and custom designed electronics.

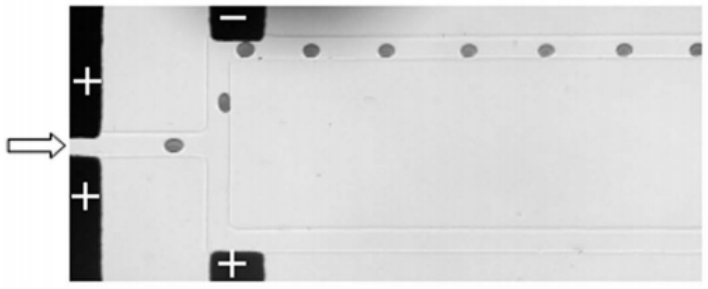


Figure 2.6: Niu et al. electrode sensing[6]

Figure 2.7 shows a particle steering system invented by Armani[7]. The microfluidic chip contained multiple electrodes for inducing 2-dimensional electrokinetic flow. Image processing was used to measure instantaneous position of the particle. Electrode voltages were actuated to achieve pre-calculated electric field profiles, in order to steer the particle.

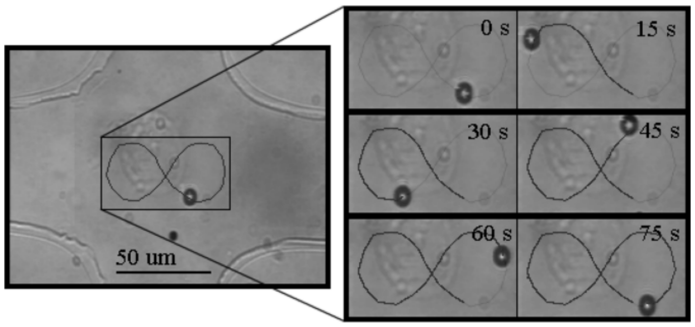


Figure 2.7: Armani et al. particle steering[7]

Separately, Shenoy[22] was able to steer particle in a similar manner, but by actuating chip inlet pressures, thereby eliminating the need of electrodes. A model predictive controller was employed to handle the non linear fluid dynamics.

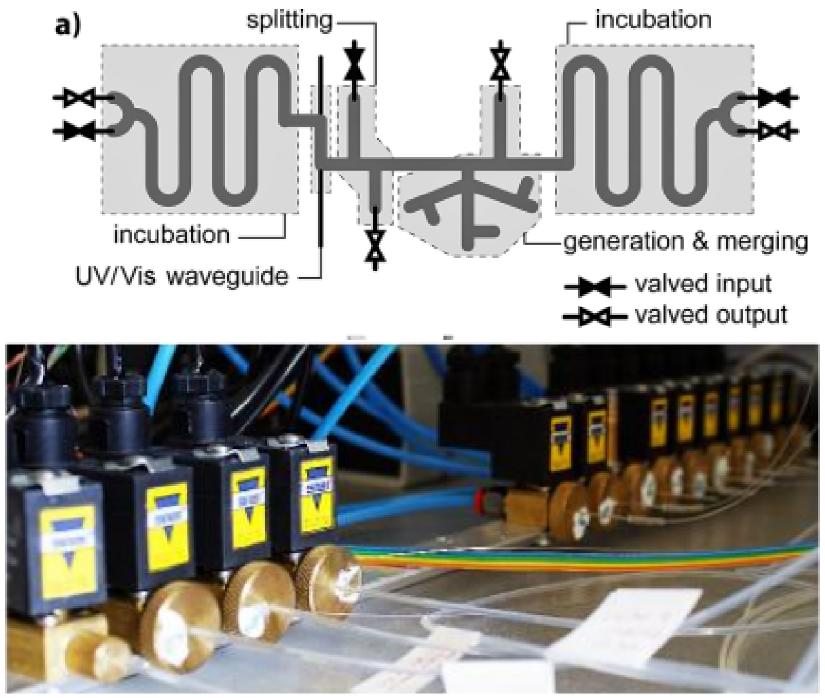


Figure 2.8: Garstecki's chemostat[8]

Piotr Garstecki, a leading researcher in droplet microfluidics, has recently demonstrated the potential of automated droplet-based system. In his work[8], droplet generation, splitting, merging, and sorting were performed all in a single chip, creating micro-chemostats for cultivating and studying bacterial growth. The device is shown in figure 2.8, and its operation relies on a series of high precision on-off valves. Little information is provided regarding the control principles. It is possible that droplet presences were detected in image processing, and a group of pre-calculated valve actions were executed in a timed sequence. This technique is effective and straight forward, but might be difficult to expand to accommodate larger channel networks.

Chapter 3

Hardware and Methodology

In this chapter, I will describe the equipment used in the project, and provide instructions on setting up hardware and fabricating devices, such that experiment can be replicated. In figure 3.1, all the controls related hardware are shown. The flow of information starts at the microfluidic chip, where droplet movements take place. A scientific camera captures the movements and provide video live-stream to the personal computer(PC). In the PC, image processing and control calculations are performed in real time using a custom program, which will be described in chapter 7. The PC then command the pressure pump to supply varying inlet pressures to the microfluidic chip, affecting droplet movements, and completing the information loop.

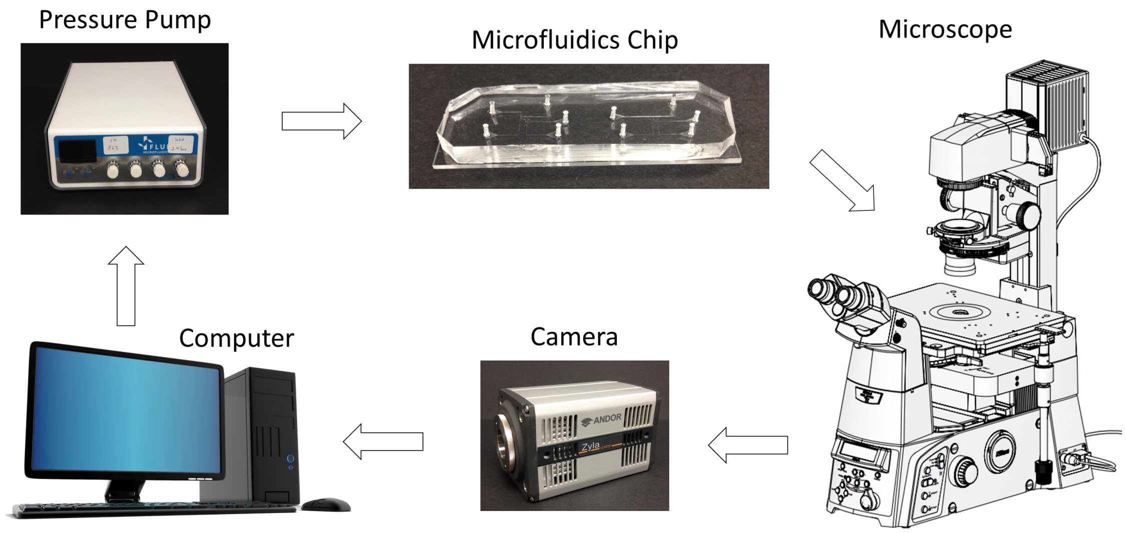


Figure 3.1: Hardware used in this project, arrow represents the flow of information. Microscope drawing obtained from user manual [23]

3.1 Microfluidic Chip

The microfluidic chip is constructed with standard soft-lithography techniques [24]. First, channel networks are drawn in AutoCAD and printed onto transparent photomasks. Then, a silicon wafer is prepared with positive photoresist (MicroChem Su-8 2000). The photoresist is selectively cured by being exposed to UV through the photomasks. By developing the photoresist, an extrusion of the channel network is formed on the silicon wafer. Afterwards, PDMS (Dow Corning sylgard 184) is pour onto the silicon wafer, and peeled off after cure, with the photoresist extrusions now creating cavities in the PDMS. The PDMS is then plasma bonded with glass slides to form a sealed channel network. Lastly, to recover the hydrophobocity lost during plasma treatment, the chip is heated to $195C$ for 48 hours prior to use.

A typical channel network design is shown in figure 3.2b. The center cross ($500um$ end-to-end) is used for image processing calibration. Lengths of each channels are patterned onto the PDMS for easy record tracking. In addition, a drain channel is added to assist initial bleeding, but is plugged during experiment.

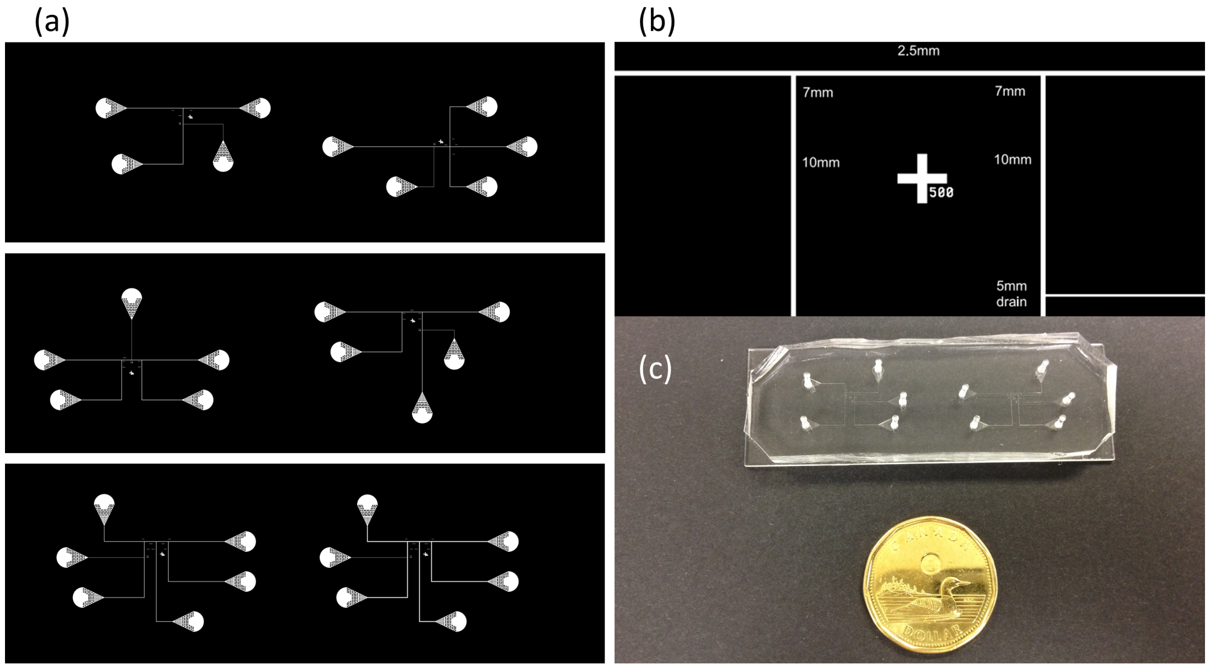


Figure 3.2: (a) Photo masks for various channel network designs. (b) Closed-up view of a 4 inlets channel network. (c) Fabricated microfludic chip.

3.2 Optics and Camera

An Andor Zyla 5.5 scientific camera and a Nikon Eclipse Ti-E microscope are used in this project. Neither of them are designed for computer vision applications, and were chosen simply because they were available in the laboratory. Nevertheless, I have carefully reviewed their specifications to ensure that their performance are sufficient.

The Zyla camera shown in figure 3.3 has a $16.6mm \times 14.0mm$ CMOS sensor. It is designed for capturing high quality scientific images, hence is equipped with exotic features such as Peltier cooling, 16 bit depth, global shutter, and a maximum quantum efficiency of 60%. Although the large sensor dimensions result in very high image quality, it necessitates the use of high quality microscope objectives and a uniform light source. The need of a microscope in turn prevents the control system from becoming portable, and will potentially interfere with biochemical procedures that require different magnifications and illuminations. In chapter 9, I will recommend replacing the current set up with much cheaper cameras and lenses that are used in smart phones and mobile devices.

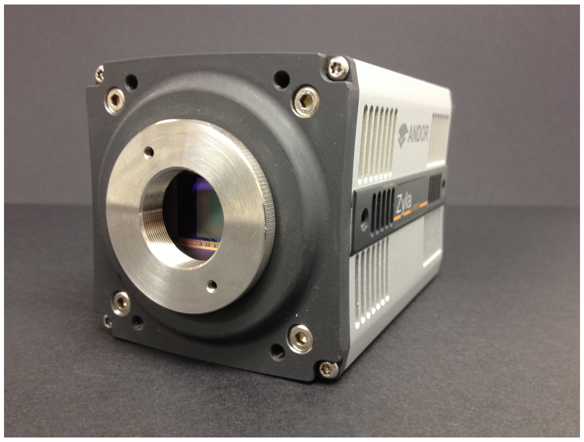


Figure 3.3: Scientific CMOS camera

The camera connects to PC through a USB 3.0 connection, and has a purchasable software development kit (SDK). Although it can live-stream full frame resolution (2560×2160) videos at 40 frame-per-second (fps), processing that many images, each 5.5 megapixel with a $16bit$ depth, is beyond the capability of consumer grade PC. Therefore, images are binned to reduce resolution and processed at lower rates. Typical camera settings are listed in table .

Settings	Value	Settings	Value
Overlap	false	ElectronicShutteringMode	Rolling
StaticBlemishCorrection	true	PixelEncoding	Mono16
SensorCooling	true	TargetSensorTemperature	0
FullAOIControl	true	AOIBinning	2x2
ExposureTime	0.004	FrameRate	40
TriggerMode	Internal	CycleMode	Continuous
FrameCount	1	AccumulateCount	1

Table 3.1: Camera settings

3.3 Pressure Pump

In this project, the Fluigent MFCS series pressure pumps are used. Figure 3.4a shows the pump system in experiment configuration. The pump receives $35psi$ pressurized air from an external compressors, and regulates pressure of each outlets within the range of $0 \rightarrow 2Bar$ at a $0.3mBar$ resolution. Pump outlets are connected through Tygon tubings to the reservoirs, in which chemical solutions reside. When pressurized, solutions flow through PFA tubings into the chip.

The pressure pump connects to PC through a USB 2.0 connection. Due to a proprietary communication protocol, command speed is limited to 10 samples per second per outlet. Because of this, the sampling frequency of the entire control and image processing system is set to $10Hz$.

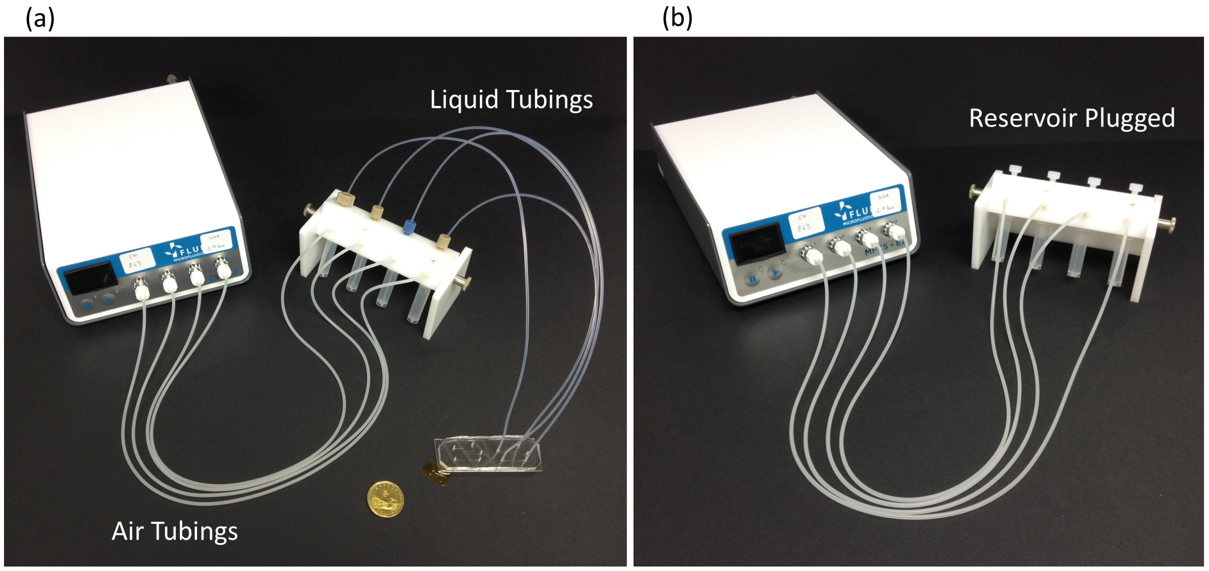


Figure 3.4: (a) Experiment set up showing pressure pump, air tubings, reservoirs, liquid tubings, and microfluidic chip. (b) System identification set up showing reservoir being plugged

Although the pressure pump technology is proprietary, I reasoned that its underlying mechanism is an array of electro-pneumatic (E/P) transducers. A typical E/P transducer is comprised of an embedded micro-controller and a integral volume booster. The micro-controller regulates airflow into and out of the integral volume booster base on feedback from an integrated pressure sensor. The integral volume booster is a flexible chamber, whose shape controls the main valves to which the outlet port is connected to the inlet and exhaust ports.

Behaviour of the integral volume booster and computation sequences in the micro-controller give rise to delays and overshoots that characterize a certain E/P transducer. In the rest of the chapter, I will approximate these behaviour using system identification techniques. For the system identification experiments, the pump is set up as shown in figure 3.4b. In this configuration, the reservoirs are plugged, and the pump model will capture pump dynamics as well as effects due to compliance of the air tubings and the reservoir.

3.3.1 System Identification in Time Domain

A linear time-invariant (LTI) system is represented by an ordinary differential equation (ODE). Given input $u(t)$ the system will output $v(t)$, and the system's characteristics are determined by constant coefficients a and b.

$$v^{(n)} + a_{n-1}v^{(n-1)} + \cdots + a_1 v + a_0 = b_m u^m + b_{(m-1)}u^{(m-1)} + \cdots + b_1 u + b_0 \tag{3.1}$$

Using the derivative property of Laplace transforms, the ODE is converted into a transfer function.

$$\mathcal{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}dt \tag{3.2}$$

$$\mathcal{L}\{\frac{df}{dt}\} = s\mathcal{L}\{f(t)\} \qquad when f(0) = 0 \tag{3.3}$$

$$H(s) = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} = \frac{V(s)}{U(s)} \tag{3.4}$$

The convolution theorem states that output in the Laplace domain is simply multiplication of input with the system transfer function $V(s) = H(s)U(s)$. In addition, roots of the nominator polynomial (zeros) and roots of the denominator polynomial (poles) provide insights into the system transient response. Intuitive understanding can be drawn from values of poles and zeros without having to solve the ODE.

Since I can only command and sample the pressure pump at $10Hz$, it is more accurate to describe the system with a difference equation instead of an ODE.

$$v_k = \alpha_1 v_{k-1} + \cdots + \alpha_n v_{k-n} + \beta_0 u_k + \beta_1 u_{k-1} + \cdots + \beta_m u_{k-m} \tag{3.5}$$

A discrete transfer function is obtained from the difference equation using the shifting property of the Z transform.

$$\mathcal{Z}\{f_k\} = \sum_{k=0}^{\infty} f_k z^{-k} \tag{3.6}$$

$$\mathcal{Z}\{f_{k-1}\} = z^{-1}\mathcal{Z}\{f_k\} \tag{3.7}$$

$$H(z) = \frac{\beta_0 + \beta_1 z^{-1} + \cdots + \beta_m z^m}{1 - \alpha_1 z^{-1} - \cdots - \alpha_n z^{-n|}} = \frac{V(z)}{U(z)} \tag{3.8}$$

To identify the characterizing coefficients α and β, equation 3.5 is re-written in matrix form, where N is the total number of data points, and e is the error between model and measurement.

$$\overbrace{\begin{bmatrix} v_N \\ v_{N-1} \\ \vdots \\ v_n \end{bmatrix}}^{V} = \overbrace{\begin{bmatrix} v_{N-1} & \dots & v_{N-n} & u_N & u_{N-1} & \dots & u_{N-m} \\ v_{N-2} & \dots & v_{N-n-1} & u_{N-1} & u_{N-2} & \dots & u_{N-m-1} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ v_{n-1} & \dots & v_0 & u_n & u_{n-1} & \dots & u_{n-m} \end{bmatrix}}^{\Psi} \overbrace{\begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \\ \beta_0 \\ \beta_1 \\ \vdots \\ \beta_m \end{bmatrix}}^{\Theta} + \overbrace{\begin{bmatrix} e_N \\ e_{N-1} \\ \vdots \\ e_n \end{bmatrix}}^{E} \tag{3.9}$$

Experiments were performed to obtain an input time series and an output time series, allowing vector V and regression matrix Ψ to be constructed. Provided that $\Psi^T \Psi$ is invertible, the best fit Θ was obtained by minimizing E

$$\Theta = (\Psi^T \Psi)^{-1} \Psi^T V \tag{3.10}$$

Using the above method, pump models of different orders $m = n = 1, 2, \dots, 10$ were fitted. Figure 3.5 shows a comparison between the model predictions and experiment data. It is obvious that low order models did not capture pressure pump dynamics very well.

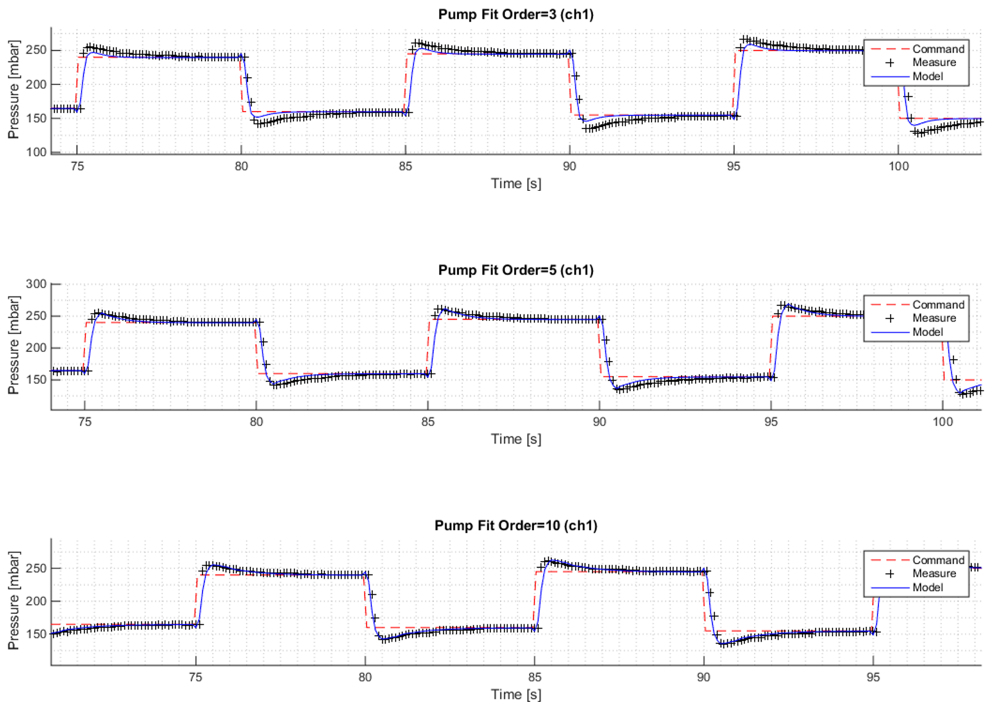
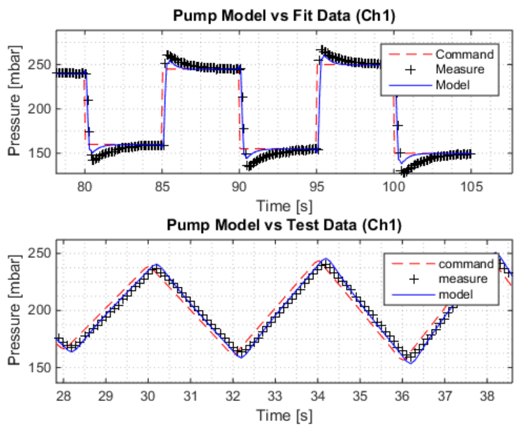
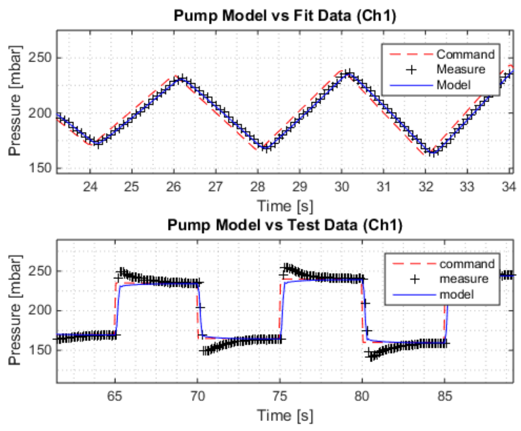


Figure 3.5: Pump time domain fit (3rd, 5th, and 10th order)

Furthermore, models obtained from one response type performed poorly when predicting a different response type. In figure 3.6a, a model that was fitted from step response yielded significant error when predicting ramp responses. On the contrary, figure 3.6b shows a model being fitted from ramp response having problems predicting step responses.



(a) Step fit vs ramp data



(b) ramp fit vs step data

Figure 3.6: Pump fit vs. test data

These results suggested that pressure pump dynamics are non-linear. Instead of spending time on modelling the non-linearity, which would certainly complicates later modelling and the following controller designs, I decided to approximate the pressure pump dynamics in frequency domain instead, and move on to more important issues.

3.3.2 System Identification in Frequency domain

The convolution theorem states that given a LTI system, the corresponding output is convolution between input $u(t)$ and the system's impulse response $h(t)$.

$$v(t) = \int_{-\infty}^{\infty} u(\tau)h(t-\tau)d\tau = \int_{-\infty}^{\infty} u(t-\tau)h(\tau)d\tau \tag{3.11}$$

In the event when input is an exponent $u^*(t) = e^{j\omega t}$, the time domain output $v(t)$ becomes simply the multiplication of $u^*(t)$ and system transfer function $H(j\omega t)$

$$v(t) = \int_{0}^{\infty} e^{j\omega(t-\tau)}h(\tau)d\tau \tag{3.12}$$

$$= e^{j\omega t}\int_{0}^{\infty} h(\tau)e^{j\omega\tau}d\tau \tag{3.13}$$

$$= u^*(t)H(j\omega) \tag{3.14}$$

This implies that a sinusoidal input will result in a sinusoidal output with equal frequency. Furthermore, magnitude and phase shift of the output signal are each functions of frequency, and that function is the system transfer function.

$$u(t) = \cos(\omega t) = \frac{e^{j\omega t} + e^{-j\omega t}}{2} \tag{3.15}$$

$$v(t) = |H(j\omega)|\cos[\omega t + \angle H(j\omega)] \tag{3.16}$$

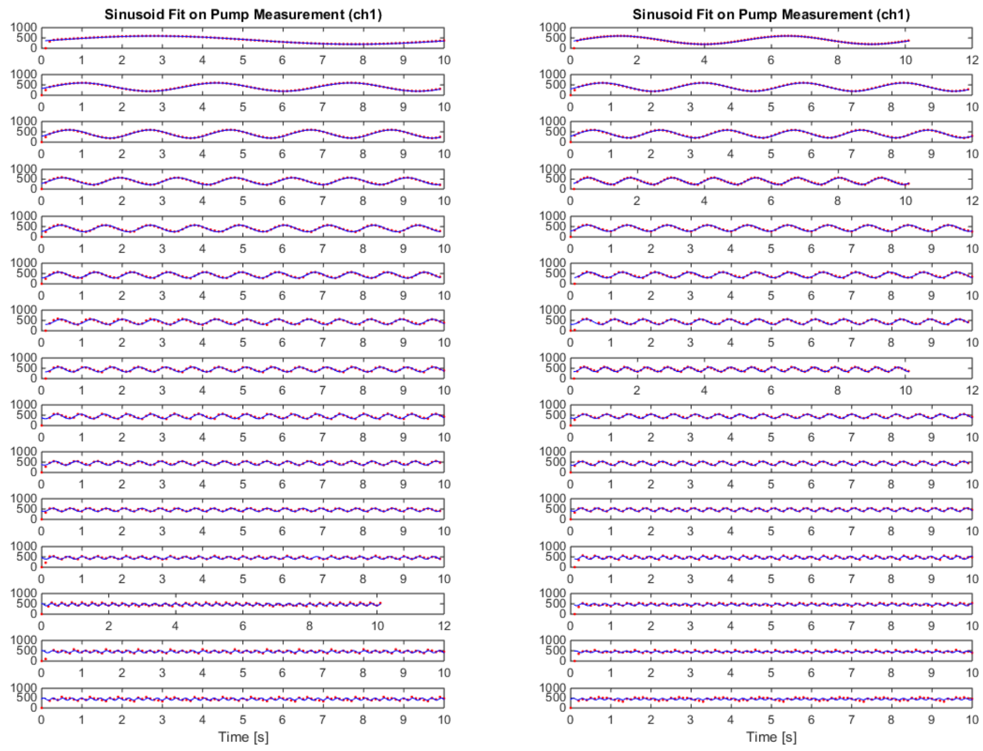


Figure 3.7: Pump output data, $0.1Hz$ to $5Hz$

Experiments were performed where the pressure pump was excited by sinusoidal inputs of various frequencies. Figure 3.7 shows the corresponding output measurements. To extract the relative magnitude $|H(j\omega)|$ and phase $\angle H(j\omega)$, the least square regression technique was used again.

$$\overbrace{\begin{bmatrix} v_N \\ v_{N-1} \\ \vdots \\ v_0 \end{bmatrix}}^{V} = \overbrace{\begin{bmatrix} \sin(\omega t_N) & \cos(\omega t_N) & 1 \\ \sin(\omega t_{N-1}) & \cos(\omega t_{N-1}) & 1 \\ \vdots & \vdots & \vdots \\ \sin(\omega t_0) & \cos(\omega t_0) & 1 \end{bmatrix}}^{\Psi} \overbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}}^{\Theta} + \overbrace{\begin{bmatrix} e_N \\ e_{N-1} \\ \vdots \\ e_0 \end{bmatrix}}^{E} \tag{3.17}$$

$$|H(j\omega)| = \sqrt{\theta_1^2 + \theta_2^2} \tag{3.18}$$

$$\angle H(j\omega) = \arctan(\frac{\theta_2}{\theta_1}) \tag{3.19}$$

$$offset = \theta_3 \tag{3.20}$$

Once the experimental magnitudes and phases were known, I attempted to construct a pump transfer function base on the experiment data. Base on observation, the pump transfer function $P_{pump}(s)$ should contain a delay component and a low-pass filter component.

$$P_{pump}(s) = \overbrace{e^{-sd}}^{Delay} \overbrace{\frac{K}{s+p}}^{Low-pass} \tag{3.21}$$

$$P_{pump}(s) \approx \frac{1 - s\frac{d}{2}}{1 + s\frac{d}{2}} \frac{K}{s+p} \qquad (Pade \quad approximation) \tag{3.22}$$

Delay d is caused by communication with the embedded micro-controller, and is measured on the PC during experiment. Gain K and pole p are unknowns need to be found. Knowing that $DCgain = 1$, final value theorem shows that $K = p$.

$$\lim_{t\to\infty} v(t) = \lim_{s\to 0} sP_{pump}(s)\frac{1}{s} \tag{3.23}$$

$$DCgain = 1 = P_{pump}(0) = \frac{1-0}{1+0}\frac{K}{0+p} = \frac{K}{p} \tag{3.24}$$

Lastly, to account for discretization effects of the experiment data, a zero-order-hold component $H_{zoh}(s)$ was added to the pump transfer function $P_{pump}(s)$.

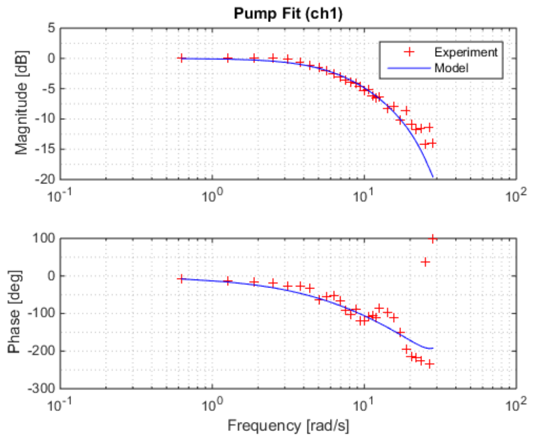
$$\tag{3.25}$$

$$H_{zoh}(s) = \mathcal{L}\{step(t)\} - \mathcal{L}\{step(t-T)\} \tag{3.26}$$

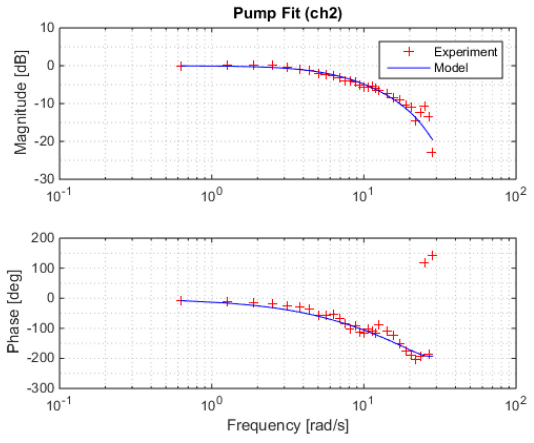
$$= \frac{1 - e^{sT}}{s} \tag{3.27}$$

$$G_{pump}(s) = H_{zoh}(s)P_{pump}(s) \tag{3.28}$$

In figure 3.8, the pump model $G_{pump}(s)$ is compared with experimental $|H(j\omega)|$ and $\angle H(j\omega)$, to confirm that the fit is reasonable.



(a) Pump channel 1 Bode



(b) Pump channel 2 Bode

Figure 3.8: Pump frequency response

Chapter 4

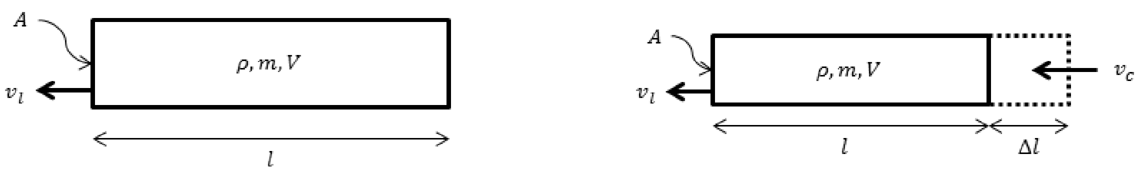
Modelling

In this chapter, I attempt to model fluid dynamics in a microfluidic channel network. In particular, I am interested in predicting droplet positions when the network is subjected to known inlet pressures. The problem is multi-input-multi-output (MIMO) by nature, i.e. the chip has multiple inlets, and changing pressures at each inlet will affect the entire chip and disrupt droplets everywhere on the network. A second challenge concerns with how to capture a huge variety of different channel dimensions and network topologies that are commonly used in microfluidic experiments.

The MIMO problem is solved by representing the model in state space form. On the other hand, the challenge of topology varieties is solved using electric circuit analogy. Fluid dynamics of a single channel is captured as passive circuit elements, i.e. resistance, inductance, and capacitance. They are then combined into an electric circuit that can represent any channel networks. The state space model is then extracted from the electric circuit, and used for controller design in chapter 6.

The model assumes that fluid flow in channels are 1-dimensional, and that droplet lengths are small ($< 1mm$) compare to channel lengths ($\sim 10mm$). It also assumes that droplets are in the squeezing regime, and move with relatively small slippages relative to the continuous phase.

4.1 Passive Circuit Elements



(a) Rigid control volume (b) Axial flexible control volume

Figure 4.1: Control volumes

Figure 4.1 shows two control volumes (CV) representing the same fluid channel. The first CV is rigid, has cross section area A, contains incompressible fluid of density ρ, mass m, volume V, and is moving at velocity v_l. The second CV is axial flexible, and changes length $\triangle l$ depends on incoming flow with velocity v_c.

Applying Reynolds transport theorem on the rigid CV's momentum results in

$$\frac{d}{dt}\int \rho \cdot v_l dV = \int \frac{\partial}{\partial t}(\rho \cdot v_l)dV + flux \tag{4.1}$$

The $flux$ term is eliminated since CV is rigid and contains incompressible fluid. The LHS of the equation is derivative of momentum, which is equal to $netforce$ acting on the CV. Realizing that $netforce = \triangle P \cdot A$, the equation becomes

$$netforce = \frac{d}{dt}(m \cdot v_l) = m \cdot \dot{v_l} + \dot{m} \cdot v_l \tag{4.2}$$

$$\triangle P \cdot A = \rho \cdot l \cdot A \cdot \dot{v_l} \tag{4.3}$$

Making the analogy that pressure $\triangle P$ is voltage, velocity v_l is current, definition of inductance L is obtained. In this analogy, L depends only on density of the fluid and length of the channel.

$$L = \frac{\triangle P}{\dot{v_l}} = \rho \cdot l \tag{4.4}$$

Applying Reynolds transport theorem on the mass of the axial flexible CV results in

$$\frac{d}{dt}\int \rho dV = \int \frac{\partial \rho}{\partial t} dV + flux \tag{4.5}$$

By neglecting $\frac{\partial \rho}{\partial t}$ and substituting $flux = \rho \cdot A \cdot v_c$, the equation becomes

$$\frac{d}{dt}(\rho \cdot V) = \rho \cdot \dot{V} + \dot{\rho} \cdot V = \rho \cdot A \cdot v_c \tag{4.6}$$

Density change is substituted as $\dot{\rho} = (\frac{\rho}{\beta}\frac{\triangle P}{\triangle t})$, where β is the adiabatic bulk modulus.

$$\rho \cdot (A \cdot \frac{\triangle l}{\triangle t}) + (\frac{\rho}{\beta}\frac{\triangle P}{\triangle t}) \cdot A \cdot l = \rho \cdot A \cdot v_c \tag{4.7}$$

Lastly, Hooke's law is applied to recover $\triangle l = \frac{\triangle P \cdot A}{\kappa}$. The equation is integrated with respect to time and becomes

$$A \cdot \frac{\triangle P \cdot A}{\kappa} + \frac{\triangle P \cdot A \cdot l}{\beta} = A \cdot x_c \tag{4.8}$$

$$\triangle P(\frac{A}{\kappa} + \frac{l}{\beta}) = x_c \tag{4.9}$$

Making the analogy that pressure $\triangle P$ is voltage, displacement x_c is charge, this equation defines capacitance C. In this definition, C depends on the cross section area of the channel A, effective stiffness κ which is calculated from the chip material Young modulus, channel length l, as well as adiabatic bulk modulus β of the fluid.

$$C = \frac{x_c}{\triangle P} = \frac{A}{\kappa} + \frac{l}{\beta} \tag{4.10}$$

The definition of resistance is attained from the Hagen-Poiseuille law.

$$R = \frac{\triangle P}{v} = \frac{32 \cdot \mu \cdot l}{d_h^2} \tag{4.11}$$

Altogether, equation 4.4, 4.10 and 4.11 describes the inertial, compliance, and damping effects that take place in a single microfludic channel.

4.2 Channel Network

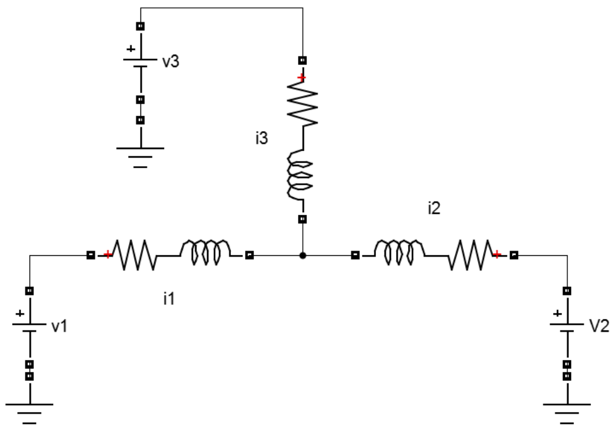


Figure 4.2: An arbitrary electric circuit

For an arbitrary electric circuit such as the one shown in figure 4.2, its state space model is derived using the Kirchoff's circuit laws. For example, a hand derived state space model is shown below, which describes the currents at all three branches $i_1(t), i_2(t), i_3(t)$ given any voltage inputs $v_1(t), v_2(t), v_3(t)$.

$$\dot{\begin{bmatrix} i_2 \\ i_1 \end{bmatrix}} = \begin{bmatrix} \frac{R_1+R_2-R_1L_1/(L_1+L_3)}{L_1+L_2-(L_1^2/(L_1+L_3))} & \frac{R_1-L_1(R_1+R_3)/(L_1+L_3)}{L_1+L_2-(L_1^2/(L_1+L_3))} \\ \frac{R_1-L_1(R_1+R_2)/(L_1+L_2)}{L_1+L_3-L_1^2/(L_1+L_3)} & \frac{R_1+R_3-R_1L_1/(L_1+L_2)}{L_1+L_3-L_1^2/(L_1+L_3)} \end{bmatrix} \begin{bmatrix} i_2 \\ i_1 \end{bmatrix} + \begin{bmatrix} \frac{L_1/(L_1+L_3)-1}{L_1+L_2-(L_1^2/(L_1+L_3))} & \frac{1}{L_1+L_2-(L_1^2/(L_1+L_3))} & \frac{(-L_1)/(L_1+L_3)}{L_1+L_2-(L_1^2/(L_1+L_3))} \\ \frac{L_1/(L_1+L_2)-1}{L_1+L_3-L_1^2/(L_1+L_3)} & \frac{(-L_1)/(L_1+L_2)}{L_1+L_3-L_1^2/(L_1+L_3)} & \frac{1}{L_1+L_3-L_1^2/(L_1+L_3)} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{4.12}$$

$$\begin{bmatrix} i_3 \\ i_2 \\ i_1 \end{bmatrix} = \begin{bmatrix} -1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_2 \\ v_1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{4.13}$$

However, deriving state space models in symbolic form is tedious and time consuming. The task is automated using the Simulink Simscape toolbox instead.

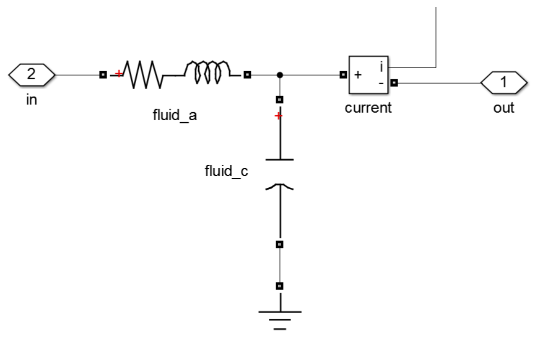


Figure 4.3: The electric circuit that describe a single microfluidic channel

First, each channel in a network is represented by the circuit shown in figure 4.3. Resistors and inductors are connected in series since they contribute mutual exclusively to the pressure loss of a channel. Capacitors on the other hand, are connected in parallel to ground, since the pressure that drives fluid motion simultaneously expands the channel.

To represent a vast network, figure 4.3 is cloned and connected according to the design of the microfluidic chip. A Matlab script then calculates the values of R, L, and C automatically according to channel geometries and material properties. In figure 4.4, the Simulink model is shown together with its matching chip.

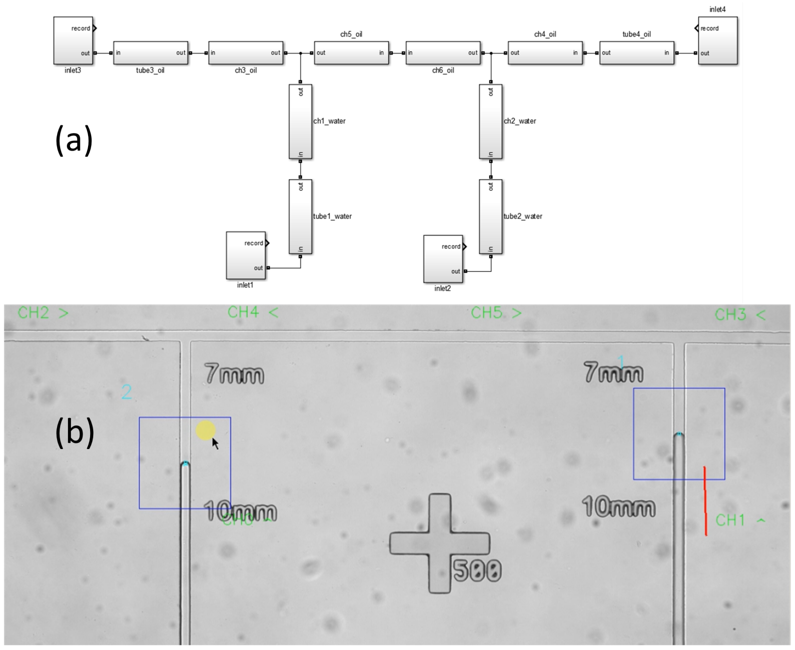


Figure 4.4: (a) Simulink model for a 4 inlets chip, each block contains the circuit shown in figure 4.3. (b) The 4 inlets chip viewed under microscope

$$\begin{bmatrix} \dot{i} \\ v^* \end{bmatrix} = [A] \overbrace{\begin{bmatrix} i \\ v^* \end{bmatrix}}^{states} + [B] \overbrace{[v]}^{inputs} \tag{4.14}$$

$$\underbrace{[i]}_{outputs} = [C] \underbrace{\begin{bmatrix} i \\ v^* \end{bmatrix}}_{states} + [D] \underbrace{[v]}_{inputs} \tag{4.15}$$

The state space model obtained is shown above in equation 4.14. The states contain currents i and junction voltages v^* internal to the circuit. The model inputs are source voltages v, while the outputs are channel currents i. Recall from the fluid-electric analogy that source voltages represent inlet pressures driven by the pressure pump, while output currents represent fluid velocities in each individual channels inside a network.

4.3 Discretization

The governing equations of electric circuits are ODEs, which corresponds to continuous state space models. However, with a low sampling frequency, models in discrete form is preferred. Conversion between continuous and discrete states space is discussed below. The process is analogous to the conversion between ODEs and difference equations shown earlier.

$$\dot{x} = Ax + Bu \tag{4.16}$$
$$y = Cx + Du \tag{4.17}$$

Given the above general state space model, its homogeneous solution $x_h(t)$ corresponds to when input $u(t)$ is absent. $x_h(t)$ is solved by defining the matrix exponent e^{At} as a power series.

$$\dot{x}_h = Ax \tag{4.18}$$
$$x_h(t) = e^{At}x(0) \tag{4.19}$$
$$e^{At} = \sum_{N=0}^{\infty} \frac{1}{N!}A^k = I + tA + \frac{t^2 AA}{2!} + \dots \tag{4.20}$$
$$\tag{4.21}$$

To solve for the particular solution $x_p(t)$, an unknown function $v(t)$ is introduced.

$$x_p(t) = e^{At}v(t) \tag{4.22}$$

$x_p(t)$ is substituted back to the state space equation, then expanded using chain rule. This allows the identity of $v(t)$ to be found, as well as $x_p(t)$.

$$\dot{x}_p = Ax_p + Bu \tag{4.23}$$
$$Ae^{At}v(t) + e^{At}\dot{v}(t) = Ae^{At}v(t) + Bu(t) \tag{4.24}$$
$$\dot{v}(t) = [e^{At}]^{-1}Bu(t) = e^{-At}Bu(t) \tag{4.25}$$
$$v(t) = \int_{t_0}^{t} e^{-A\tau}Bu(\tau)d\tau \tag{4.26}$$
$$x_p(t) = e^{At}\int_{t_0}^{t} e^{-A\tau}Bu(\tau)d\tau = \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \tag{4.27}$$

The general solution to the state space model is a combination of its homogeneous solution and particular solution.

$$x(t) = x_h(t) + x_p(t) \tag{4.28}$$
$$= e^{At}x(0) + \int_{t_0}^{t} e^{A(t-\tau)}Bu(\tau)d\tau \tag{4.29}$$

For a given sampling period T, the general solution allows current sampled state $x(kT+T)$ to be related to previous sampled state $x(kT)$.

$$x(kT+T) = e^{AT}x(kT) + \int_{kT}^{kT+T} e^{A(t-\tau)}Bu(\tau)d\tau \tag{4.30}$$

$$= \underbrace{e^{AT}}_{Ad} x(kT) + \underbrace{B\int_{0}^{T} e^{A\eta}d\eta}_{Bd}\, u(kT) \tag{4.31}$$

By applying 1) the zero-order-hold condition $u(\tau) = u(kT)$ for $kT < \tau < (kT+T)$ and 2) a change of variable $\eta = kT - \tau$, the equation is grouped into the desired form $x(kT+T) = A_d x(kT) + B_d u(kT)$, and the conversion $A \rightarrow A_d$ and $B \rightarrow B_d$ is found. The discrete state space model is summarized below.

$$x_k = A_d x_{k-1} + B_d u_{k-1} \tag{4.32}$$

$$y_k = C_d x_k + D_d u_k \tag{4.33}$$

$$A_d = e^{AT} \tag{4.34}$$

$$B_d = B\int_{0}^{T} e^{A\eta}d\eta \tag{4.35}$$

$$C_d = C \tag{4.36}$$

$$D_d = D \tag{4.37}$$

4.4 Current to Charge

The discrete conversion is applied to equation 4.14 to yield its discrete counterpart.

$$\begin{bmatrix} i_k \\ v_k^* \end{bmatrix} = \begin{bmatrix} A_d \end{bmatrix} \begin{bmatrix} i_{k-1} \\ v_{k-1}^* \end{bmatrix} + \begin{bmatrix} B_d \end{bmatrix} \begin{bmatrix} v_{k-1} \end{bmatrix} \tag{4.38}$$

$$\underbrace{\begin{bmatrix} i_k \end{bmatrix}}_{currents} = \begin{bmatrix} C_d \end{bmatrix} \begin{bmatrix} i_k \\ v_k^* \end{bmatrix} + \begin{bmatrix} D_d \end{bmatrix} \begin{bmatrix} v_k \end{bmatrix} \tag{4.39}$$

Equation 4.38 is the discrete model. A_d, B_d, C_d, D_d receives voltage inputs v and outputs currents i, which translate to pump pressures and droplet velocities. For control purposes, the outputs of interest are droplet displacements, or electric charges q. Hence, A_d, B_d, C_d, D_d are integrated.

$$\begin{bmatrix} q_k \\ i_k \\ v_k^* \end{bmatrix} = \overbrace{\begin{bmatrix} [1] & C_d T \\ [0] & A_d \end{bmatrix}}^{A_c} \begin{bmatrix} q_{k-1} \\ i_{k-1} \\ v_{k-1}^* \end{bmatrix} + \overbrace{\begin{bmatrix} D_d T \\ B_d \end{bmatrix}}^{B_c} \begin{bmatrix} v_{k-1} \end{bmatrix} \tag{4.40}$$

$$\underbrace{\begin{bmatrix} q_k \end{bmatrix}}_{charge} = \underbrace{\begin{bmatrix} [1] & [0] \end{bmatrix}}_{C_c} \begin{bmatrix} q_k \\ i_k \\ v_k^* \end{bmatrix} + \underbrace{\begin{bmatrix} [0] \end{bmatrix}}_{D_d} \begin{bmatrix} v_k \end{bmatrix} \tag{4.41}$$

Equatioin 4.40 is the chip model. A_c, B_c, C_c, D_c now outputs charges q, which represent droplet displacements in a channel network.

4.5 Chip with Pump

To account for the pressure pump's dynamics, pump transfer functions attained earlier from experiments are converted to state space form, and combined with the chip model.

Given a pump transfer function $P_{pump}(s)$, its corresponding state space model is obtained from canonical realization.

$$P_{pump}(s) = \frac{V(s)}{U(s)} = \frac{b_{n-1}s^{n+1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} \tag{4.42}$$

$$\dot{\begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n-1} \\ p_n \end{bmatrix}} = \overbrace{\begin{bmatrix} 0 & 1 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 1 \\ -a_0 & -a_1 & \dots & -a_{n-2} & -a_{n-1} \end{bmatrix}}^{A_p} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n-1} \\ p_n \end{bmatrix} + \overbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}}^{B_p} \begin{bmatrix} u \end{bmatrix} \tag{4.43}$$

$$\begin{bmatrix} v \end{bmatrix} = \underbrace{\begin{bmatrix} b_0 & b_1 & \dots & b_{n-2} & b_{n-1} \end{bmatrix}}_{C_p} \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} + \underbrace{\begin{bmatrix} [0] \end{bmatrix}}_{D_p} \begin{bmatrix} u \end{bmatrix} \tag{4.44}$$

Equation 4.43 is the pump state space model. A_p, B_p, C_p, D_p receives command u as inputs, and outputs pressures v. Pump states are denoted by p.

$$\begin{bmatrix} \begin{bmatrix} q_k \\ i_k \\ v_k^* \end{bmatrix} \\ p_k \end{bmatrix} = \begin{bmatrix} A_c & B_c C_p \\ [0] & A_p \end{bmatrix} \begin{bmatrix} \begin{bmatrix} q_{k-1} \\ i_{k-1} \\ v_{k-1}^* \end{bmatrix} \\ p_{k-1} \end{bmatrix} + \begin{bmatrix} B_c D_p \\ B_p \end{bmatrix} \begin{bmatrix} u_{k-1} \end{bmatrix} \tag{4.45}$$

$$\begin{bmatrix} q_k \end{bmatrix} = \begin{bmatrix} C_c & D_c C_p \end{bmatrix} \begin{bmatrix} \begin{bmatrix} q_k \\ i_k \\ v_k^* \end{bmatrix} \\ p_k \end{bmatrix} + \begin{bmatrix} D_c D_p \end{bmatrix} \begin{bmatrix} u_k \end{bmatrix} \tag{4.46}$$

Since pump outputs are essentially inputs to the microfluidic chip, the pump model in equation 4.43 is connected in series with the chip model in equation 4.40, yielding the combined state space model shown in equation 4.45.

Chapter 5

Validation

A tailor measures his or her client carefully, prior to making a suit. It is equally important to validate a model prior to designing a controller. The task of validation, however, presents its own challenges.

For a single-input-single-output (SISO) system, the transfer function $H(s)$ can be factorized into two polynomials. The roots of the denominator polynomial become poles p_i of the system.

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1}s^{m-1} + \cdots + b_1 s + b_0}{s^n + a_{n-1}s^{n-1} + \cdots + a_1 s + a_0} = K\frac{(s+z_1)(s+z_2)\ldots(s+z_m)}{(s+p_1)(s+p_2)\ldots(s+p_n)} \tag{5.1}$$

Poles are important because they dictate system's response. From equation 5.2, it is apparent that negative poles $\mathrm{Re}(p) < 0$ would result in output decay hence a stable system, while positive poles $\mathrm{Re}(p) \geq 0$ would result output divergence hence an unstable system.

$$y(t) = c_0 + c_1 e^{p_1 t} + c_2 e^{p_2 t} + \cdots + c_n e^{p_n t} \tag{5.2}$$

A MIMO system has many transfer functions, yet its poles can be found in a similar fashion. The following equations show that all MIMO transfer functions share the same denominator polynomial $\det(sI - A)$, and the eigenvalues of A are the poles.

$$\dot{x(t)} = Ax(t) + Bu(t) \tag{5.3}$$

$$sX(s) = AX(s) + BU(s) \tag{5.4}$$

$$X(s) = (sI - A)^{-1}BU(s) \tag{5.5}$$

$$Y(s) = C[(sI - A)^{-1}BU(s)] + DU(s) \tag{5.6}$$

$$H(s) = \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D \tag{5.7}$$

$$= \frac{C\,\mathrm{adj}(sI - A)B}{\underbrace{\det(sI - A)}_{charisteristic \quad eqn.}} + D \tag{5.8}$$

Referring back to the chip model shown in equation 4.40, multiple zero poles $p_i = 0$ appear as eigenvalues of A_c. Consequently, the system of interest will oscillate indefinitely at best, and diverge at the slightest disturbance.

5.1 Validation Approach

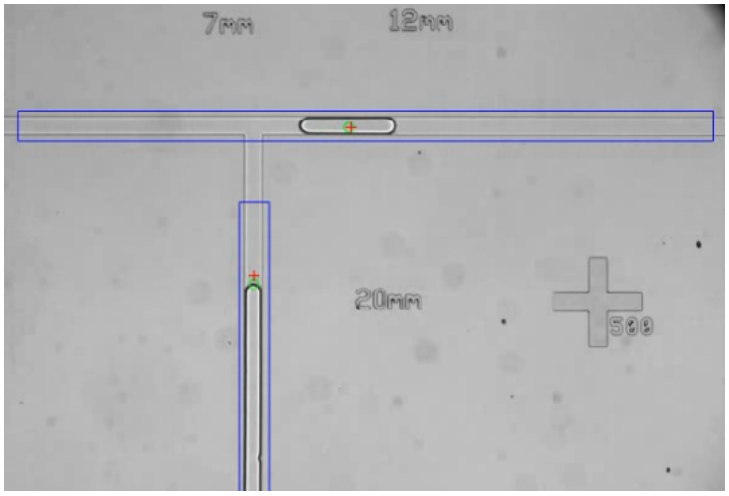


Figure 5.1: Droplet and interface

Figure 5.1, shows a T-junction with a newly generated droplet. Without intervention, the droplet would quickly drift out of view, while the water interface would likely overflow and flood the visible portion of the chip. Since measurements are made with image processing, no valuable data would be obtained in such scenarios.

To overcome this challenge, the validation process begins with designing simple PID controllers to stabilize the system. Unlike the more sophisticated controllers presented in chapter 6, the PID controllers are expected to perform poorly because they ignore coupling dynamics. Furthermore, they lack generality, and will only work in the specific geometry shown in figure 5.1. Nevertheless, they stabilize the system, allowing data acquisition to take place.

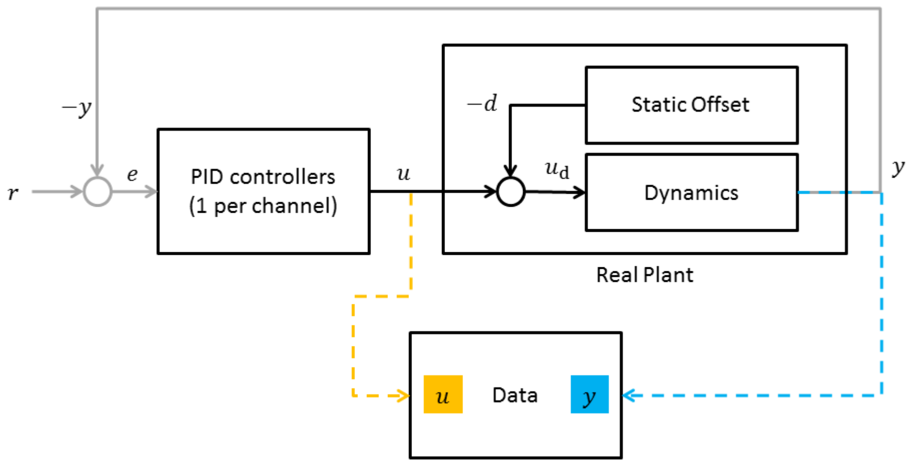


Figure 5.2: Step 1: Experiment with PID controllers

Figure 5.2 shows the experiment set up with PID controllers. During the test, droplet and interface are requested to move according to references r, while the commanded pressures u and measured positions y are recorded.

These raw data are not yet ready to be used for comparison with simulation due to the effects of Laplace pressure. The water-oil interface is known to exhibit a pressure discontinuity due to surface tension. The Laplace pressure effect necessitates a constant bias term in the model, to which a linear system is by definition in-capable of capturing.

Instead, the Laplace pressure is modelled as disturbances d. Although d can not be measured directly, a Kalman filter can provide reasonable estimations d^*, base on analysis of the recorded inputs u and outputs y.

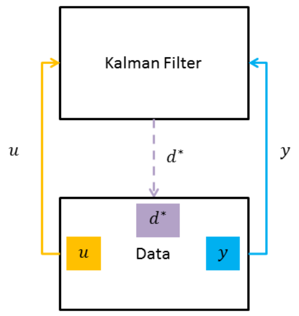


Figure 5.3: Step 2: Estimate disturbance with Kalman filter

After post processing, the estimated disturbances d^* are time averaged and subtracted from inputs u. The modified inputs u^* are fed into the chip state space model in equation 4.40, to generate open-loop output predictions y^*. Finally, predictions y^* are compared with experimental measurements y to validate the model.

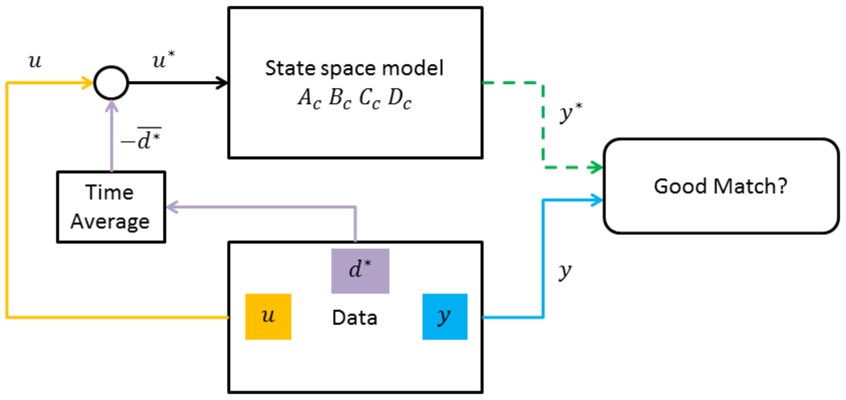


Figure 5.4: Step 3: Open loop simulation

5.2 PID Controller

To quickly stabilize the system, a pair of PID controllers are designed following the emulation approach. Figure 5.5 shows a close-loop system formed by a PID controller, a delay component, and a simplified plant of the channel network. Technically, the problem at hand is a sampled-data system, and controllers ought to be designed in discrete-time. Nevertheless, as a crude approximation, controllers are designed in continuous time, and the sampled-data system is emulated coarsely by the addition of a delay component.

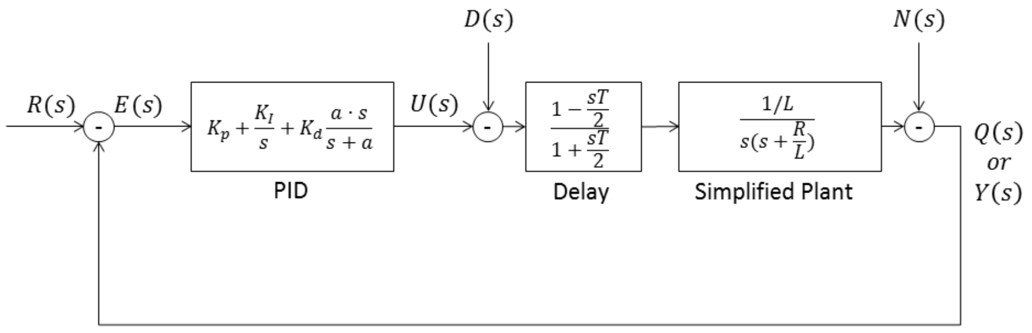

$D(s)$
$N(s)$
$R(s)$
$E(s)$
$K_p + \frac{K_I}{s} + K_d \frac{a \cdot s}{s+a}$
$U(s)$
$\frac{1-\frac{sT}{2}}{1+\frac{sT}{2}}$
$\frac{1/L}{s(s+\frac{R}{L})}$
-
-
-
$Q(s)$
or
$Y(s)$
PID
Delay
Simplified Plant

Figure 5.5: PID block diagram

5.2.1 Closed-loop Stability

$$C = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d \frac{a \cdot s}{s+a} = \frac{g_2 s^2 + g_1 s + g_0}{f_2 s^2 + f_1 s} \tag{5.9}$$

A PID controller consists of position gain K_p, integral gain K_i, and differential gains K_d, a is listed above. For design purposes, it is expanded into polynomials $g(s)$ and $f(s)$.

$$\mathcal{L}\{u(t - \frac{T}{2})\} = e^{-s\frac{T}{2}} U(s) \approx \frac{1 - s\frac{T}{2}}{1 + s\frac{T}{2}} U(s) \tag{5.10}$$

As mentioned earlier, a delay equals to half the sampling-period $T/2$ is included to emulate the discretization effects. Pade approximation is used to describe the delay as a transfer function.

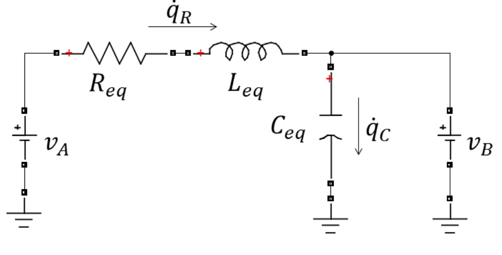


Figure 5.6: Simplified plant

The simplified plant is derived from the governing equations of the fluid element shown in figure 5.6. The plant transfer function is obtained and expanded into polynomials $b(s)$ and $a(s)$ for controller design purposes.

$$v_A - v_B = \dot{q_R} R_{eq} + \ddot{q_R} L_{eq} \tag{5.11}$$

$$v_B = \frac{q_C}{C_{eq}} \tag{5.12}$$

$$\dot{q} = \dot{q_R} + \dot{q_C} \tag{5.13}$$

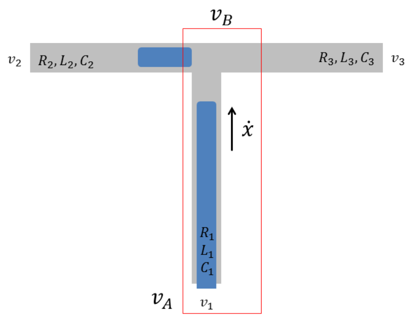
$$\frac{Q(s)}{\triangle P(s)} = \frac{1/L_{eq}}{s(s + R_{eq}/L_{eq}} \tag{5.14}$$

$$P = \frac{Q(s)}{\triangle P(s)} \frac{1 - s\frac{T}{2}}{1 + s\frac{T}{2}} == \frac{b_3 s^3 + b_2 s^2 + b_1 s + b_0}{a_3 s^3 + a_2 s^2 + a_1 s + a_0} \tag{5.15}$$

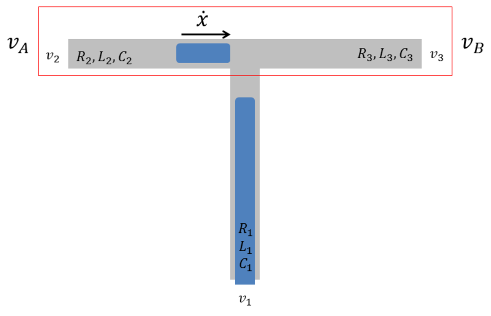
Referring back to figure 5.1, the T-junction has two degrees of freedom, thus requires two simplified plants and two separate PID controllers. Definition of the two simplified plants are shown in figure 5.7, while table 5.1 contains the definitions of R_{eq}, L_{eq}, C_{eq}, v_A, and v_B in each simplification.

Variable	First Plant	Second Plant
v_A	v_2	v_1
v_B	v_3	$\frac{v_2+v_3}{2}$
R_{eq}	$R_1 + \frac{R_2 R_3}{R_2+R_3}$	$R_2 + \frac{R_3 R_1}{R_3+R_1}$
L_{eq}	$L_1 + \frac{L_2 L_3}{L_2+L_3}$	$L_2 + \frac{L_3 L_1}{L_3+L_1}$
C_{eq}	$C_1 + C_2 + C_3$	$C_1 + C_2 + C_3$

Table 5.1: Simplified plant definitions



(a) Definition of first plant



(b) Definition of second plant

Figure 5.7: Definition of simplified plants

The PID controller C, simplified plant with delay P now forms a closed-loop system T_{ry} as shown in figure 5.5. The objective here is to stabilized T_{ry} by finding the appropriate controller gains.

$$\begin{aligned} Q(s) &= PCE(s) && (5.16)\\ &= PC(R(s) - Q(s)) && (5.17)\\ T_{ry} = \frac{Q(s)}{R(s)} &= \frac{PC}{\underbrace{1+PC}_{charisteristic \quad eqn.}} && (5.18) \end{aligned}$$

Within T_{ry}, the characteristic equation contains the closed-loop poles. A desirable closed-loop system will contain a characteristic equation that is composed from stable and desirable poles. By choosing the values of these poles and expanding the characterising equation, the following is obtained, in which $\triangle_i$ are the desirable coefficients.

$$0 = \triangle_5 s^5 + \triangle_4 s^4 + \triangle_3 s^3 + \triangle_2 s^2 + \triangle_1 s + \triangle_0 \qquad (desirable \quad characteristic \quad eqn.) \quad (5.19)$$

To realize the desired closed-loop system, equation 5.9 and 5.15 are substituted into equation 5.18 to yield the following matrix.

$$\overbrace{\begin{bmatrix} a_3 & 0 & b_3 & 0 & 0 \\ a_2 & a_3 & b_2 & b_3 & 0 \\ a_1 & a_2 & b_1 & b_2 & b_3 \\ a_0 & a_1 & b_0 & b_1 & b_2 \\ 0 & a_0 & 0 & b_0 & b_1 \\ 0 & 0 & 0 & 0 & b_0 \end{bmatrix}}^{\Phi} \overbrace{\begin{bmatrix} f_2 \\ f_1 \\ g_2 \\ g_1 \\ g_0 \end{bmatrix}}^{\theta} = \overbrace{\begin{bmatrix} \triangle_5 \\ \triangle_4 \\ \triangle_3 \\ \triangle_2 \\ \triangle_1 \\ \triangle_0 \end{bmatrix}}^{\triangle} \quad (5.20)$$

$$\theta = (\Phi^T \Phi)^{-1} \Phi^T \triangle \quad (5.21)$$

The desirable controller gains are obtained by solving for θ.

5.2.2 Noise Rejection

In reality, the T junction is a coupled system, where actuating the droplet will influence the interface, and vice versa. From a SISO closed-loop system's perspective, these coupling effects appear as noise $N(s)$. To maintain stability, the PID controllers need to be good at noise rejection.

The noise-to-output T_{ny} and disturbance-to-output T_{dy} transfer functions are derived below. Closed-loop frequency responses with respect to noise and disturbance are then extracted.

$$T_{ny} = \frac{Q(s)}{N(s)} = -\frac{1}{1+PC} \quad (5.22)$$

$$T_{dy} = \frac{Q(s)}{D(s)} = -\frac{P}{1+PC} \quad (5.23)$$

Figure 5.8(a) and (b) shows frequency response comparison between T_{ry} and T_{ny}. Controller gains are tuned such that noise induced by another system is attenuated instead of magnified. In other words, the two SISO closed-loop systems each responds to noise and reference in different frequency spectrum, eliminating the ability to resonate with each other.

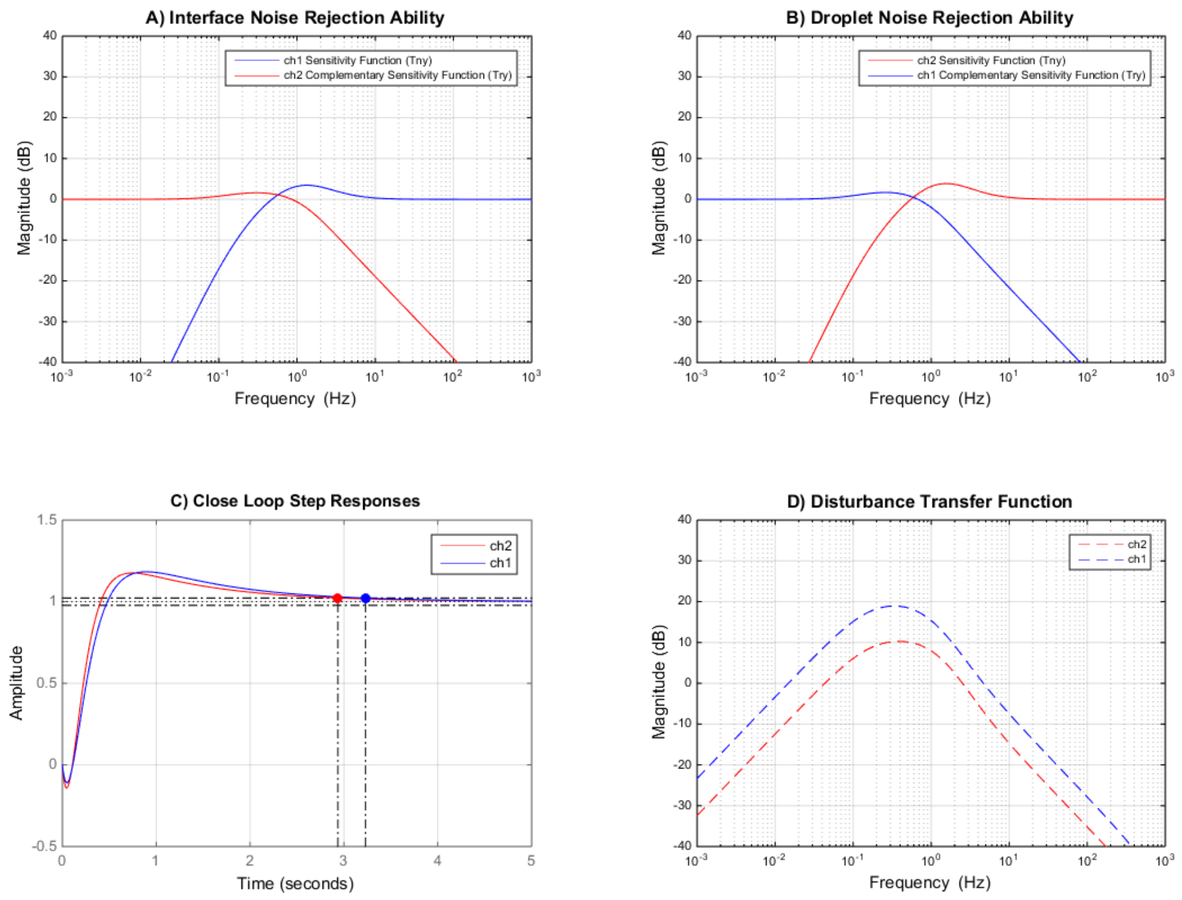


Figure 5.8: (a,b) Noise from one plant interfering with reference of the other plant. (c)Predicted closed-loop step responses. (d) Frequency response of the disturbance to output transfer function

Once satisfactory controllers are obtained, they are discretized using the trapezoid approximation, and implemented as difference equations in the computer program.

5.3 Kalman Filter

Here, the Kalman filter is used as a data post-processing tool. Later on, it will be used in real-time for control purposes. The derivation is provided below.

5.3.1 Derivation

For a general state space model A, B, C, D where $D = [0]$ for a physical system, errors e_k are defined as the difference between measured outputs y_k and predicted outputs $y_{k|k-1}$. Such errors e_k provide information that can be used upon the state predictions $x_{k|k-1}$, turning them into the more accurate state estimates $x_{k|k}$.

$$x_{k|k-1} = Ax_{k-1|k-1} + Bu_{k-1} \tag{5.24}$$
$$y_{k|k-1} = Cx_{k|k-1} \tag{5.25}$$
$$e_k = y_k - y_{k|k-1} \tag{5.26}$$
$$x_{k|k} = x_{k|k-1} + K_k e_k \tag{5.27}$$

The state estimation error covariance matrix $P_{k|k}$ is defined below. Having a minimized $P_{k|k}$ is akin to having state estimates $x_{k|k}$ being as close to the real states x_k as possible.

$$P_{k|k} = Cov\{x_k - x_{k|k}\} = E\{(x_k - x_{k|k})(x_k - x_{k|k}^T\} \tag{5.28}$$
$$= \begin{bmatrix} E\{(x_{1k} - x_{1k|k})(x_{1k} - x_{1k|k})^T\} & \dots & E\{(x_{1k} - x_{1k|k})(x_{nk} - x_{nk|k})^T\} \\ \vdots & & \vdots \\ E\{(x_{nk} - x_{nk|k})(x_{1k} - x_{1k|k})^T\} & \dots & E\{(x_{nk} - x_{nk|k})(x_{nk} - x_{nk|k})^T\} \end{bmatrix} \tag{5.29}$$

The steps below present the means to calculate Kalman gains K_k such that $P_{k|k}$ is minimized. First, the real behaviour of the system is represented as the model plus process noises w_{k-1} and sensor noises v_k. The noises are assumed to have zero bias, and their covariance are renamed as R_w and R_v for convenience.

$$x_k = Ax_{k-1} + Bu_{k-1} + Ww_{k-1} \tag{5.30}$$
$$y_k = Cx_k + v_k \tag{5.31}$$
$$E\{w_k\} = E\{v_k\} = 0 \tag{5.32}$$
$$Cov\{w_k\} = R_w \tag{5.33}$$
$$Cov\{v_k\} = R_v \tag{5.34}$$

Then, the state prediction error covariance matrix $P_{k|k-1}$ is written in terms of the state space model and noises.

$$P_{k|k} = Cov\{x_k - x_{k|k-1}\} \tag{5.35}$$
$$= Cov\{Ax_{k-1} + Bu_{k-1} + Ww_{k-1} - Ax_{k-1|k-1} - Bu_{k-1}\} \tag{5.36}$$
$$= ACov\{x_{k-1} - x_{k-1|k-1}\}A^T + WRwW^T \tag{5.37}$$
$$= AP_{k-1|k-1}A^T + WR_wW^T \tag{5.38}$$

The error covariance matrix S_k is also derived, and represents covariances of e_k

$$\begin{aligned} S_k &= Cov\{e_k\} = Cov\{y_k - y_{k|k-1}\} && (5.39)\\ &= CP_{k|k-1}C^T + Rv && (5.40) \end{aligned}$$

Then, the state estimation error covariance matrix $P_{k|k}$ is also written as

$$\begin{aligned} P_{k|k} &= Cov\{x_k - x_{k|k}\} = Cov\{x_k - (x_{k|k-1} + K_k e_k)\} && (5.41)\\ &= P_{k|k-1} - P_{k|k-1}C^T K_k^T - K_k C P_{k|k-1} + K_k S_k K_k^T && (5.42) \end{aligned}$$

Finally, $P_{k|k}$ is minimized by requiring $\frac{\partial}{\partial K_k} trace(P_{k|k}) = 0$.

$$\frac{\partial tr}{\partial K_k}(P_{k|k}) = \frac{\partial tr}{\partial K_k}(P_{k|k-1}) - \frac{\partial tr}{\partial K_k}(P_{k|k-1}C^T K_k^T) - \frac{\partial tr}{\partial K_k}(K_k C P_{k|k-1}) + \frac{\partial tr}{\partial K_k}(K_k S_k K_k^T) \quad (5.43)$$

$$\begin{aligned} \frac{\partial tr}{\partial K_k}(P_{k|k-1}) &= 0 && (5.44)\\ \frac{\partial tr}{\partial K_k}(P_{k|k-1}C^T K_k^T) &= P_{k|k-1}C^T && (5.45)\\ \frac{\partial tr}{\partial K_k}(K_k C P_{k|k-1}) &= P_{k|k-1}C^T && (5.46)\\ \frac{\partial tr}{\partial K_k}(K_k S_k K_k^T) &= 2K_k S_k && (5.47) \end{aligned}$$

Which yields the expression for obtaining the optimal K_k.

$$\begin{aligned} 0 &= 2K_k S_k - 2P_{k|k-1}C^T && (5.48)\\ K_k &= P_{k|k-1}C^T (CP_{k|k-1}C^T + R_v)^{-1} && (5.49) \end{aligned}$$

5.3.2 Disturbance Estimation

For the problem at hand, R_v manifests from quantization of the camera sensor. Each pixel of the sensor represents a length $\triangle q$, which is measured during camera calibration. When a pixel is triggered, the location probability of the actual object distributes uniformly over the area represented by the pixel. The sensor noise covariance is consequently derived.

$$R_v = [1] \times \frac{\triangle q^2}{12} \tag{5.50}$$

To estimate disturbances, d^* has to be incorporated into the chip state space model (equation 4.40). This is achieved by adding $u_d^* = u - d^*$ and $d_k^* = d_{k-1}^* + [1] \times R_d$ into the model, yielding the augmented plant A_l, B_l, C_l, W_l, shown in 5.51.

$$\overbrace{\begin{bmatrix} \begin{bmatrix} q_k \\ i_k \\ v_k^* \end{bmatrix} \\ d_k^* \end{bmatrix}}^{x_{k|k-1}} = \overbrace{\begin{bmatrix} A_c & -B_c \\ [0] & [1] \end{bmatrix}}^{A_l} \overbrace{\begin{bmatrix} \begin{bmatrix} q_{k-1} \\ i_{k-1} \\ v_{k-1}^* \end{bmatrix} \\ d_{k-1}^* \end{bmatrix}}^{x_{k-1|k-1}} + \overbrace{\begin{bmatrix} B_c \\ [0] \end{bmatrix}}^{B_l} \overbrace{\begin{bmatrix} v_{k-1} \end{bmatrix}}^{u_{k-1}} + \overbrace{\begin{bmatrix} [1] \end{bmatrix}}^{W_l} \overbrace{\begin{bmatrix} [1] \times R_m & [0] \\ [0] & [1] \times R_d \end{bmatrix}}^{R_w} \tag{5.51}$$

$$\underbrace{\begin{bmatrix} q_k \end{bmatrix}}_{y_{k|k-1}} = \underbrace{\begin{bmatrix} C_c & [0] \end{bmatrix}}_{C_l} \begin{bmatrix} \begin{bmatrix} q_k \\ i_k \\ v_k^* \end{bmatrix} \\ d_k^* \end{bmatrix} \tag{5.52}$$

To summarized, the equations used for post processing are listed in table 5.2.

State Prediction Error Covariance	$P_{k\|k-1} = A_l P_{k-1\|k-1} A_l^T + W_l R_w W_l^T$
Kalman Gains	$K_k = P_{k\|k-1} C_l^T (C_l P_{k\|k-1} C_l^T + R_v)^{-1}$
State Prediction	$x_{k\|k-1} = A_l x_{k-1\|k-1} + B_l u_{k-1}$
Output Prediction	$y_{k\|k-1} = C_l x_{k\|k-1}$
Error	$e_k = y_k - y_{k\|k-1}$
State Estimation	$x_{k\|k} = x_{k\|k-1} + K_k e_k$
State Estimation Error Covariance	$P_{k\|k} = ([1] - K_k C_l) P_{k\|k-1}$

Table 5.2: Kalman filter post processing

5.4 Open-Loop Validation

Figure 5.9 shows measurements of system output during experiment. Channel 1 output corresponds to the water interface displacement. Channel 2 output corresponds to the droplet displacement. Channel 3 is empty, but for the sake of completeness, it's displacement is calculated as $y_3 = -(y_1 + y_2)$ as dictated by conservation of mass. The data shows the PID controller regulating droplet and interface displacements according to references that is consisted of a series of steps. Open loop simulation results are overlapped on top to show model accuracy.

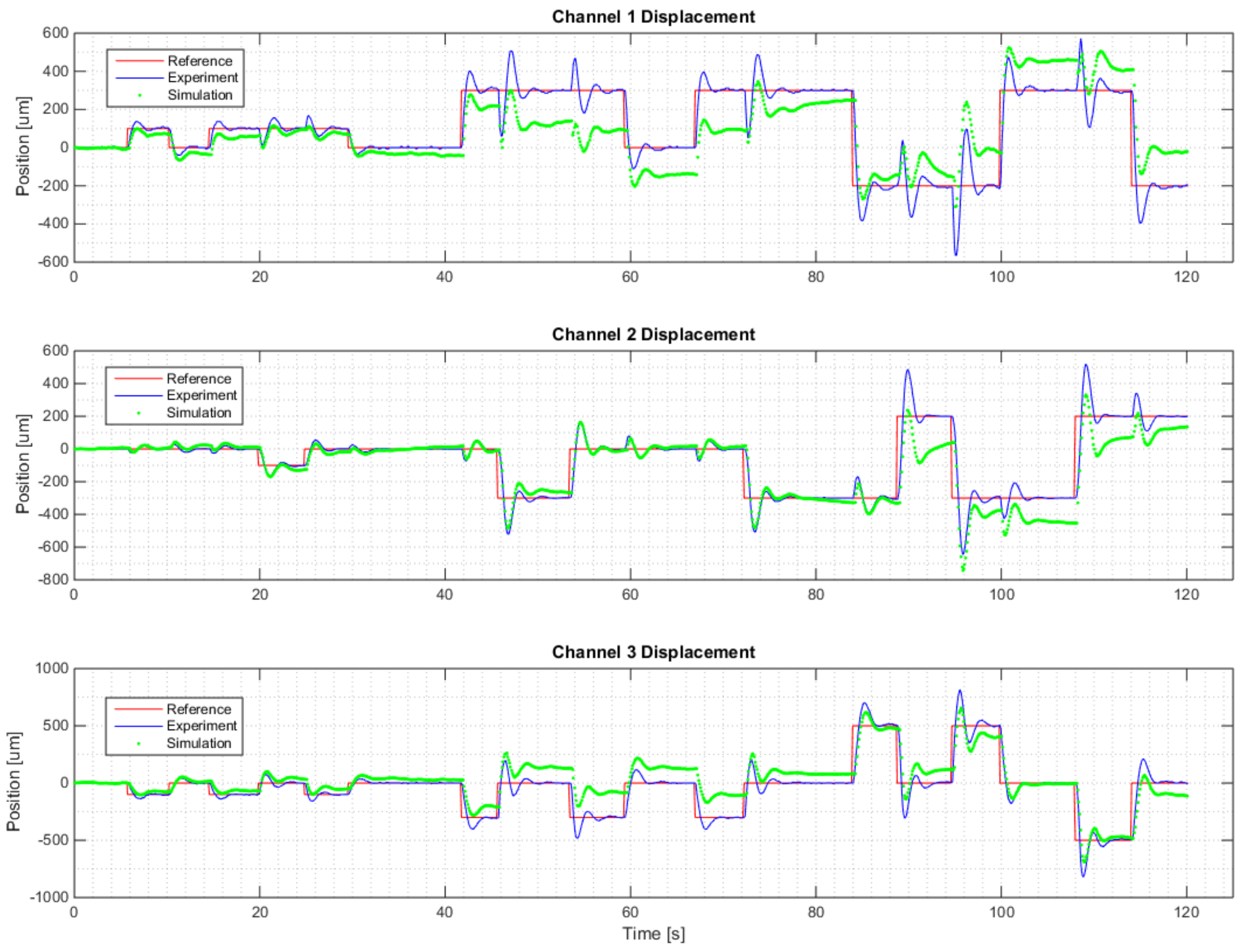


Figure 5.9: Validation data: Displacement

Figure 5.10 plots the raw commands u as well as the modified commands $u^* = u - avg(d^*)$ used in simulation. The Kalman filter successfully estimated the disturbances due to Laplace pressure at the interface.

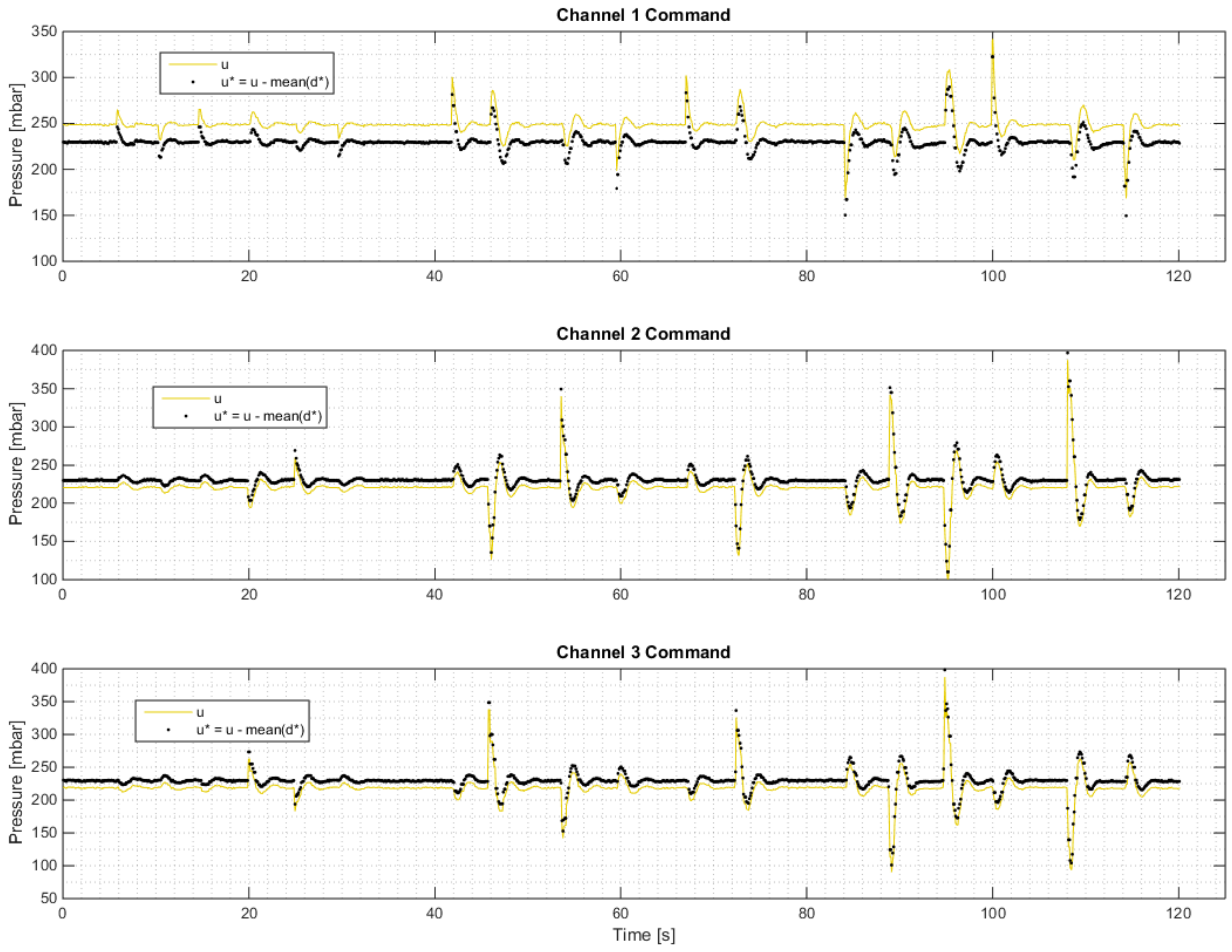


Figure 5.10: Validation data: Command

In figure 5.11, measured droplet and interface velocities are plotted against open loop simulation results. The match is reasonably good, and confirms that the model obtained from chapter 4 is successful at capturing the dynamics of the system.

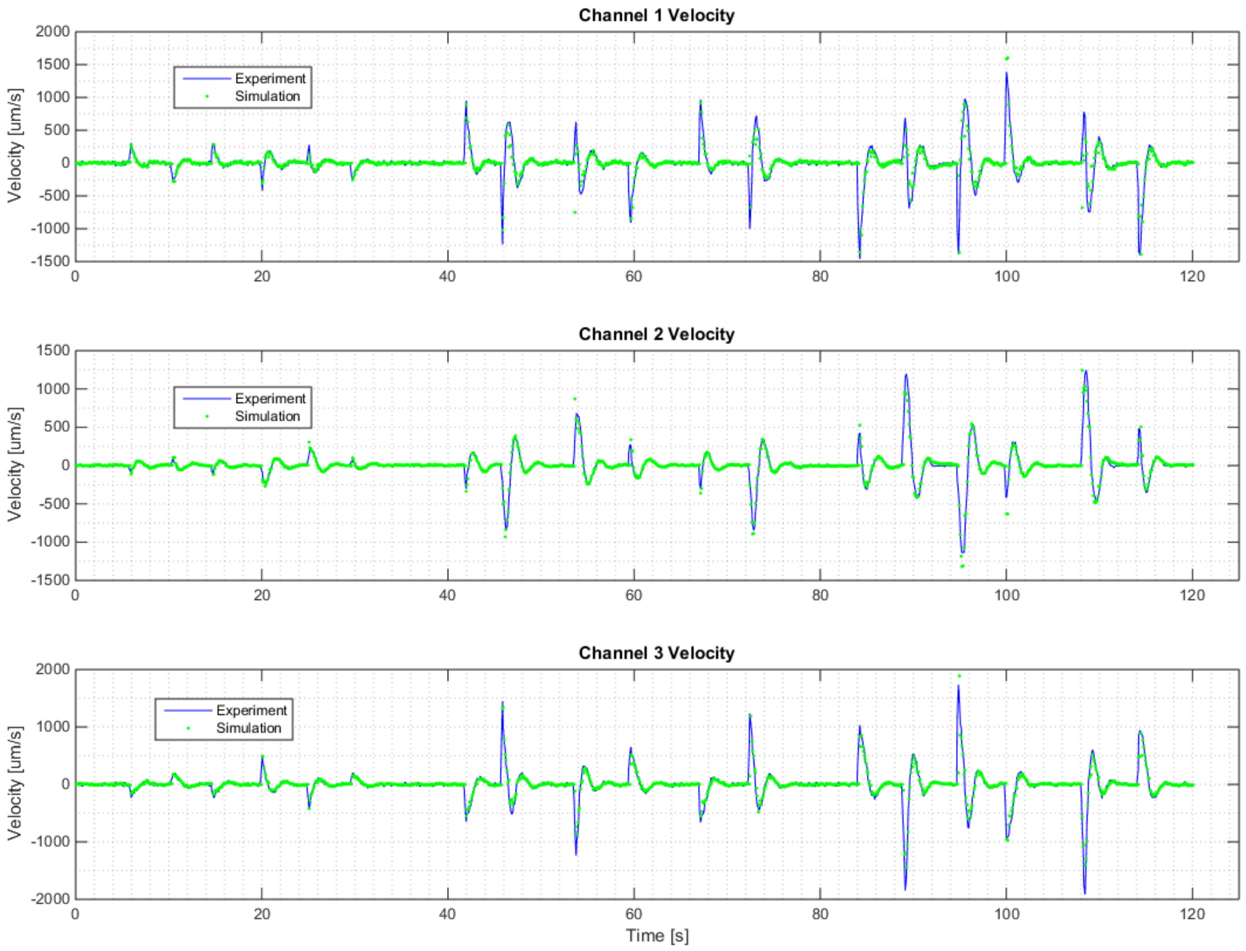


Figure 5.11: Validation data: Velocity

Chapter 6

Controller Design

I begin this chapter by describing the challenges I faced when attempting to design controllers. Given an arbitrary channel network, the first challenge concerns with identifying the independent degree of freedoms. This problem is solved by computing the controllability and observability of the plant state space model. The second challenge arise from the computer vision generated feedbacks. Only droplets that are visible could be detected and measured. As droplets move through a channel network, they create many permutations of the same chip model, all of which have to be controlled. This challenge is handled by expanding a single chip model into multiple reduced plants, each corresponds to a special situation. I will then design specific controllers for each of the reduced plant.

Unlike the PID controllers in chapter 5, controllers in this chapter are aware of coupling effects, and can compensate to provide much improved tracking performance. In addition, the controller design process is fully automatic. For a new network topology or channel dimensions, once the chip model is obtained, optimized controllers are calculated without requiring further human interactions.

In the end of this chapter, I test the controllers by applying them in a simulated environment. The ability to "hot-swap" controllers is demonstrated, and the disturbance cancelling feature is verified.

6.1 Model Reduction

Figure 6.1 illustrates a channel network divided into individual channels $ch_1, ch_2, \ldots, ch_5$. The network has four inlets labelled in_1, in_2, in_3, in_4.

Recall that feedback from the system comes from image data. From the controller's perspective, when a droplet moves from ch_3 to ch_5, it is as if ch_3 ceased to exist, while ch_5 suddenly appeared from thin air, even though in reality fluid continues to flow inside ch_3. Similarly, when the droplet moves from ch_5 to ch_4, controllers will need to be switched to reflect that ch_5 is no longer measured, while ch_4 has become the new target.

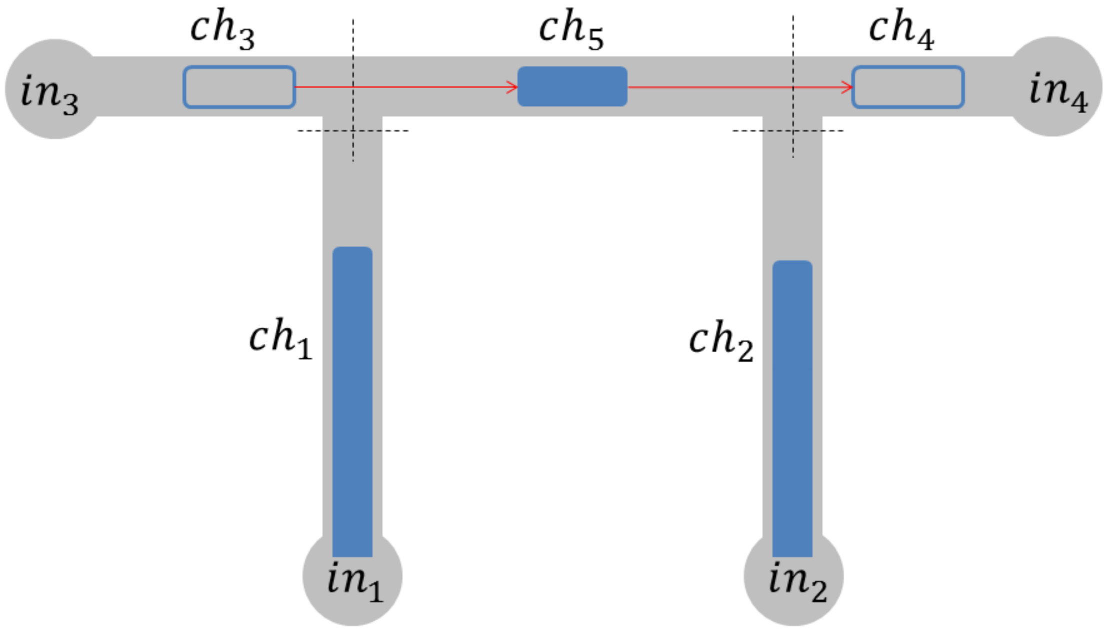


Figure 6.1: Droplet moves from channel to channel

In fact, there are many channel combinations in which the controllers might have to serve. In a given instance, there can be a single droplet visible in ch_1, ch_2, ch_3, ch_4, ch_5, or there might be two droplets visible in (ch_1, ch_2), (ch_1, ch_3), (ch_1, ch_4), etc. To identify all possible combinations, binomial coefficients are used iteratively. Given a network with c number of channels and i number of inlets, the number of combinations N_f with a certain degree-of-freedoms f is calculated as follow.

$$N_{i-1} = \binom{c}{i-1} \tag{6.1}$$

$$N_{i-2} = \binom{c}{i-2} \tag{6.2}$$

$$\vdots = \vdots \tag{6.3}$$

$$N_1 = \binom{c}{1} \tag{6.4}$$

For each channel combination, a reduced plant is created by modifying the full state space chip model. Figure 6.2 shows the chip model A_c, B_c, C_c, D_c derived in equation 4.40 being reduced to the reduced plant A_r, B_r, C_r, D_r by eliminating rolls and columns that correspond to the invisible output q_{k2}

While the full chip model describes droplet displacement q_k in every channel, the reduced plant only contains output q_{k1} and q_{k3}. It is important to note that dynamics of channel 2 remains and continues to affect the model, as evidenced by its velocity state i_{k2} and internal pressure states v^*_{k2}. It is only the displacement state and output q_{k2} that is no longer keep tracked of.

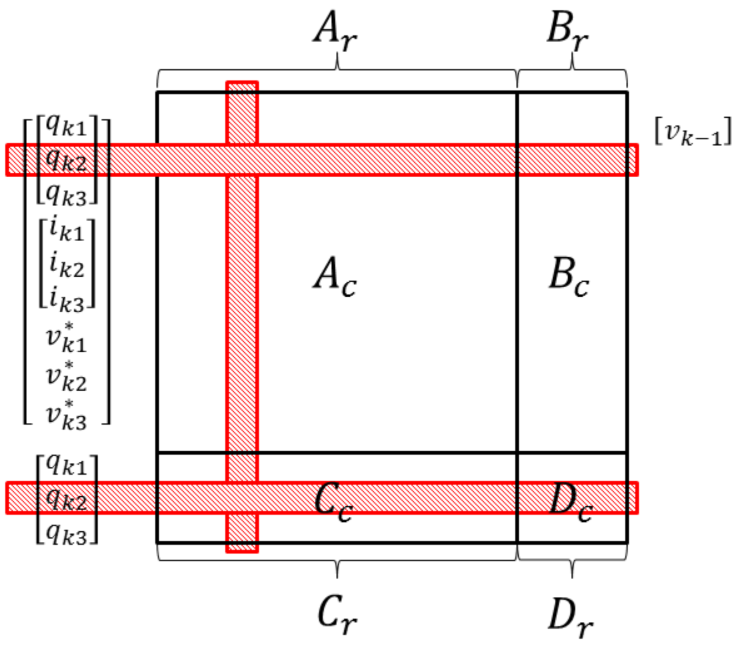


Figure 6.2: Plant reduction

For practical reasons, a channel network as shown in figure 6.1 will be divided into 6 channels instead of 5. For $i = 4, c = 6$, the process described above yields 41 reduced plants A_r, B_r, C_r, D_r, each describing a unique possibility, and each requires a specifically designed controller.

6.2 Controllability and Observability

Not all channel combinations can be controlled. Refer to figure 6.3, it is intuitively impossible to move all three high-lighted droplets independently. For example, if the interface in ch_1 is moved downward while the droplet in ch_3 is moved to the left, conservation of mass dictates that the ch_5 droplet must move to the left. In other words, there are only 2 degree-of-freedoms in those 3 chosen channels.

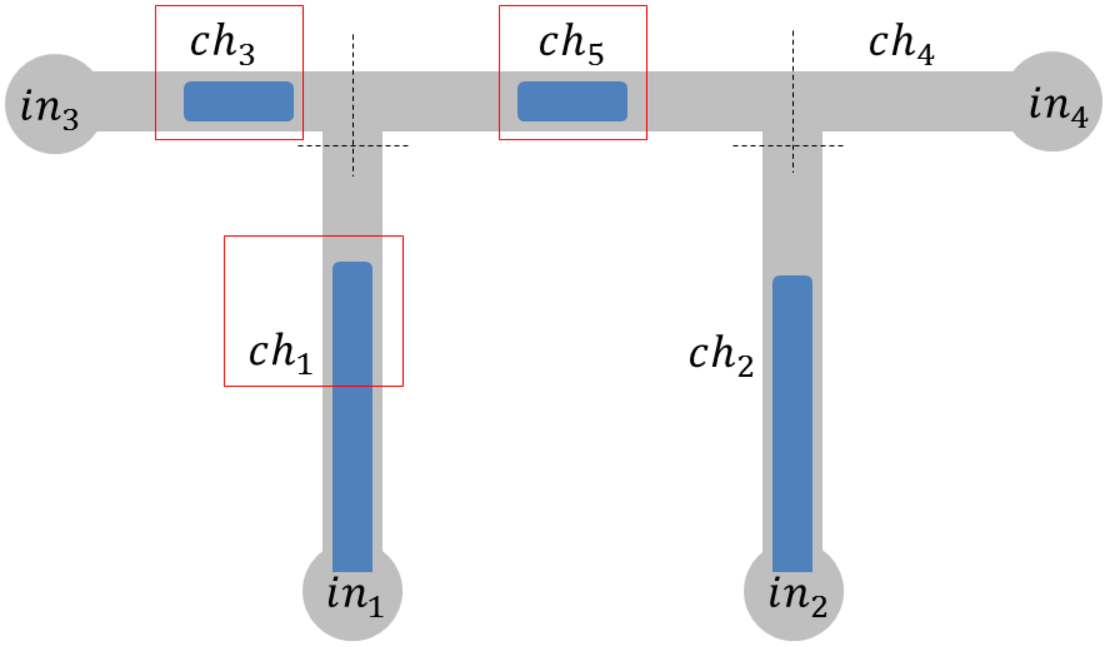


Figure 6.3: Droplet motions are no longer independent

Formally, controllability is defined as: given initial states x_0 and final states x_k, the corresponding inputs $u_0, \ldots u_{k-1}$ exist and can be solved for.

$$\begin{aligned}
x_1 &= Ax_0 + Bu_0 && (6.5)\\
x_2 &= Ax_1 + Bu_1 = A^2x_0 + ABu_0 + Bu_1 && (6.6)\\
x_3 &= Ax_2 + Bu_2 = A^3x_0 + A^2Bu_0 + ABu_1 + Bu_2 && (6.7)\\
\vdots &= \vdots && (6.8)\\
x_k &= A^kx_0 + \begin{bmatrix} B & AB & A^2B & \ldots & A^{k-1}B \end{bmatrix} \begin{bmatrix} u_{k-1} \\ \vdots \\ u_0 \end{bmatrix} && (6.9)\\
u &= \begin{bmatrix} B & AB & A^2B & \ldots & A^{k-1}B \end{bmatrix}^{-1} (x_k - A^kx_0) && (6.10)
\end{aligned}$$

For u to have a solution, $\begin{bmatrix} B & AB & A^2B & \ldots & A^{k-1}B \end{bmatrix}$ needs to be invertible. This criteria is further simplified by the Popov-Belevitch-Hautus (PBH) test, which states that a system is controllable if and only if rank($\begin{bmatrix} A-\lambda[1] & B \end{bmatrix}$) $= n$, where λ and n are eigenvalues and dimension of A respectively.

The observability of a system is defined as: given outputs y_k in the absence of inputs $u_0, \ldots, u_k$, the initial states x_0 can be found.

$$x_1 = Ax_0 \tag{6.11}$$
$$x_2 = Ax_1 = A^2x_0 \tag{6.12}$$
$$\vdots = \vdots \tag{6.13}$$
$$x_k = A^kx_0 \tag{6.14}$$
$$y_k = Cx_k = CA^{k-1}x_0 \tag{6.15}$$
$$\begin{bmatrix} y_0 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{k-1} \end{bmatrix} x_0 \tag{6.16}$$

For the initial states x_0 to have a solution, $\begin{bmatrix} C & CA & \ldots & CA^{k-1} \end{bmatrix}^T$ needs to be invertible. Conversely, using the PBH test, a system is observable if and only if rank($\begin{bmatrix} C \\ A-\lambda[1] \end{bmatrix}$) $= n$.

Conceptually, testing for controllability and observability is as simple as finding the rank of a matrix. However, as Paige [25] has thoroughly demonstrated, matrix rank calculation is numerically challenging. Instead, the following alternative algorithm is used to test if a given reduced plant is controllable and observable.

```
Given matrix $A_r$ of size $n \times n$
Given matrix $B_r$ of size $n \times m$
Given matrix $C_r$ of size $p \times n$
Generate random matrix $G$ of size $m \times p$

let $\lambda = \text{eigenvalue}(A_r)$
let $\zeta = \text{eigenvalue}(A_r + B_r\,\text{random}(m,p)C_r)$

For $i = 1 \to n$: For $j = 1 \to n$:
If $|\lambda_i - \zeta_j| < \text{toleranace} \times \max(|\lambda_i|, |\zeta_j|)$
Then **system is controllable and observable**
```

6.3 Control Law and Observer

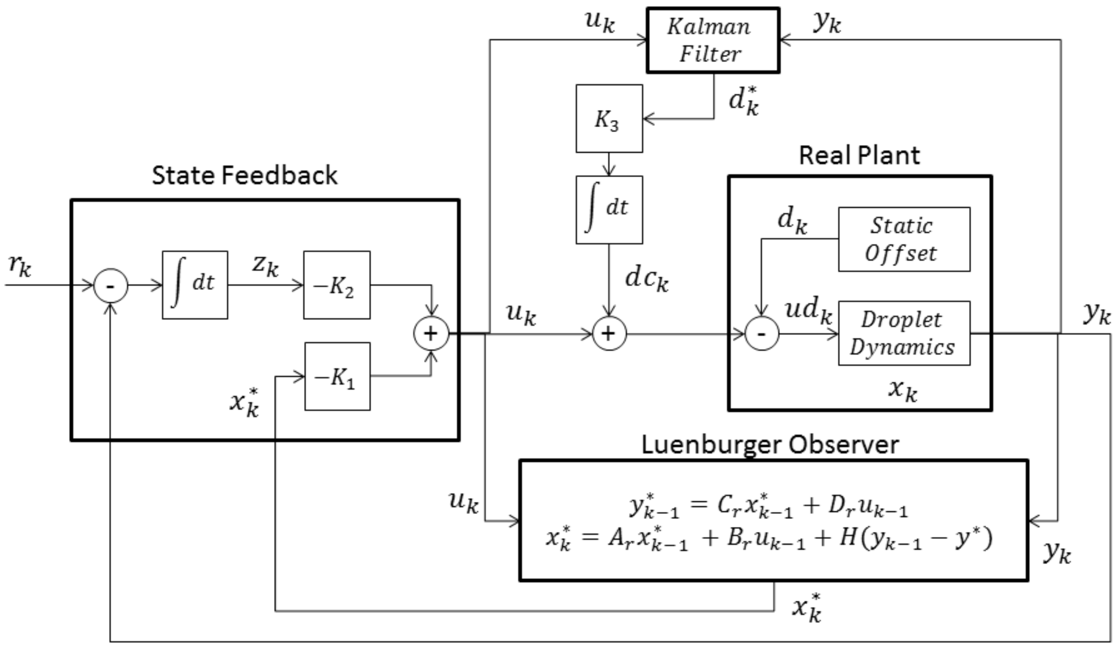


Figure 6.4: Controller topology

Figure 6.4 shows the controller topology that is applied to every channel combinations. A user would indicates the desired droplet movements as reference r_k. This topology would then calculates the required inlet pressure commands u_k, such that droplet displacements y_k moves according to r_k.

The real plant block in figure 6.4 represents the microfluidic chip. Due to assumptions made during modelling, chip manufacturing tolerance, and environmental disturbance, the real plant will behave slightly different from the reduced plant A_r, B_r, C_r, D_r. Furthermore, the real states x_k corresponds to instantaneous pressures and flow velocities that are internal of the chip and cannot be measured directly. Since the controller can not function without knowing these system states, a Luenburger observer is introduced to calculate the state estimates x_k^*. It does so by making predictions y_k^* based on A_r, B_r, C_r, D_r and comparing the predictions with measurements y_k to generate state estimates.

The state feedback block comprise of two parts. First, state estimates x_k^* are multiplied with gain matrix K_1. Secondly, the difference between measurements and references are integrated and then multiplied with gain matrix K_2. The use of integral action $z_k = \int(y_k - r_k)dt$ allows r_k to track y_k with zero steady-state error even when system is subjected to noise. Both state feedback $-K_1 x_k^*$ and integral state feedback $-K_2 z_k$ contributes to forming the new commands u_k.

As mentioned in chapter 5, Laplace pressure across oil-ater interfaces result in static offsets d_k internal to the real plant, to which linear models A_r, B_r, C_r, D_r are incapable of handling. While the integral state feedback mechanism will correct for this effect in a matter of seconds, problem arise when the system switches from one channel combination to another. When A_r, B_r, C_r, D_r is switched, so must K_1, K_2, H. During a switch, the states estimates x_k^* and integral states z_k reset to zero, as neither their identities nor their dimensions are compatible across combinations. Unfortunately, this reset process $z_k = [0]$ is equivalent to erasing the "memory" of the integral state feedback mechanism. This will result in an unsettling oscillation during every combination switch that is damaging and unacceptable.

Consequently, a Kalman filter is employed to estimate d^*_k. The estimated disturbances are accumulated over time to form dc_k, and added to command u_k. This way, over time the effects of d_k are cancelled by dc_k, and u_k converges into the imaginary ud_k. Since the identities and dimensions of dc_k and u_k are the same across all combinations, the Kalman filter disturbance correction mechanism essentially relieves any static correction burden from the integral state feedback mechanism, over a longer period of time.

6.3.1 Integral States

$$z_k = \int (y_k - r_k)dt \tag{6.17}$$

$$= z_{k-1} + \int_{(k-1)T}^{kT} C_r x(t) + D_r u(t) - r(t)dt \tag{6.18}$$

$$= z_{k-1} + TC_r x_{k-1} + TD_r u_{k-1} - T[1]r_{k-1} \tag{6.19}$$

In the equations above, the integral states are written in discrete form under the zero-order-hold assumption. The reduced plant A_r, B_r, C_r, D_r is then augmented with the integral states to yield the integral plant A_z, B_z, C_z, D_z as shown in equation 6.20.

$$\begin{bmatrix} x_k \\ z_k \end{bmatrix} = \overbrace{\begin{bmatrix} A_r & [0] \\ TC_r & [1] \end{bmatrix}}^{A_z} \begin{bmatrix} x_{k-1} \\ z_{k-1} \end{bmatrix} + \overbrace{\begin{bmatrix} B_r \\ TD_r \end{bmatrix}}^{B_z} \overbrace{\begin{bmatrix} v_{k-1} \end{bmatrix}}^{u_{k-1}} + \begin{bmatrix} [0] \\ -T[1] \end{bmatrix} \begin{bmatrix} r_{k-1} \end{bmatrix} \tag{6.20}$$

$$\underbrace{\begin{bmatrix} q_k \end{bmatrix}}_{y_k} = \underbrace{\begin{bmatrix} C_r & [0] \end{bmatrix}}_{C_z} \begin{bmatrix} x_{k-1} \\ z_{k-1} \end{bmatrix} + \underbrace{\begin{bmatrix} D_r \end{bmatrix}}_{D_z} \underbrace{\begin{bmatrix} v_k \end{bmatrix}}_{u_k} \tag{6.21}$$

6.3.2 Closed Loop State Space

From a designer's point of view, commands u are merely dependent variables in the system, and might as well be replaced by $u_{k-1} = -k_1 x^*_{k-1} - K_2 z_{k-1}$. On the other hand, observer performance is quantified by the state estimation errors $e_{k-1} = x_{k-1} - x^*_{k-1}$. Substituting these relations into the reduced plant results in the closed-loop state space as shown in equation 6.38. $A_{cl}, B_{cl}, C_{cl}, D_{cl}$ describes how the system outputs y_k response to references r_k.

$$
\begin{aligned}
x_k &= A_r x_{k-1} + B_r u_{k-1} && (6.22)\\
&= A_r x_{k-1} + B_r(-k_1 x^*_{k-1} - K_2 z_{k-1}) && (6.23)\\
&= A_r x_{k-1} - B_r K_1 (x_{k-1} - e_{k-1}) - K_2 z_{k-1} && (6.24)\\
&= (A_r - B_r K_1) x_{k-1} - B_r K_2 z_{k-1} + B_r K_1 e_{k-1} && (6.25)\\
z_k &= z_{k-1} + T C_r x_{k-1} + T D_r u_{k-1} - T[1] r_{k-1} && (6.26)\\
&= T C_r x_{k-1} + z_{k-1} + T D_r(-k_1 x^*_{k-1} - K_2 z_{k-1}) - T[1] r_{k-1} && (6.27)\\
&= T C_r x_{k-1} + ([1] - K_2 T D_r) z_{k-1} - T D_r K_1 (x_{k-1} - e_{k-1}) - T[1] r_{k-1} && (6.28)\\
&= (T C_r - T D_r K_1) x_{k-1} + ([1] - K_2 T D_r) z_{k-1} + T D_r K_1 e_{k-1} - T[1] r_{k-1} && (6.29)\\
e_k &= x_k - x^*_k && (6.30)\\
&= A_r x_{k-1} + B_r u_{k-1} - A_r x^*_{k-1} - B_r u_{k-1} - H(y_{k-1} - y^*_{k-1}) && (6.31)\\
&= A_r (x_{k-1} - x^*_{k-1}) - H(C_r x_{k-1} + D_r u_{k-1} - C_r x^*_{k-1} - D_r u_{k-1}) && (6.32)\\
&= A_r (x_{k-1} - x^*_{k-1}) - H C_r (x_{k-1} - x^*_{k-1}) && (6.33)\\
&= (A_r - H C_r) e_{k-1} && (6.34)\\
y_k &= C_r x_k + D_r u_k && (6.35)\\
&= C_r x_k - D_r K_1 (x_k - e_k) - D_r K_2 z_k && (6.36)\\
&= (C_r - D_r K_1) x_k - D_r K_2 z_k + D_r K_1 e_k && (6.37)
\end{aligned}
$$

$$
\begin{bmatrix} x_k \\ z_k \\ e_k \end{bmatrix} = \overbrace{\begin{bmatrix} A_r - B_r K_1 & -B_r K_2 & B_r L_1 \\ T C_r - T D_r K_1 & [1] - T D_r K_2 & T D_r K_1 \\ [0] & [0] & A_r - H C_r \end{bmatrix}}^{A_{cl}} \begin{bmatrix} x_{k-1} \\ z_{k-1} \\ e_{k-1} \end{bmatrix} + \overbrace{\begin{bmatrix} [0] \\ -T[1] \\ [0] \end{bmatrix}}^{B_{cl}} \begin{bmatrix} r_{k-1} \end{bmatrix} \qquad (6.38)
$$

$$
\begin{bmatrix} y_k \end{bmatrix} = \underbrace{\begin{bmatrix} C_r - D_r K_1 & -D_r K_2 & D_r K_1 \end{bmatrix}}_{C_{cl}} \begin{bmatrix} x_k \\ z_k \\ e_k \end{bmatrix} + \underbrace{\begin{bmatrix} [0] \end{bmatrix}}_{D_{cl}} \begin{bmatrix} r_k \end{bmatrix} \qquad (6.39)
$$

Alternatively, the closed loop state space matrices $A_{cl}, B_{cl}, C_{cl}, D_{cl}$ can be rewritten in terms of the integral plant A_z, B_z, C_z, D_z. Notice that in the following equations, $K = \begin{bmatrix} K_1 & K_2 \end{bmatrix}$.

$$
\begin{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix} \\ e_k \end{bmatrix} = \overbrace{\begin{bmatrix} A_z - B_z K & \begin{bmatrix} B_r K_1 \\ T D_r K_1 \end{bmatrix} \\ [0] & A_r - H C_r \end{bmatrix}}^{A_{cl}} \begin{bmatrix} \begin{bmatrix} x_{k-1} \\ z_{k-1} \end{bmatrix} \\ e_{k-1} \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} [0] \\ -T[1] \end{bmatrix} \\ [0] \end{bmatrix} \begin{bmatrix} r_{k-1} \end{bmatrix} \qquad (6.40)
$$

$$
\begin{bmatrix} y_k \end{bmatrix} = \begin{bmatrix} C_z - D_z K & D_r K_1 \end{bmatrix} \begin{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix} \\ e_k \end{bmatrix} + \begin{bmatrix} [0] \end{bmatrix} \begin{bmatrix} r_k \end{bmatrix} \qquad (6.41)
$$

Recall that a dynamic system is defined by its A matrix eigenvalues. On the other hand, for a block triangular matrix such as A_{cl} as shown in equation 6.40, its eigenvalues are simply the union of its diagonal sub-matrix's eigenvalues.

$$
\mathrm{eig}(A_{cl}) = \mathrm{eig}(A_z - B_z K) \cup \mathrm{eig}(A_r - H C_r) \qquad (6.42)
$$

The above logic implies that the closed-loop system behaviour is completely dependent on K_1, K_2, H. Furthermore, in the spirit of the separation principle, the state feedback block can be designed in isolation from the observer block.

6.3.3 Linear Quadratic Regulation

Linear Quadratic Regulation (LQR) is employed to identify the values of K_1, K_2, H that corresponds to a desirable closed-loop system.

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{6.43}$$

$$x^T Q x = Q_{11} x_1^2 + Q_{22} x_2^2 + \dots \tag{6.44}$$

$$u^T R u = R_{11} u_1^2 + R_{22} u_2^2 + \dots \tag{6.45}$$

$$\tag{6.46}$$

A desirable closed-loop system is one that is stable and responses quickly. In controls terms, that means having states x and commands u decay to zero as fast as possible. The cost function J above precisely quantifies the system's tendency to settle. By minimizing J, the desirable closed-loop system will be obtained.

In addition to creating a stable closed-loop system, J provides a means to penalize individual states and commands, allowing the controller to be tuned. For example, a large Q_{11} results in droplet displacement state x_1 being penalized heavily, hence droplet in channel 1 will move quicker to achieve its reference position, at the expense of having big velocity and pressure states.

To minimize J, equation 6.40 is simplified by neglecting the effects of r, which yields $x_k = (A_z - B_z K)x_{k-1}$ and $u_k = -K(A_z - B_z K)x_{k-1}$. Substituting them into the cost function in equation 6.43, the algebraic Riccati equation(ARE) is obtained. The optimal state feedback gain K_{opt} is then found by solving for P in the ARE.

$$A_z^T P + P A_z - P B_z R^{-1} B_z^T P + Q = 0 \tag{6.47}$$

$$K_{optimum} = R^{-1} B^T P \tag{6.48}$$

For the observer, the duality property allows $e_k = (A_r - HC_r)e_{k-1}$ to be modified into $e_k^* = (A_r^T - C_r^T H^T)e_{k-1}^*$, which shares the same format as $x_k = (A_z - B_z K)x_{k-1}$. The $H_{optimum}$ is then obtained by solving the ARE again and transposing the result.

6.4 Closed-loop Simulation

A simulink model is created to assess the closed-loop performance of the designed controllers. In simulation, the real plant is represented by the full chip model that contains outputs to all channels, regardless of whether they are visible or invisible. Also, a continuous version of the full chip model is simulated in contrast to the discrete controllers and observers. This allows discretization and quantization effects to be studied.

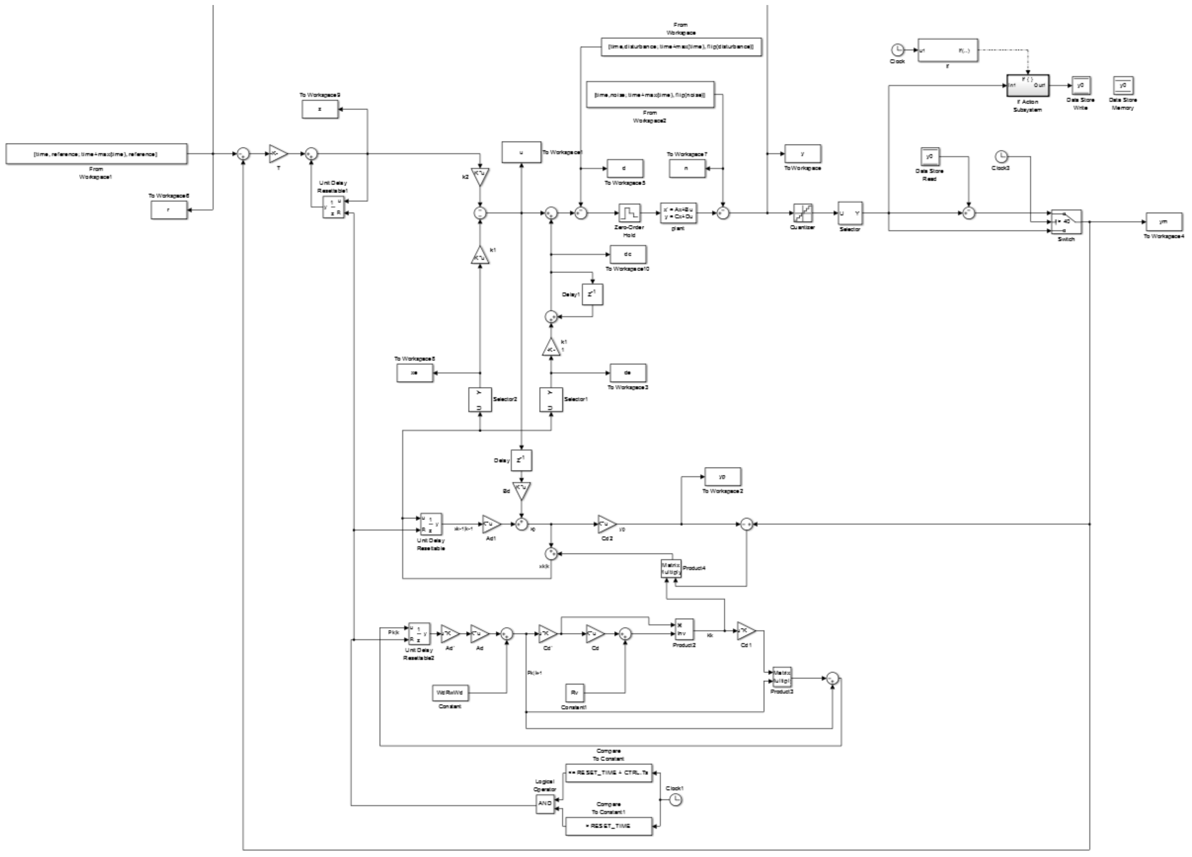


Figure 6.5: Simulation set up

Figure 6.6 shows the simulation results. The plant contains 6 channels and 4 inlets. Only ch_1, ch_2, ch_5 are being controlled as shown, the rest are invisible and can be thought of as having no residing droplets. At $t = 10s, t = 20s, t = 30s$, controller references r is stepped up, as if the user is requesting each droplets to move to a new location. At $t = 40s$, the controller is reset, states and outputs are "zeroed" or "re-calibrated", while the real plant is not affected.

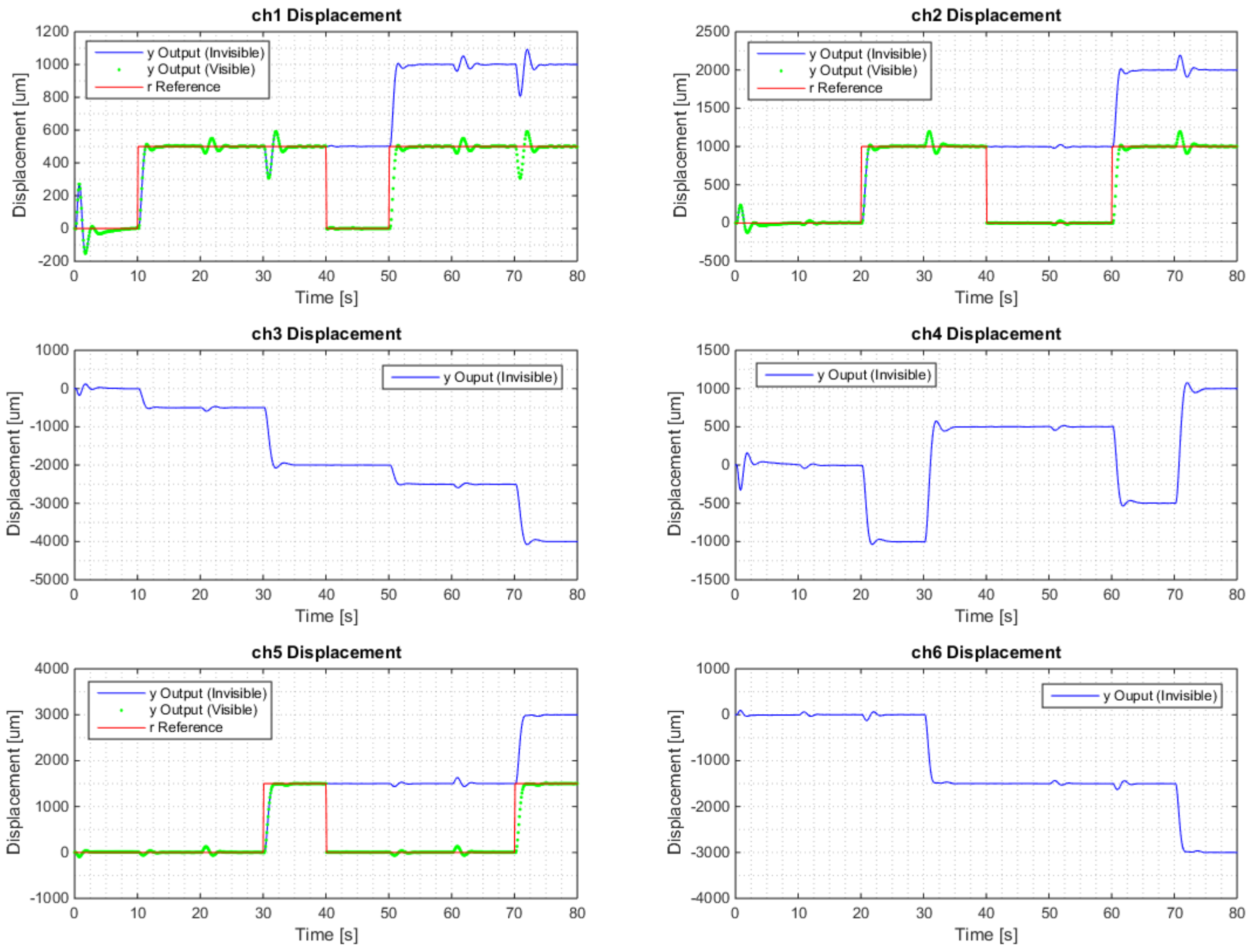


Figure 6.6: Simulation data: Displacement

It is observed that at each step, the output response settles within 3 seconds with no steady-state error. The coupling responses are minor and are dependent on channel geometry and fluid properties. Overall, controller performance is acceptable.

Paying closer attention to $t = 40s$, a very small fluctuation can be spotted on ch_1 output. But comparing to fluctuation at $t = 0s$, where the integral state feedback action has to scramble to counteract initial disturbance, the oscillation induced during reset is negligible, suggesting that the Kalman filter disturbance mechanism is performing. Based on this observation, it can be expected that controller switching during experiment should be a smooth and unnoticeable process.

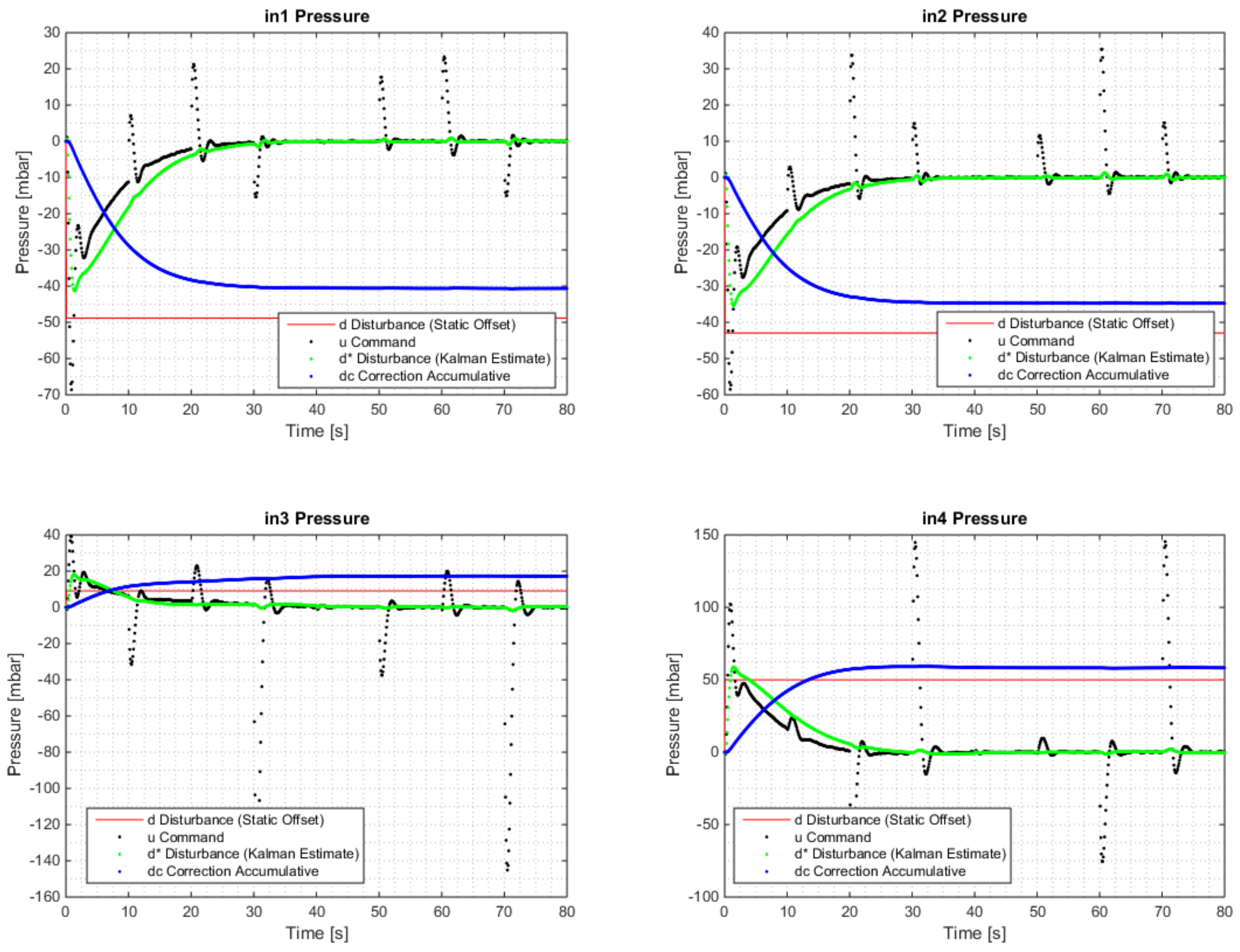


Figure 6.7: Simulation data: Pressure

Figure 6.7 shows pressure related data from the simulation. To simulate the effect of Laplace pressures in the real plant, static offset disturbances d is introduced. When the simulation starts, the integral states feedback mechanism took action and command u immediately to approach d to cancel the static offsets. Over a period of $20s$, the Kalman filter estimates such disturbances d^*, allowing disturbance corrections dc to accumulate and counteract the disturbances. Gradually, u operate closer and closer to 0, and the integral states feedback mechanism is relieved from having to deal with Laplace pressures.

Chapter 7

Software Implementation

RoboDrop is a software created for implementing the controllers designed in chapter 6. RoboDrop contains a Graphic User Interface (GUI) in which user generates, splits, merges, and moves droplets by clicking and dragging them with the mouse. RoboDrop is coded in C++, and the GUI portion is constructed using the Qt library. Underneath the GUI, RoboDrop runs on a multi-thread architecture that facilitates connection with cameras and pressure pumps. The real-time image processing and controller calculation is coded using openCV.

The general work flow for using RoboDrop is as follow. First, the user clones and rearranges the Simulink electric circuit model to represent the microfluidic chip at hand. Then, the optimized controllers are generated automatically by MATLAB and exported into a YAML configuration file. The user then imports the YAML file into RoboDrop, and proceeds to carry out experiments.

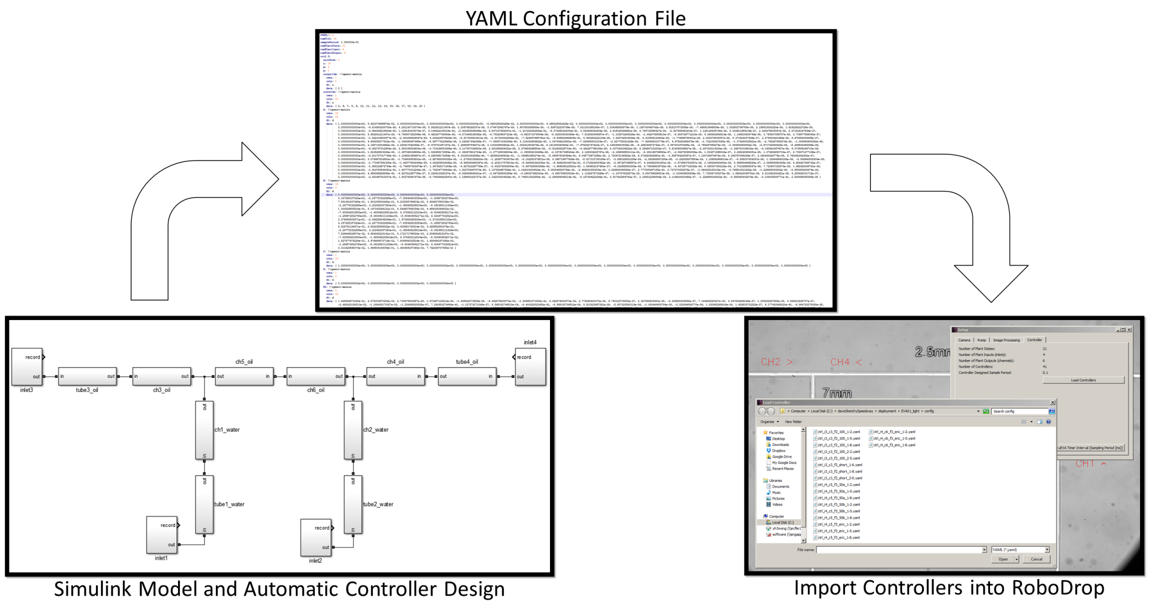
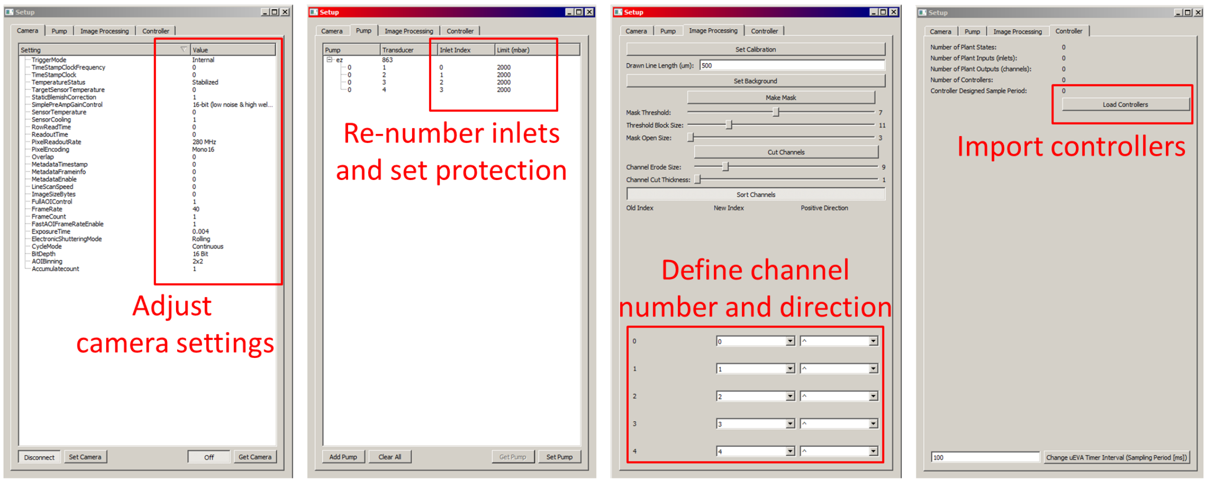


Figure 7.1: RoboDrop work flow

7.1 RoboDrop Architecture

7.1.1 GUI Components

The RoboDrop GUI has four components: Setup, Dashboard, Mainwindow, and Plotter. Setup is a window that contains all the initial settings. The user only interacts with Setup when the program starts up. The Setup window is hidden away during experiment.



Setting	Value
TriggerMode	Internal
TimeStampClockFrequency	0
TimeStampClock	0
TemperatureStatus	Stabilized
TargetSensorTemperature	0
StaticBlemishCorrection	1
SimplePreAmpGainControl	16-bit (low noise & high wel...
SensorTemperature	0
SensorCooling	1
RowReadTime	0
ReadoutTime	0
PixelReadoutRate	280 MHz
PixelEncoding	Mono16
Overlap	0
MetadataTimestamp	0
MetadataFrameinfo	0
MetadataEnable	0
LineScanSpeed	0
ImageSizeBytes	0
FullAOIControl	1
FrameRate	40
FrameCount	1
FastAOIFrameRateEnable	1
ExposureTime	0.004
ElectronicShutteringMode	Rolling
CycleMode	Continuous
BitDepth	16 Bit
AOIBinning	2x2
Accumulatecount	1

Pump	Transducer	Inlet Index	Limit (mbar)
ez	863		
0	1	0	2000
0	2	1	2000
0	3	2	2000
0	4	3	2000

Drawn Line Length (um): 500
Mask Threshold: 7
Threshold Block Size: 11
Mask Open Size: 3
Channel Erode Size: 9
Channel Cut Thickness: 1

Old Index	New Index	Positive Direction
0	0	^
1	1	^
2	2	^
3	3	^
4	4	^

Number of Plant States: 0
Number of Plant Inputs (inlets): 0
Number of Plant Outputs (channels): 0
Number of Controllers: 0
Controller Designed Sample Period: 0
100

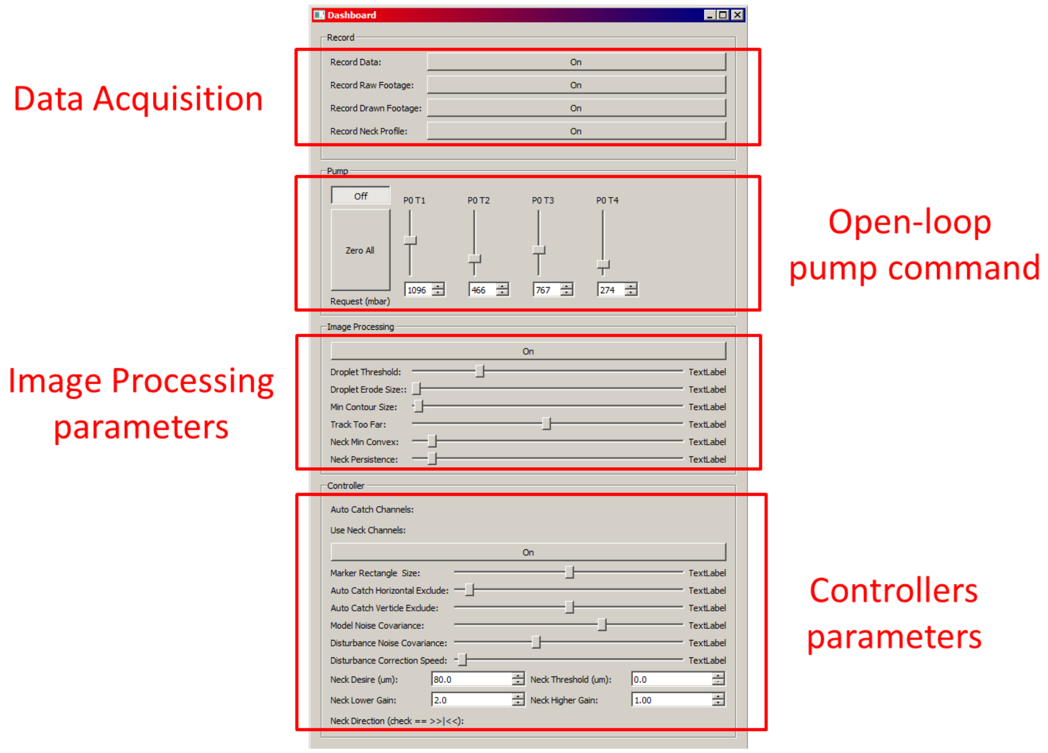
Figure 7.2: RoboDrop Setup panels

Figure 7.2 shows each of the four panels inside the Setup window. The first panel is for connecting to camera, adjusting camera settings, reading camera status, and starting the video stream.

The second panel configures pressure pumps. RoboDrop is designed to work with more than one pressure pumps. Each of the pressure channels can be re-numbered according to how the model is built in Simulink, and how the tubings are connected to the microfluidic.

The third panel contains image processing preparations. Here, the user would create pixel-to-micron calibrations, record a background for use in later subtractions, and define image masks that represent channels.

The last panel displays information about controllers that are currently loaded. It provides the option to load a different sets of controllers, or to change the global sampling period of RoboDrop.



Record
Record Data: On
Record Raw Footage: On
Record Drawn Footage: On
Record Neck Profile: On

Pump
Off
Zero All
Request (mbar)
P0 T1: 1096
P0 T2: 466
P0 T3: 767
P0 T4: 274

Image Processing
On
Droplet Threshold: TextLabel
Droplet Erode Size:: TextLabel
Min Contour Size: TextLabel
Track Too Far: TextLabel
Neck Min Convex: TextLabel
Neck Persistence: TextLabel

Controller
Auto Catch Channels:
Use Neck Channels:
On
Marker Rectangle Size: TextLabel
Auto Catch Horizontal Exclude: TextLabel
Auto Catch Verticle Exclude: TextLabel
Model Noise Covariance: TextLabel
Disturbance Noise Covariance: TextLabel
Disturbance Correction Speed: TextLabel
Neck Desire (um): 80.0 Neck Threshold (um): 0.0
Neck Lower Gain: 2.0 Neck Higher Gain: 1.00
Neck Direction (check == >>|<<):

Figure 7.3: RoboDrop Dashboard

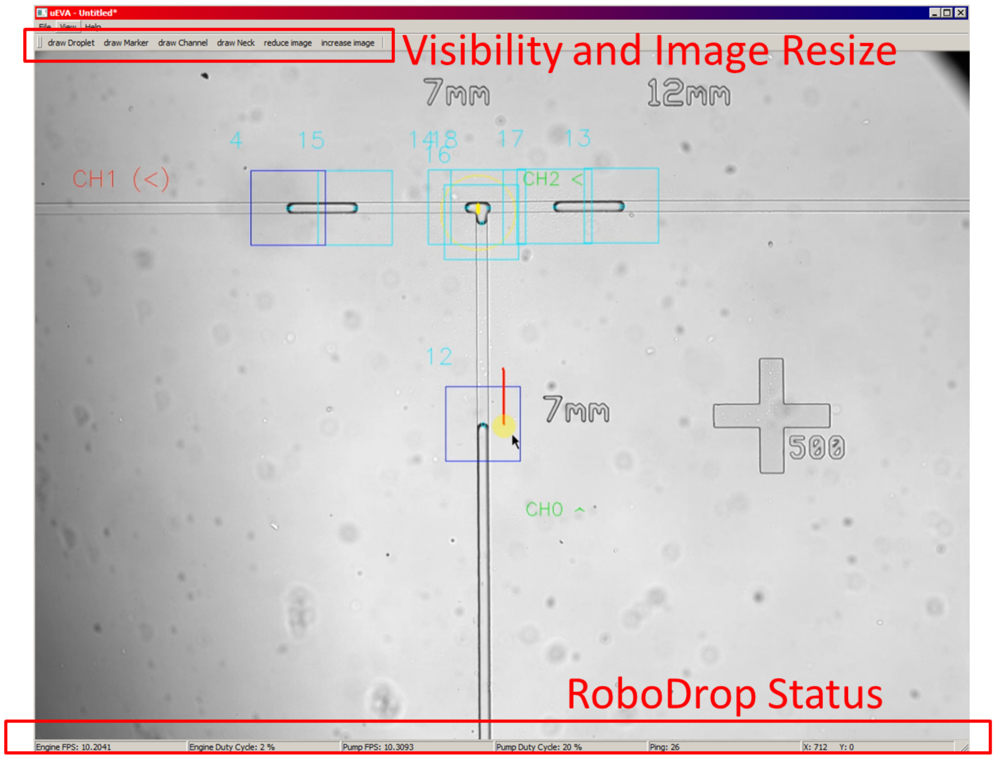
The second component of the RoboDrop GUI is the Dashboard. It contains all the frequently used settings that the user would need to generate droplets, and perform split and merge operations.

The Dashboard window is shown in figure 7.3. The block at the top of the window activates data acquisition functions. When activated, controllers data such as state estimates and measured displacements are recorded into a .csv file. Similarly, real-time droplet necking information can be recorded. RoboDrop can also record the raw video stream and the augmented video stream into .avi files.

Underneath, several sliders allow users to manually change pump pressures when the controllers are not active. This is necessary for purging and bleeding the microfluidic chip prior to starting experiments. Note that the amount and the order of the sliders change depends on the number of pressure pumps that are connected.

The rest of the Dashboard contains image processing parameters such as filter size and threshold, as well as controllers settings such as Kalman filter process noise R_w and disturbance correction gain K_3.

The Mainwindow is where users interact most with RoboDrop. In figure 7.4, the Mainwindow is displaying a live video stream from the camera. The stream has been augmented, highlighting the detected interface, heads and tails of droplets, and the instantaneous neck length. By dragging on these highlighting rectangles, the droplets or interface will follow the user's mouse movement. Splitting and merging can be achieved by clicking and dragging in combination with keyboard short-cuts. The Mainwindow also contains a task bar at the top for resizing the display and for turning the image augmentation on and off. Meanwhile, a status bar at the bottom provides information on computation delay, pump delay, and total delay.



```
Engine FPS: 10.2041   Engine Duty Cycle: 2 %   Pump FPS: 10.3093   Pump Duty Cycle: 20 %   Ping: 26   X: 712   Y: 0
```

Figure 7.4: RoboDrop Mainwindow

The last GUI component is the Plotter, which allows users to visualize ten seconds of data. User can choose from a range of variables such as state estimates, droplet displacements, or the pressure pump write and read values as currently shown in figure 7.5.

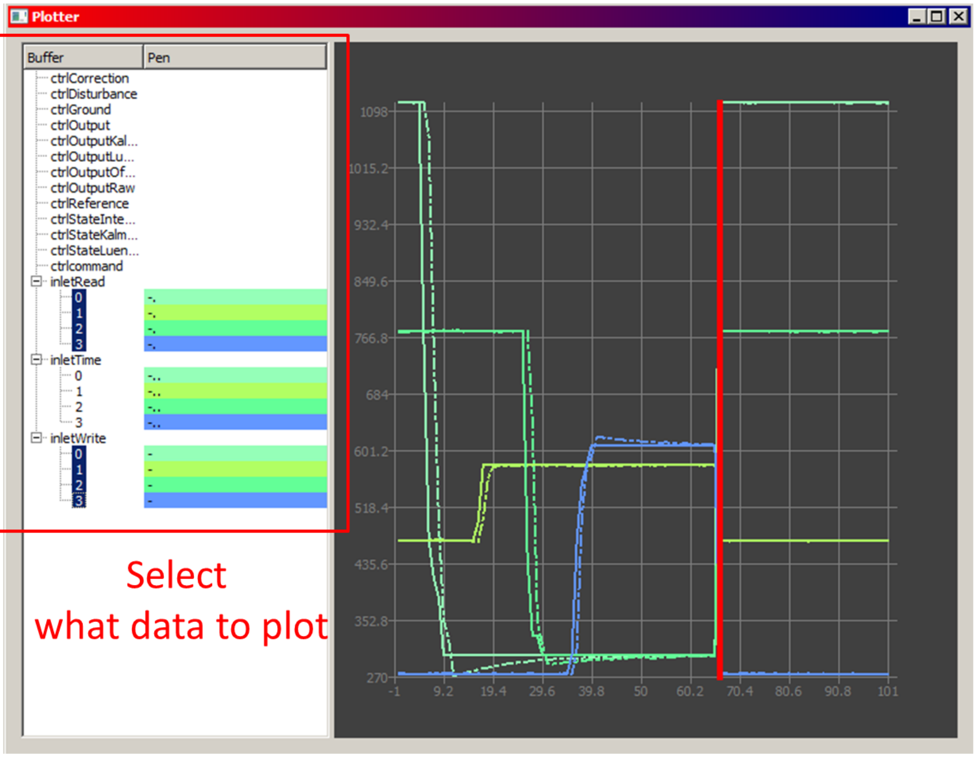


Figure 7.5: RoboDrop Plotter

7.1.2 Multi-threading

Underneath the GUI, various processes take place such as communication with pumps, receiving camera data, and image processing. To allocate computational resources to these processes while keeping the GUI responsive, a multi-threading frame work is constructed as shown in figure 7.6.

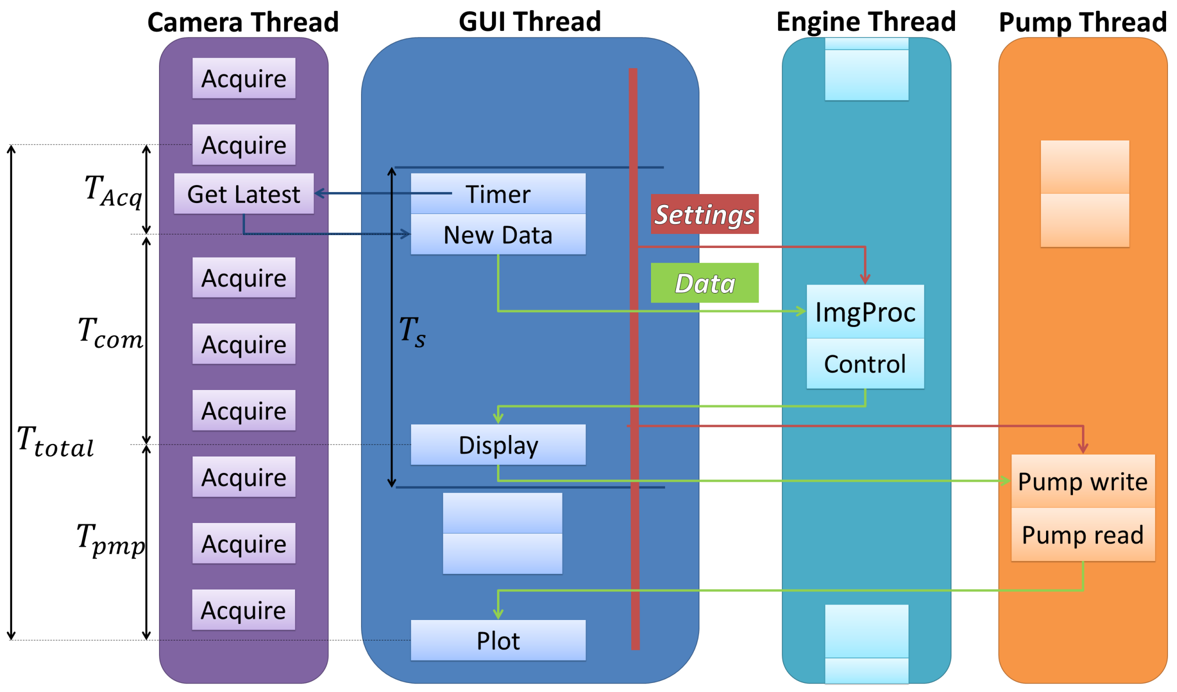


Figure 7.6: RoboDrop multi-threading

The camera thread is responsible for receiving raw video stream from the camera, and constructing image objects from the memory buffer. Occasionally, the camera thread also sends settings to the camera as requested by the user. The camera thread wakes up every $25ms$ and runs at $40fps$, which is much faster than the RoboDrop global sampling period $T_s = 100ms$. This is such that an image will never be "older" than $25ms$, hence putting a cap on acquisition delay T_{Acq}.

The engine thread is where image processing and controller calculations take place. Image processing is computationally expensive, and since the result from which is directly used by controller calculations, it is beneficial to isolate these processes into their own thread. This further allows theoretical improvements such as new image processing algorithm and better controller designs to be made separately from the software development.

The pump thread is necessary since communications with pressure pumps are slow. Pump delay T_{pmp} ranges from $30ms$ to $60ms$, so for the most part the pump thread is just waiting for the hardware to response.

When a thread is in the middle of a process, its variables must be locked, and prevented from being accessed by other threads. Therefore, communication between threads can only happen before or after a process. The Settings and Data objects are designed specifically for this. The Settings object contains user related information such as image processing parameters and pump settings. The Data object contains images, augmentation information, controller variables, and pump command. Notice that Settings is a persistent object. Its content continues to evolve as the user interacts with RoboDrop. Only a snapshot of Settings is sent out each time it is used for thread communication. In contrast, the Data object is a one-used throwaway item.

A timer resides within the GUI thread, and triggers at every $T_s = 100ms$. When it triggers, the GUI thread obtains the latest image from the camera thread, put the image into a brand new Data object, and sends it towards the engine thread while waking it up. The engine thread processes the image, and fills up the Data object with augmentation information, controller calculations, and pump commands. The engine thread then sends the Data object back to the GUI Thread, and goes back to sleep. After receiving the Data object from the engine thread, the GUI thread updates the display with the new image and corresponding augmentation, then sends the Data object towards the pump thread, waking it up in the progress. The pump thread actuates the pumps, and records additional pump read data into the Data object. The pump thread then passes the Data object back to the GUI thread and goes to sleep. At this point, the GUI thread updates the plotter, and discards the Data object.

The time it takes to update display and plotter is minimal, hence the GUI thread is mostly idle. This allows the GUI to stay responsive despite of the gruelling image processing, etc. The computational delay T_{com} is a measure of the time it takes to perform image processing and controller calculations. In this framework, RoboDrop will remain operational even when $T_{total} > T_s$, as long as $T_{com} < T_s$.

7.2 RoboDrop Image Processing

7.2.1 Object Detection

A few preparations are needed before starting the image detection process. The user performs these tasks using the Setup window in RoboDrop, prior to experiment. The preparations involve obtaining the Background, the Droplet Mask, the Marker Mask, and a series of Channel Masks. The Background is required for background subtraction during object detection. The Droplet Mask and Marker Mask are used to exclude noise, while the Channel Masks allow RoboDrop to identify which channel an object is residing in at a given moment.

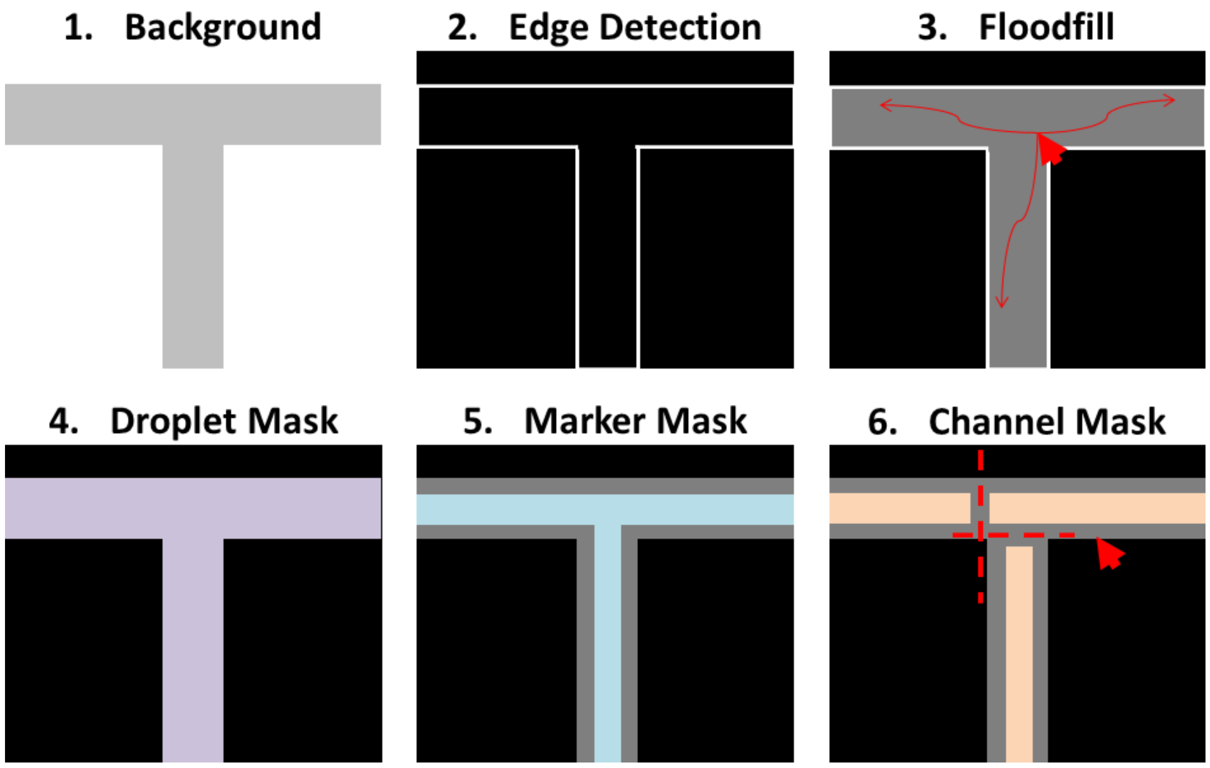


Figure 7.7: Image masks

Figure 7.7 shows the process for obtain these masks. First, an image of the Background is taken from the camera stream and stored. The channel walls are identified using a Sobel filter. Then, the user needs to click inside the channel network in order to provide a seed point for the floodfill operation. The image is then cleaned up with erosion followed by dilation to yield the Droplet Mask. The Marker Mask is obtained by further eroding the Droplet Mask, while the Channel Masks are produced when user cuts the Marker Mask into seperate regions.

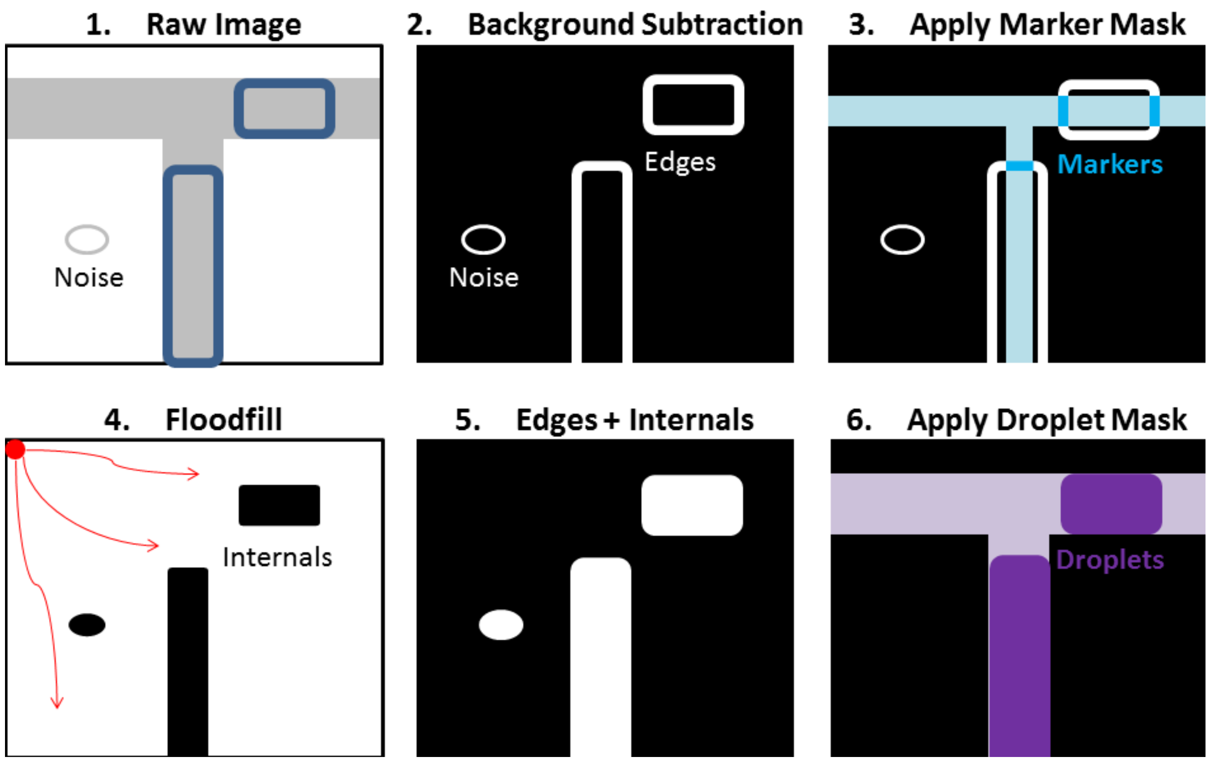


Figure 7.8: Object detection

After masks preparation, the image detection begins. The object detection process occurs repeatedly at every global sampling period $T_s = 100ms$, and provides a continuous update of information. The process is demonstrated in figure 7.8.

First, a raw image is acquired from the camera stream, which shows visible droplets captured within the frame. In addition, the raw image might contain noise and artefacts caused by light source fluctuations. In the next step, the raw image is subtracted by the background and thresholded to yield the Edges of the objects. At this point, the Marker Mask is applied to obtain the Markers, which contain droplet displacements information. Meanwhile, the Edge image is floodfilled from the corner, in order to isolate the object Internals. The object Internals are combined with object Externals to reconstruct intact objects. Lastly, the Droplet Mask is applied to exclude noise, and the remaining objects are stored as individual Droplets.

It is not enough to simply detect droplets. To calculate an object's displacement, its positions at the current instance $t = kT$ and the previous instance $t = (k-1)T$ must be known. This implies that each object must have an identity that is unchanged from time to time.

Figure 7.9 shows a simplified scenario, where droplets enter and exit a straight channel. The same channel is shown at two instances $t = (k-1)T$ and $t = kT$. The following algorithm is created to track the droplet identities.

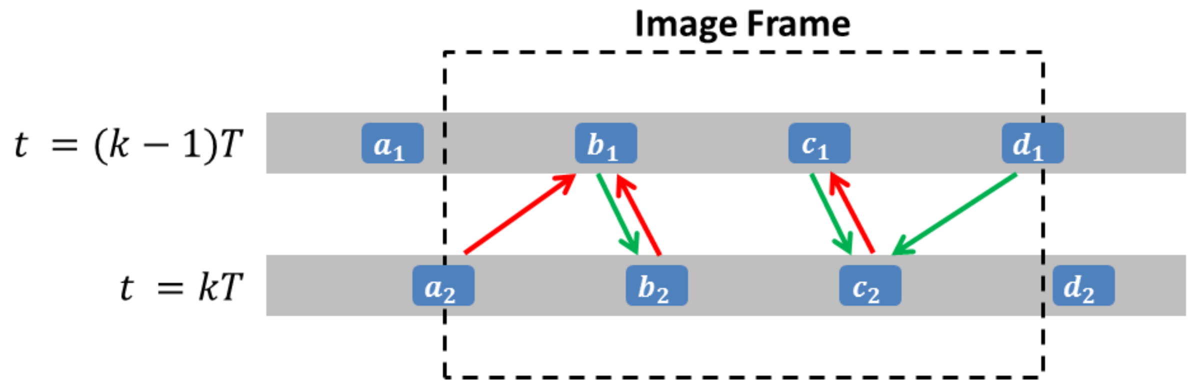


Figure 7.9: Object tracking

First, a ”new-to-old L_2 norm matrix” is formed, describing the distances from each current droplet to each previous droplet. The ”new-to-old L_2 norm matrix” would have size 3×3 as there are 3 visible droplets in both instances. By locating the minimum in each column of the ”new-to-old L_2 norm matrix”, each new droplet identifies an old droplet that is most likely to be itself. These preferences are shown as red arrows in figure 7.9.

Later, an ”old-to-new L_2 norm matrix” is formed, describing the distances from each previous droplet to each current droplet. By locating the minimum in each column, each old droplet identifies a new droplet that is most likely to be itself. These preferences are shown as green arrows in figure 7.9.

Droplets identities are transferred if a new droplet and an old droplet reach consensus that they are the same. For example, droplet c_1 and droplet c_2 are identified as the same droplet as they both prefer each other. In contrast, droplet a_2 prefers b_1, while b_1 prefers b_2, hence, they are not the same.

7.2.2 Neck Detection

For reasons that will be disclosed later, it is important to detect neck distances from droplets that are being generated. Figure 7.10 shows a screen-shot of RoboDrop detecting the neck distance.

The detection process has two phrases. First, it identifies a kink in the droplet, which is achieved by calculating the convex hull of a given droplet. Convex hull is a polygon with the least vertices that is still capable of enveloping an object. Once the convex hull is generated, the kink is identified as the point on the droplet contour, at which convex defect is the greatest.

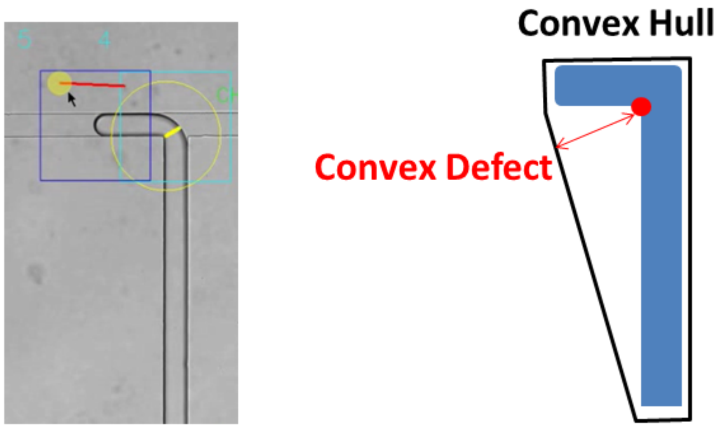


Figure 7.10: Neck detection

From the kink, L_2 norms are calculated around the droplet contour. Figure 7.11 shows the L_2 norm vs. contour index profile. In the scenario shown below, the profile's local minimum corresponds to the neck distance.

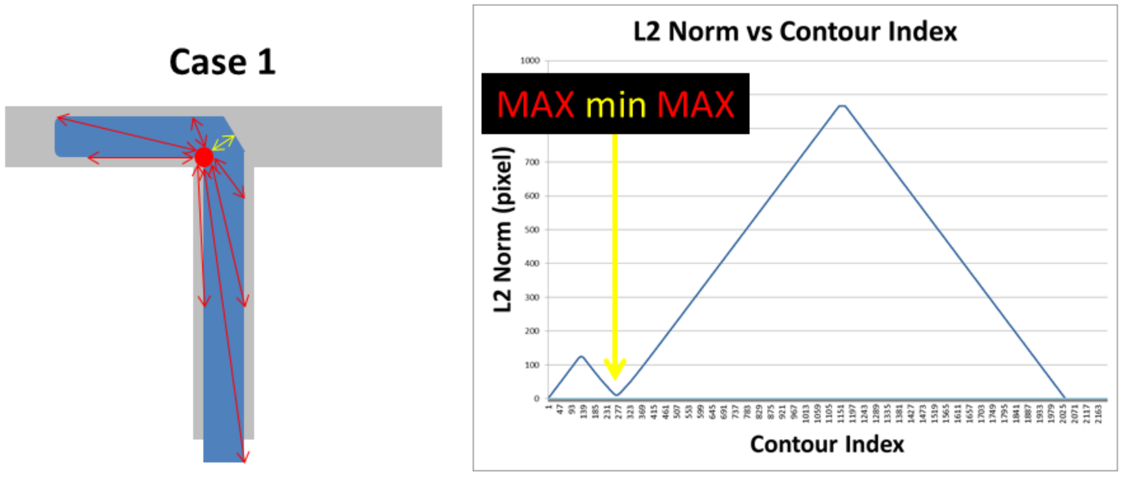


Figure 7.11: Neck distance: case 1

During droplet generation, the droplet might also overflow into the T-junction as shown in figure 7.12. In this case, the L_2 norms are again calculated, and the neck distance is found by locating the 2nd maximum in the profile.

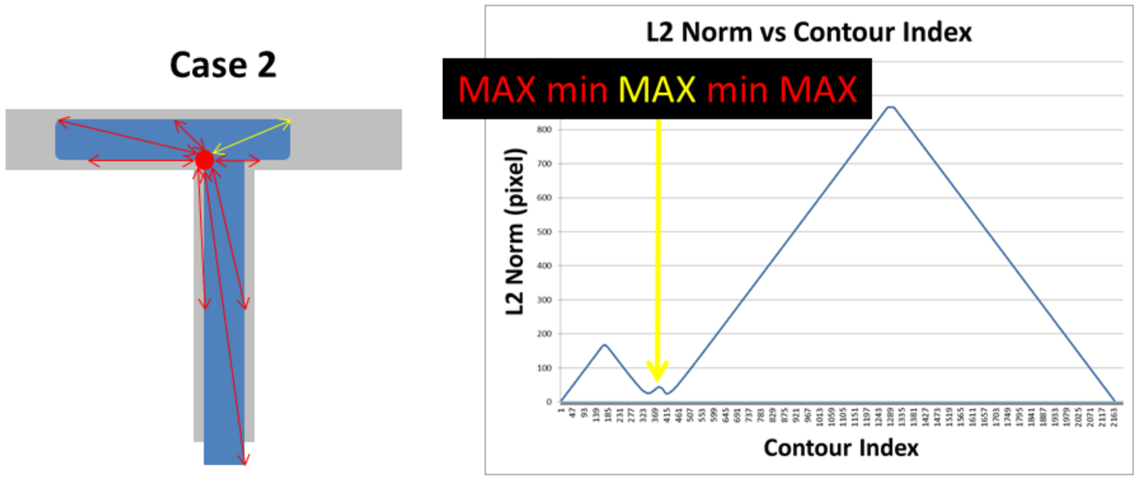


Figure 7.12: Neck distance: case 2

7.2.3 Decision Trees

Previous discussions explained the processes in which, Droplets, Markers,and neck distances are detected. To utilize these information as controllers' inputs, RoboDrop has three decision making processes. The first decision tree is shown in figure 7.13. It describes how RoboDrop discards selected Markers and neck feed-back Droplets that are no longer relevant.

The second decision tree is shown in figure 7.14, in which users are allowed to select or discard a Marker, triggering a controller switch in the process.

Lastly, when users are busy, or when in-coming droplets are moving too fast to be caught by a human, automatic selection assistance is provided according to figure 7.15.

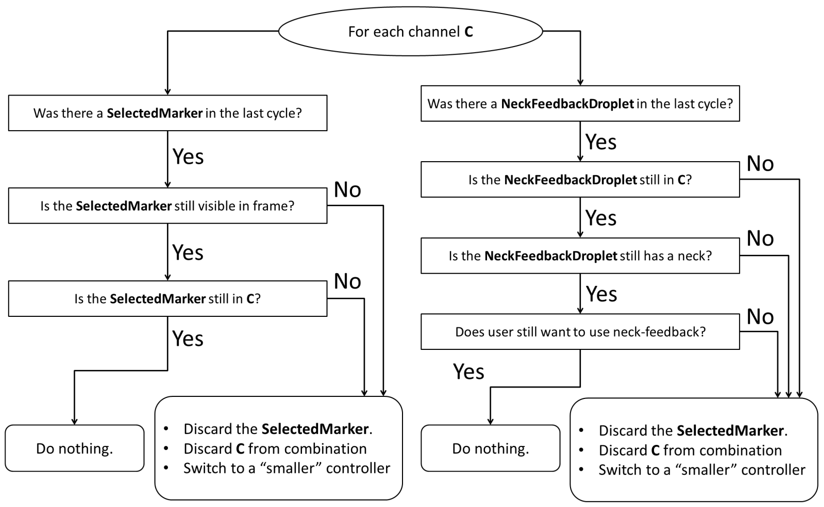

For each channel C
Was there a SelectedMarker in the last cycle?
Yes
Is the SelectedMarker still visible in frame?
No
Yes
Is the SelectedMarker still in C?
No
Yes
Do nothing.
Discard the SelectedMarker.
Discard C from combination
Switch to a "smaller" controller
Was there a NeckFeedbackDroplet in the last cycle?
Yes
No
Is the NeckFeedbackDroplet still in C?
Yes
No
Is the NeckFeedbackDroplet still has a neck?
Yes
No
Does user still want to use neck-feedback?
Yes
Do nothing.
Discard the SelectedMarker.
Discard C from combination
Switch to a "smaller" controller

Figure 7.13: Decision tree: discard

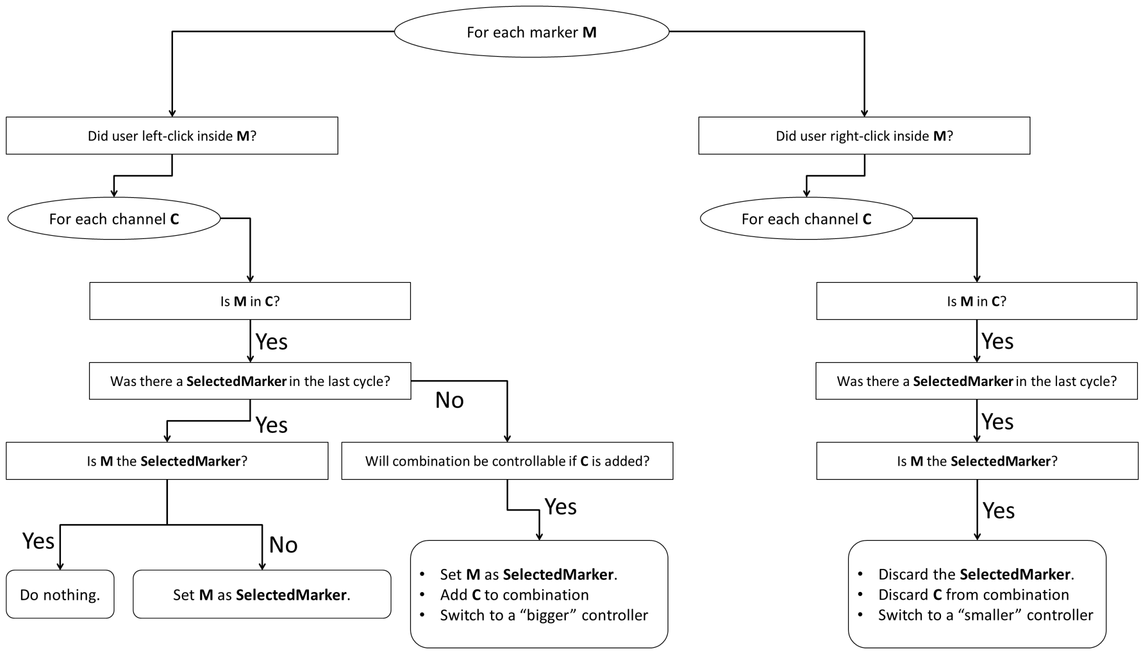

For each marker **M**
Did user left-click inside **M**?
Did user right-click inside **M**?
For each channel **C**
For each channel **C**
Is **M** in **C**?
Is **M** in **C**?
Yes
Yes
Was there a **SelectedMarker** in the last cycle?
Was there a **SelectedMarker** in the last cycle?
No
Yes
Yes
Is **M** the **SelectedMarker**?
Will combination be controllable if **C** is added?
Is **M** the **SelectedMarker**?
Yes
Yes
Yes
No
Do nothing.
Set **M** as **SelectedMarker**.
Set **M** as **SelectedMarker**. Add **C** to combination. Switch to a "bigger" controller
Discard the **SelectedMarker**. Discard **C** from combination. Switch to a "smaller" controller

Figure 7.14: Decision tree: user interactions

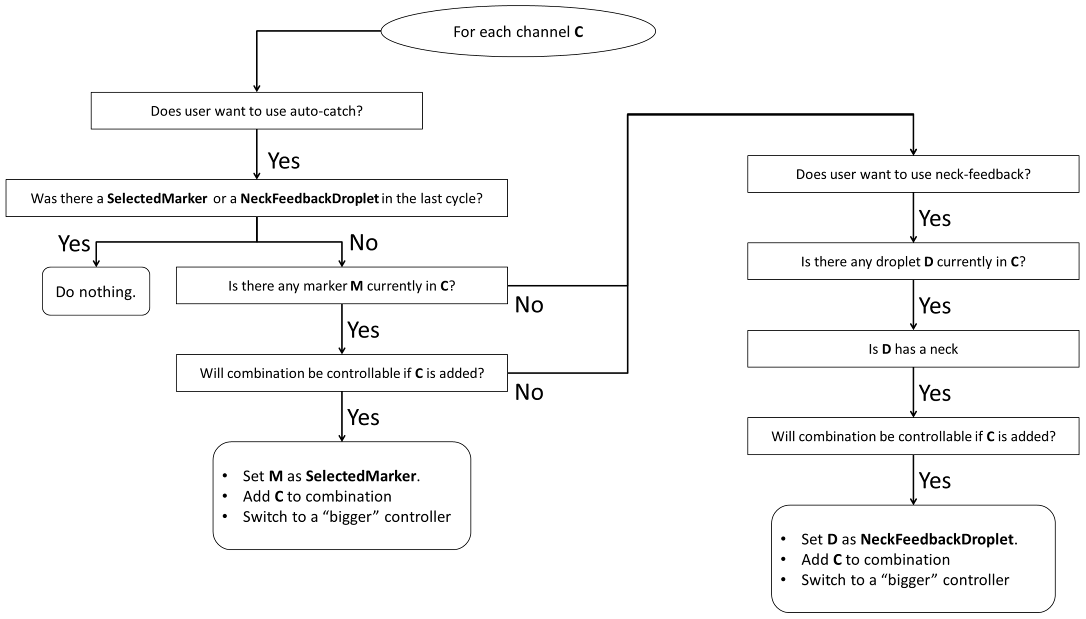

For each channel **C**
Does user want to use auto-catch?
Yes
Was there a **SelectedMarker** or a **NeckFeedbackDroplet** in the last cycle?
Yes
No
Do nothing.
Is there any marker **M** currently in **C**?
No
Yes
Will combination be controllable if **C** is added?
No
Yes
Set **M** as **SelectedMarker**. Add **C** to combination. Switch to a "bigger" controller
Does user want to use neck-feedback?
Yes
Is there any droplet **D** currently in **C**?
Yes
Is **D** has a neck
Yes
Will combination be controllable if **C** is added?
Yes
Set **D** as **NeckFeedbackDroplet**. Add **C** to combination. Switch to a "bigger" controller

Figure 7.15: Decision tree: automatic selection

7.3 RoboDrop Controller

A typical microfluidic channel network has many degree of freedoms (DOF). As discussed in chapter 6, droplets correspondingg to each DOF can be moved independent of each other. Yet, in RoboDrop, the user moves droplets with only a single device - the mouse.

7.3.1 References Linking

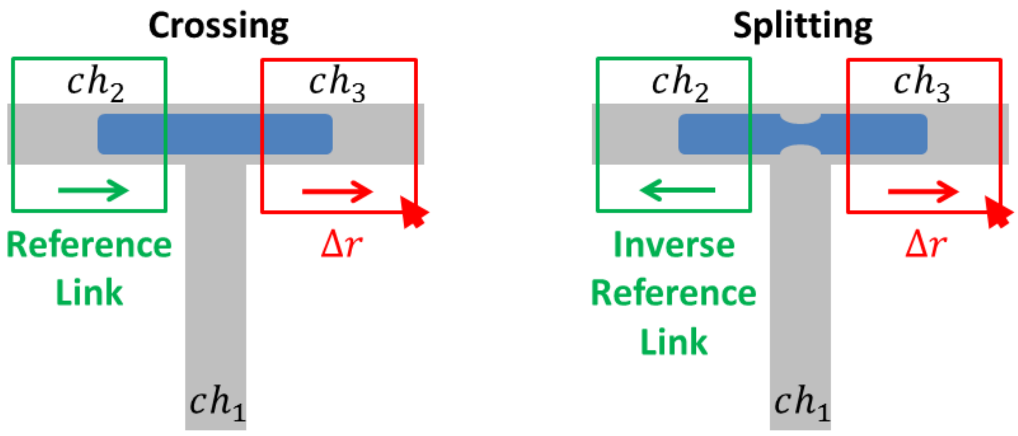


Figure 7.16: Reference linking

To fully utilize the potential of RoboDrop, signal from the mouse has to be shared between multiple channels, and this is achieved through reference linking.

Figure 7.16 shows a T-junction with 2 DOF. Without reference linking, the user can only move ch_3 or ch_2 one at a time. If a droplet happens to reside in both channels, it would be difficult to move across.

When reference linking is enabled through keyboard short-cuts, the user mouse movement $\triangle r$ is applied not only to ch_3, but ch_2 as well. This allows the droplet to cross the junction and remain intact. The reference linking can also be inverted, in which case, when the user drags on ch_3, an inverted reference $- \triangle r$ is applied to ch_2, forcing the droplet to split.

7.3.2 Necking Feedback

During droplet generation, water displacement within the supply channel is a critical feedback parameter. Yet, when water floods the entire supply channel, there is nothing for the image detection to observe, since movement is no longer visible without an interface.

This problem is solved by substituting neck distance as a feedback parameter. Figure 7.17 shows a T-junction where ch_1 is the flooded supply channel. While ch_2 displacement y_2 is measured by image detection. y_1 is not observable.

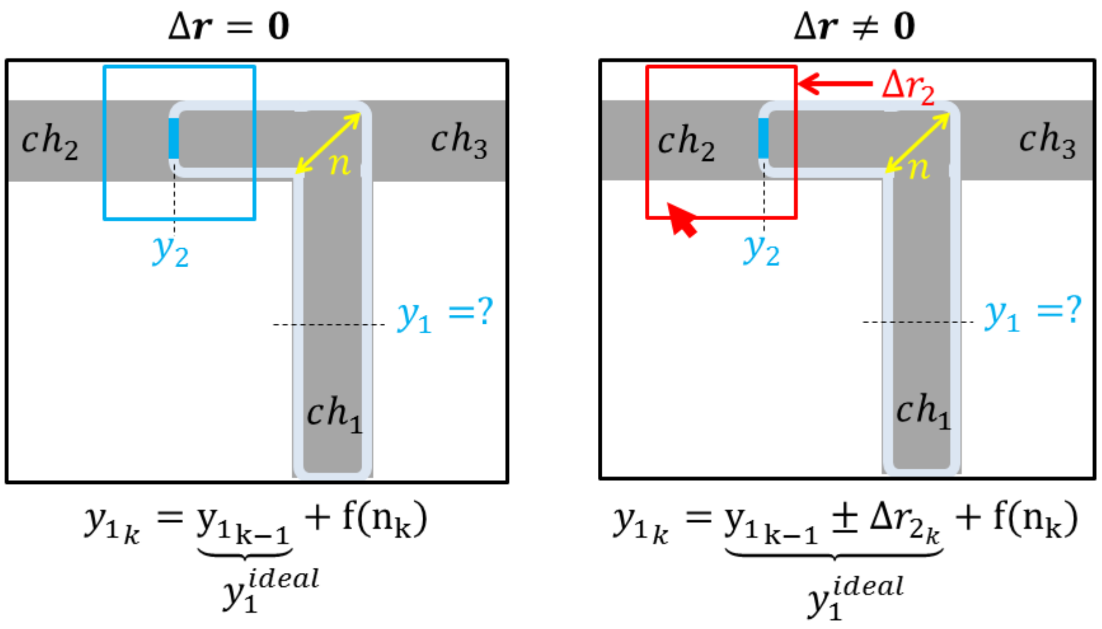

$\Delta r = 0$
ch_2
ch_3
n
y_2
$y_1 = ?$
ch_1
$y_{1_k} = \underbrace{y_{1_{k-1}}}_{y_1^{ideal}} + f(n_k)$
$\Delta r \neq 0$
Δr_2
ch_2
ch_3
n
y_2
$y_1 = ?$
ch_1
$y_{1_k} = \underbrace{y_{1_{k-1}} \pm \Delta r_{2_k}}_{y_1^{ideal}} + f(n_k)$

Figure 7.17: Neck distance as feedback

When user input is absent $\triangle r = 0$, the controllers will attempt to hold $y1$ stationary, and the ideal displacement $y1^{ideal}$ is simply its previous position $y1_{k-1}$. To reconstruct $y1_k$, a non linear neck distance function $f(n_k)$ is added to $y1^{ideal}$. The controllers will then monitor the droplet neck and adjust flow rate in ch_1 based on n.

When user lengthens or shortens the droplet, the ideal displacement becomes $y1^{ideal} = y1_{k-1} \pm \triangle r2_k$, depending on channel directions. ch_1 displacement is once again reconstructed from $y1^{ideal}$ and $f(n_k)$, allowing the controllers to maintain a targeted neck distance while moving ch_2.

7.3.3 Measurement Offset

Multiple droplets might reside within the same channel, and RoboDrop allows users to choose which Marker to be used as controller measurement. When the user selects a different Marker, or when a different channel combination is activated, the absolute Marker position y^{abs} jumps discontinuously even when there is no real movement in the channel.

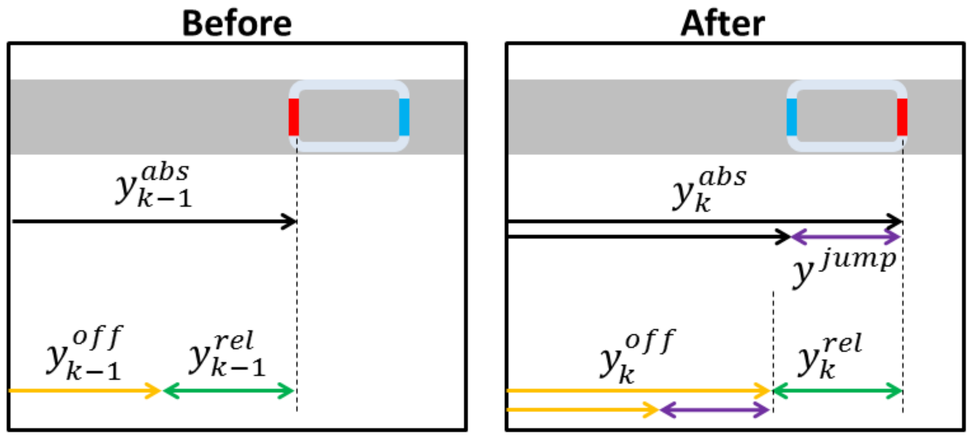


Figure 7.18: Measurement extraction

This problem is accounted for by introducing the displacement offset variable y^{off}. Figure 7.18 shows that before a switch, y_{k-1}^{off} contains the Marker's initial position, and the Marker relative displacement y_{k-1}^{rel} is calculated as $y_{k-1}^{abs} - y_{k-1}^{off}$. After the switch, y^{jump} is calculated from $y_k^{abs} - y_{k-1}^{abs}$ and added to the offset variable y_k^{off}. The relative displacement y_k^{rel} is then calculated from the updated offset variable $y_k^{abs} - y_k^{off}$ to maintain its continuity.

7.3.4 Controller Calculations

Finally, table 7.1 contains all the controller variables and corresponding equations. These variables are updated at every $T_s = 100ms$, following the order that they are listed.

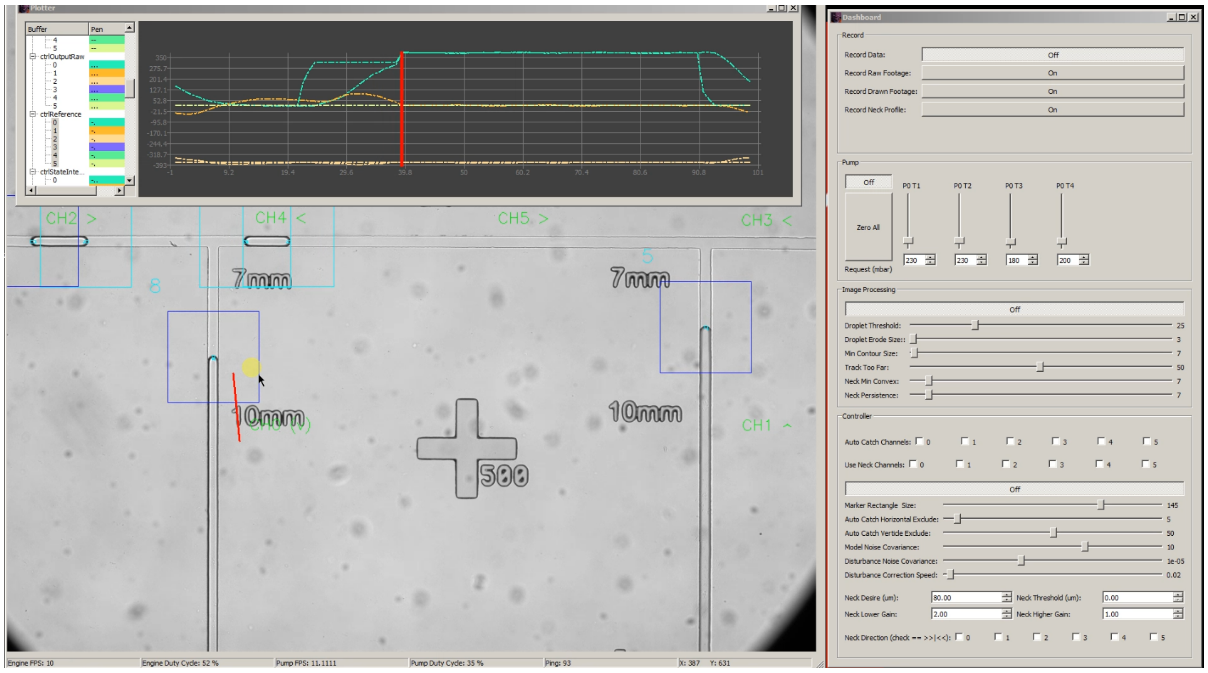
Error Covariance Matrix (States Estimation)	Pe	$Pe_{k-1} = ([1] - K_{k-1}C_l)Pp_{k-1}$
Error Covariance Matrix (States Predictions)	Pp	$Pp_k = A_l Pe_{k-1} A_l^T + W_l R_w W_l^T$
Kalman Gain Matrix	K	$K_k = Pp_k C_l^T (C_l Pp_k C_l^T + R_v)^{-1}$
States (Kalman Prediction)	xp	$xp_k = A_l xe_{k-1} + B_l u_{k-1}$
Output (Kalman Prediction)	yp	$yp_k = C_l xp_k$
States (Kalman Estimation)	xe	$xe_k = xp_k + K_k(y_k - yp_k)$
Disturbances (Kalman Estimation)	de	$de_k = \text{subset}\{xe_k\}$
Outputs (Luenburger Prediction)	yl	$yl_k = C_r xl_{k-1} + D_r u_{k-1}$
States (Luenburger Estimation)	xl	$xl_k = A_r xl_{k-1} + B_r u_{k-1} + H(y_{k-1} - yl_{k-1})$
Integral States	z	$z_k = z_{k-1} + T(y_k - r_k)$
Commands	u	$u_k = -K_1 xl_k - K_2 z_k$
Grounds	g	$g = i_0$
Disturbance Corrections	dc	$dc_k = dc_{k-1} + K_3 T de_k$
Inlet Pressures	i	$i_k = g + c_k + u_k$

Table 7.1: Controller calculations

Chapter 8

Results

Figure 8.1 shows RoboDrop in action. The chip used during experiment had 4 inlets and 5 channels. The 5 channels were further divided in 6 segments labelled $ch_0 \rightarrow ch_5$ for convenience during droplet generation. ch_0 and ch_1 contained water, while the rest contained oil. At the moment when figure 8.1 was captured, there were two droplets, one in ch_2, the other in ch_4. At that instance, the markers in ch_0, ch_1, ch_2 were coloured blue, while the rest were coloured cyan. Blue indicates that those were the selected markers, and ch_0, ch_1, ch_2 were the activated channels. Displacement data for these markers were recorded, and shown in figure 8.2.



Record

Record Data: Off
Record Raw Footage: On
Record Drawn Footage: On
Record Neck Profile: On

Pump

Off

Zero All

Request (mbar)

P0 T1: 230
P0 T2: 230
P0 T3: 180
P0 T4: 200

Image Processing

Off

Droplet Threshold: 25
Droplet Erode Size:: 3
Min Contour Size: 7
Track Too Far: 50
Neck Min Convex: 7
Neck Persistence: 7

Controller

Auto Catch Channels: ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5
Use Neck Channels: ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Off

Marker Rectangle Size: 145
Auto Catch Horizontal Exclude: 5
Auto Catch Verticle Exclude: 50
Model Noise Covariance: 10
Disturbance Noise Covariance: 1e-05
Disturbance Correction Speed: 0.02

Neck Desire (um): 80.00
Neck Threshold (um): 0.00
Neck Lower Gain: 2.00
Neck Higher Gain: 1.00

Neck Direction (check == >>|<<): ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Engine FPS: 10 | Engine Duty Cycle: 52 % | Pump FPS: 11.1111 | Pump Duty Cycle: 35 % | Ping: 93 | X: 387 Y: 631

Figure 8.1: Screen-shot: ch_0, ch_1, ch_2 active

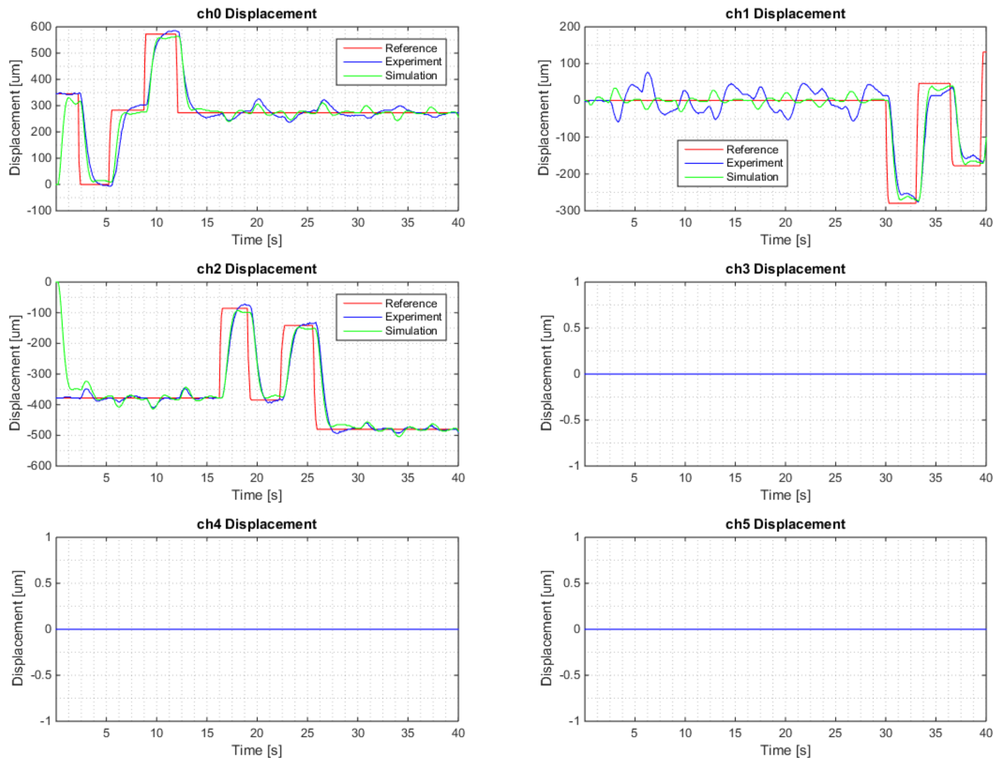
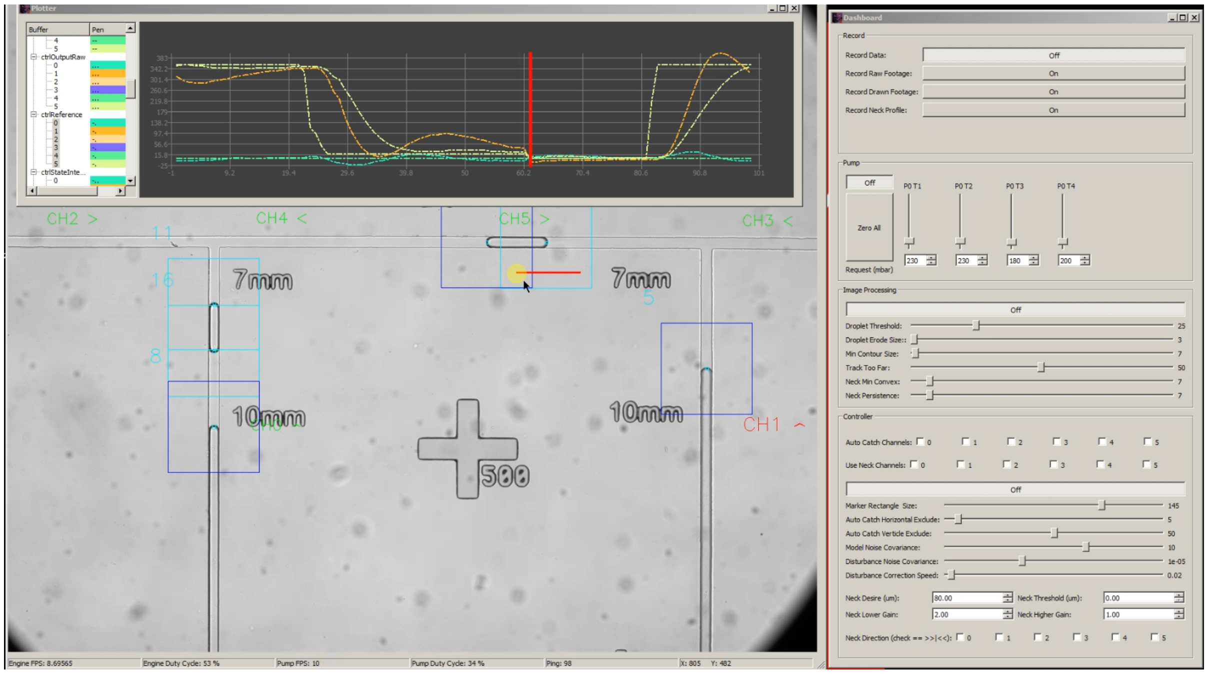


Figure 8.2: Closed-loop data: ch_0, ch_1, ch_2 active

From $t = 0$ to $t = 12s$, the water interface in ch_0 was moved up and down. From $t = 13s$ to $t = 28s$, the water droplet in ch_2 was moved left and right. From $t = 28s$ onwards, the second water interface in ch_1 was moved. The coupling responses were minimal, and the experimental closed-loop responses roughly followed the simulated responses.

In the next screen-shot, figure 8.3, ch_5 instead of ch_2 was activated, and the user can be seen dragging a marker in ch_5 to the left. Noticed that the ch_1 label had turned from green to red. This indicates that reference linking was taking place.

The corresponding displacement data is shown in figure 8.4. As expected, the references for both ch_1 and ch_5 were the same due to reference linking. This resulted in the ch_1 interface and ch_5 droplet moving simultaneously. Note that a jump occurred at $t = 56.8s$. This was caused by the user switching markers within ch_5. During the transition, controllers were unaffected, and closed-loop response remained smooth.



Record

Record Data: Off
Record Raw Footage: On
Record Drawn Footage: On
Record Neck Profile: On

Pump

Off
Zero All
Request (mbar)

P0 T1	P0 T2	P0 T3	P0 T4
230	230	180	200

Image Processing

Off
Droplet Threshold: 25
Droplet Erode Size:: 3
Min Contour Size: 7
Track Too Far: 50
Neck Min Convex: 7
Neck Persistence: 7

Controller

Auto Catch Channels: ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5
Use Neck Channels: ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Off
Marker Rectangle Size: 145
Auto Catch Horizontal Exclude: 5
Auto Catch Verticle Exclude: 50
Model Noise Covariance: 10
Disturbance Noise Covariance: 1e-05
Disturbance Correction Speed: 0.02

Neck Desire (um): 80.00
Neck Threshold (um): 0.00
Neck Lower Gain: 2.00
Neck Higher Gain: 1.00

Neck Direction (check == >>|<<): ☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Engine FPS: 8.69565 Engine Duty Cycle: 53 % Pump FPS: 10 Pump Duty Cycle: 34 % Ping: 98 X: 805 Y: 482

Figure 8.3: Screen-shot: ch_0, ch_1, ch_5 active

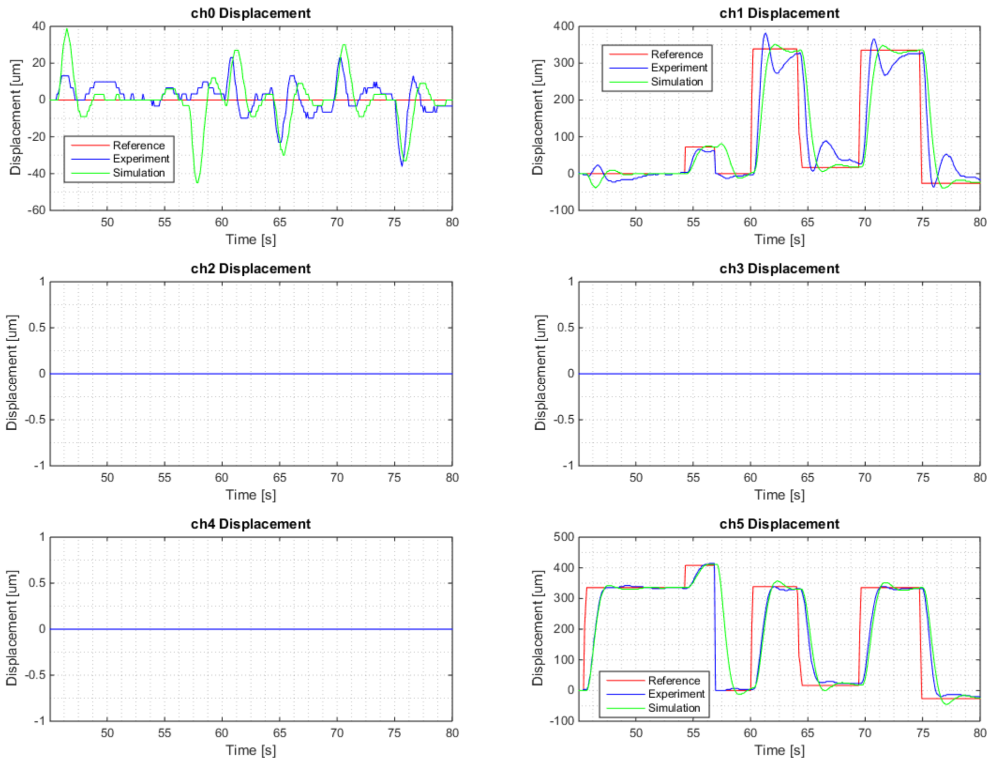


Figure 8.4: Closed-loop data: ch_0, ch_1, ch_5 active

8.1 Droplet Generation

Figure 8.5 shows the droplet generation process using neck distance as feedback. The user first moved the water interface in ch_1 towards the T-junction. Once the interface had protruded into the junction, the neck distance became regulated by the controller. The user then linked the references between ch_1 and ch_5 to adjust droplet length. Once the desired droplet length was achieved, reference linking was inverted, and the droplet was split by moving ch_5 in the opposite direction against ch_1. Later on, a similar process was repeated and a droplet was generated from ch_0 into ch_4.

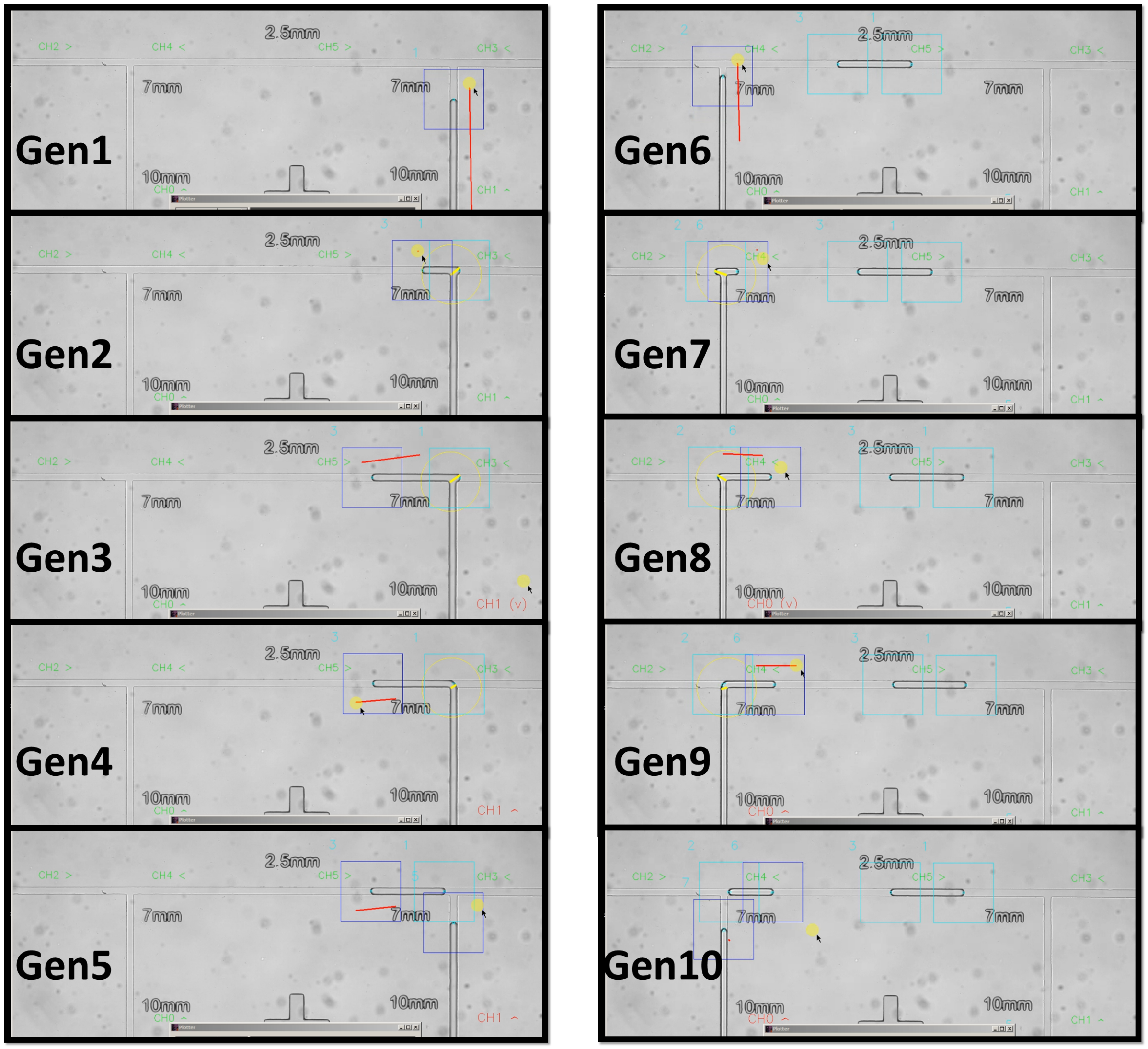


Figure 8.5: Droplet generation

8.2 Droplet Splitting and Merging

Figure 8.6 shows the user merging and splitting droplets.

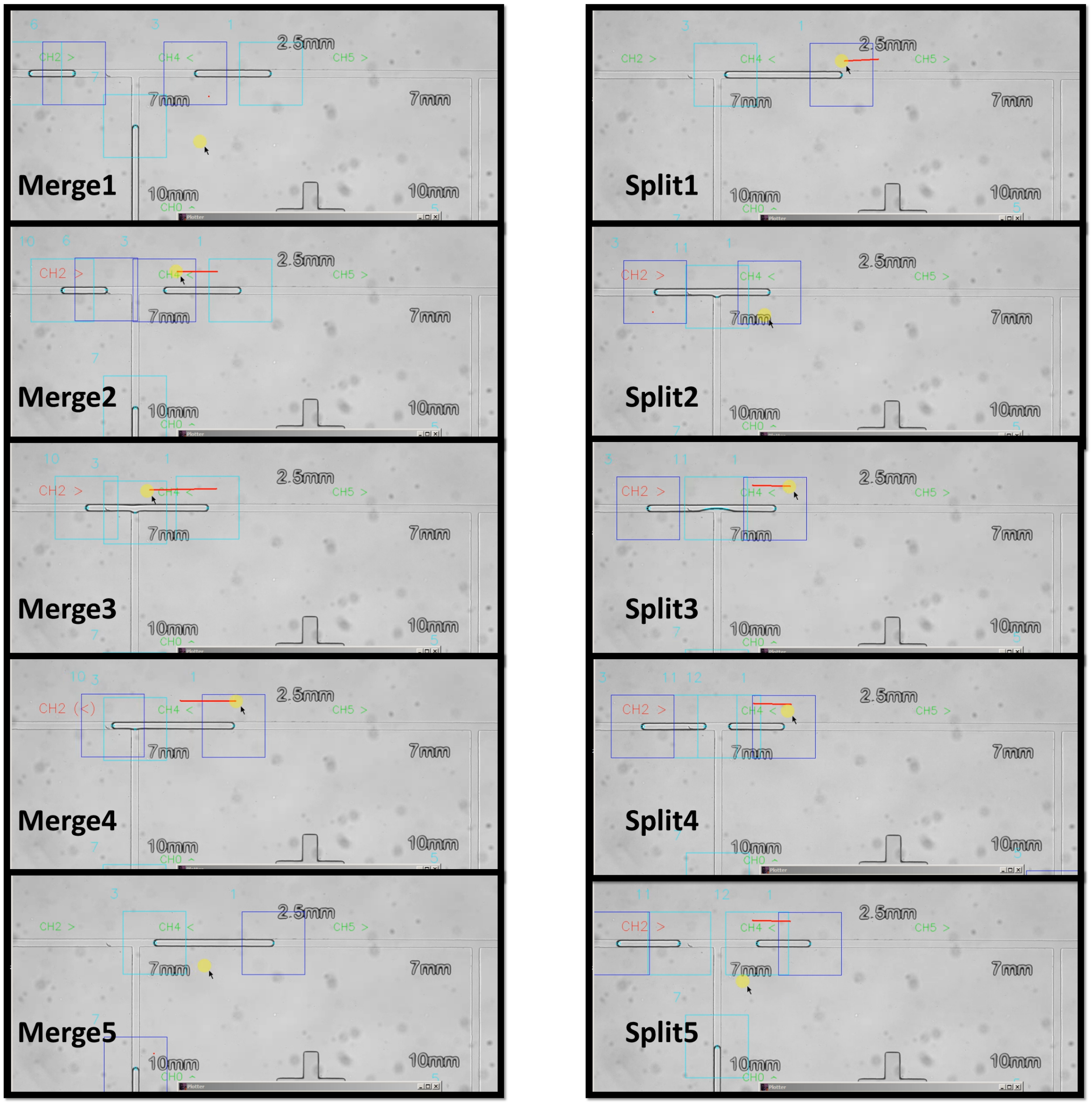


Figure 8.6: Merging and splitting

8.3 Droplet Sorting and Retrieving

Lastly, the ability to isolate an individual droplet is demonstrated. In figure 8.7, the experiment started with two droplets, one in ch_2, the other in ch_4. The ch_2 droplet was moved into ch_0. Then, the ch_4 droplet was moved across junction and pushed out-of-screen from ch_3. Finally, the droplet stored in ch_0 was retrieved.

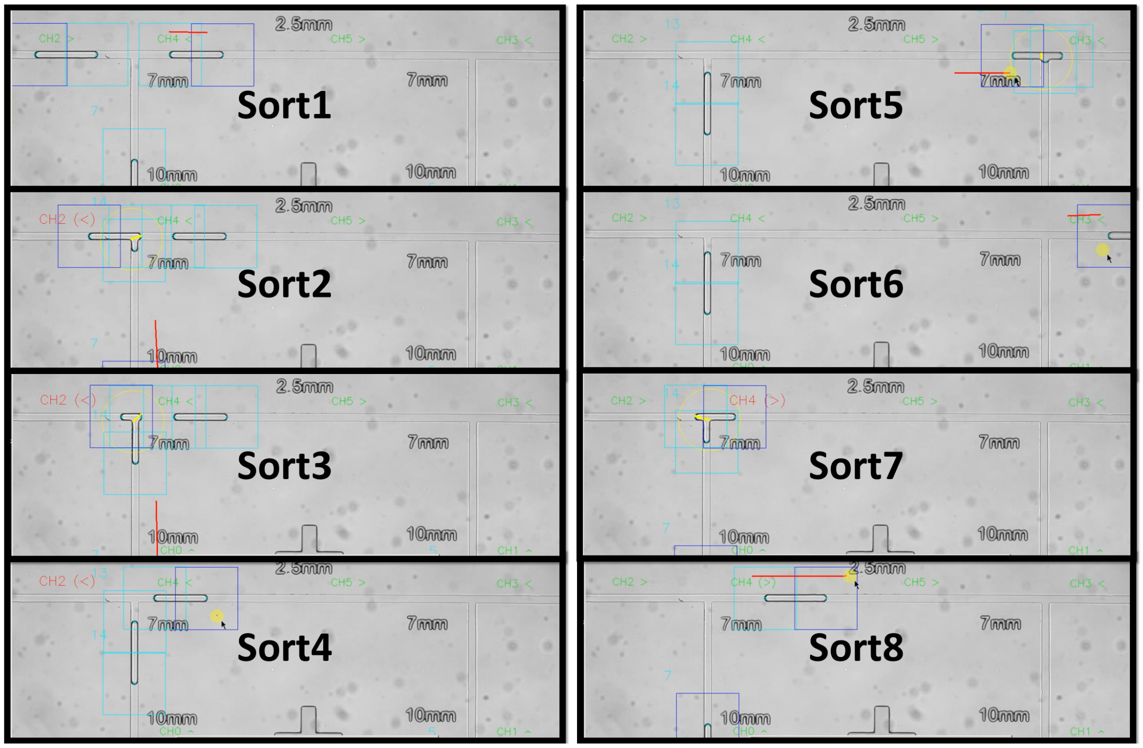


Figure 8.7: Sorting and retrieving

Chapter 9

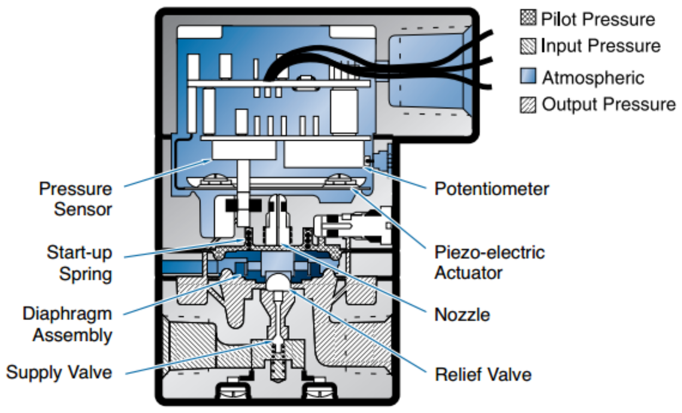
Conclusion

In conclusion, an approach to model droplet movements under pressure actuation is proposed. The model comprises of state-space matrices that represent fluid dynamics in a channel network, and can predicts droplet displacements when they are in the squeezing regime. The model is validated by comparing experimental data to open-loop simulations. Most importantly, the model building process requires minimal human intervention. By taking advantage of the electric circuit analogy, the model is also capable of scaling-up to represent any arbitrary channel networks. Control law is then designed to stabilize droplet movements, and to de-couple the system such that individual droplets can be moved independent of each other. The control law contains state feedbacks, integral feedbacks, a Luenburger observer for states estimation, and a Kalman filter for disturbance cancelling. Optimal controller gains are obtained using LQR, while attention is dedicated to maintain a smooth transition when switching from one controller to another. Once again, the controller design process is fully automated and universal. Lastly, a C++ program is created for implementing the controllers. The program detects droplet positions with computer vision, and allows users to move the droplets by clicking and dragging them in an augmented video stream. With this program, the ability to generate, move, split, merge, and sort droplets is demonstrated.

9.1 Hardware Recommendations

The hardware put together in this study is far from ideal. A few pieces of proposed equipment are shown below. Figure 9.1a shows a pocket CO2 canister commonly used by cyclist to re-inflate tires during field repairs. Figure 9.1b shows a commercially available EP transducer. A GoPro action camera is shown in Figure 9.1c, in which the camera has been modified to mount custom lenses, such as the ones shown in figure 9.1d.

(a) CO2 canister (ebay.com)



(b) EP transducer (controlair.com)

(c) goPro camera (back-bone.ca)

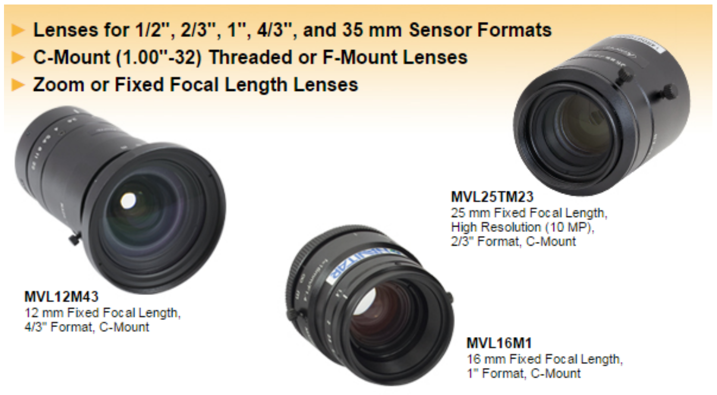

Lenses for 1/2", 2/3", 1", 4/3", and 35 mm Sensor Formats
C-Mount (1.00"-32) Threaded or F-Mount Lenses
Zoom or Fixed Focal Length Lenses
MVL25TM23
25 mm Fixed Focal Length, High Resolution (10 MP), 2/3" Format, C-Mount
MVL12M43
12 mm Fixed Focal Length, 4/3" Format, C-Mount
MVL16M1
16 mm Fixed Focal Length, 1" Format, C-Mount
(d) Macro lenses (thorlabs.com)

Figure 9.1: Proposed hardware

These hardware are chosen to improve the system in four categories. The first category is cost. Comparing to scientific grade instruments, they are a lot cheaper, yet offer sufficient or better performance for computer vision and control purposes. The second category is mobility. By replacing the air compressor and the microscope with a CO2 canister and a macro lens, the system can be designed to be portable. Replacing the microscope and scientific camera with an action-camera-macro-lens combo also results in a smaller system footprint, and frees up the microscope for other uses such as monitoring biochemical processes. Lastly, general purpose EP transducers can overcome the $T_s = 100ms$ limitation and speed up the system.

9.2 Controller Recommendations

The current system has a few major limitations. First, the use of background subtraction forbids set up changes during experiment. Not being able to move the microscope stage nor adjusting illumination are huge problems that must be solved, until then the deployment of this technology will be constrained.

The second limitation arises from the camera sensor size. At a given time, only a certain portion of the microfluidic chip can be imaged. Important information that could be used for controller feedback is lost once they are out of frame.

The third limitation concerns with the non-linearity of droplet behaviour near junctions. While the current system does a modest job during droplet generation, a non linear model would better capture surface tension related physics, and yields better performance.

For the above reasons, Artificial Neural Networks (ANN) might be used to replace the current model and controllers. Instead of extracting displacement data through image processing, the ANN would directly accept pixel values as inputs. Instead of having n modelling outputs correspond to n channels, the ANN would have as many outputs as image pixels. Given an image at instance $t = kT$ and the current applied pressures, the ANN would predict which pixels the droplets would occupy in the next instance $t = (k + 1)T$.

First of all, the ANN would have no problem capturing non-linearity. It would simply relate droplet behaviour to the channel topology, and provide location-dependent predictions. Furthermore, using methods similar to convolution neural nets, the ANN would function regardless of image zoom magnification and stage position. Lastly, the ANN approach can be expanded directly to monitor the entire microfluidic chip, either through laser scanning or opto-fluidic technologies.

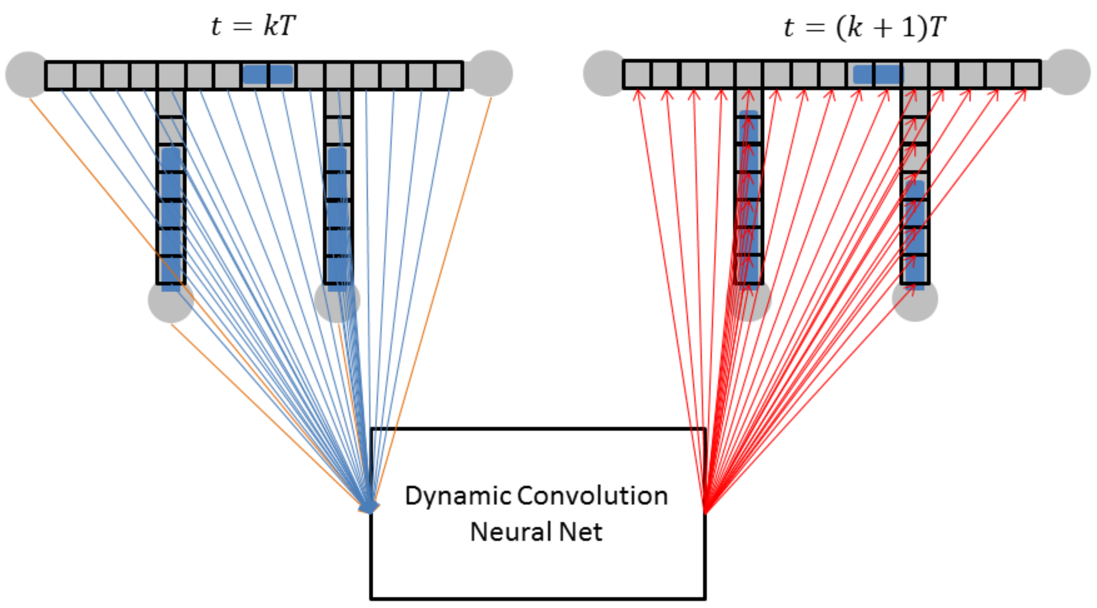


Figure 9.2: Proposed artificial neural network

Bibliography

[1] Piotr Garstecki, Michael J Fuerstman, Howard A Stone, and George M Whitesides. Formation of droplets and bubbles in a microfluidic t-junctionscaling and mechanism of break-up. *Lab on a Chip*, 6(3):437–446, 2006.

[2] Xize Niu, Shelly Gulati, Joshua B Edel, et al. Pillar-induced droplet merging in microfluidic circuits. *Lab on a chip*, 8(11):1837–1841, 2008.

[3] Ansgar Huebner, Dan Bratton, Graeme Whyte, Min Yang, Chris Abell, Florian Hollfelder, et al. Static microdroplet arrays: a microfluidic device for droplet trapping, incubation and release for enzymatic and cell-based assays. *Lab on a Chip*, 9(5):692–698, 2009.

[4] YongTae Kim, Philip LeDuc, and Willam Messner. Modeling and control of a nonlinear mechanism for high performance microfluidic systems. *Control Systems Technology, IEEE Transactions on*, 21(1):203–211, 2013.

[5] Erik Miller, Mario Rotea, and Jonathan P Rothstein. Microfluidic device incorporating closed loop feedback control for uniform and tunable production of micro-droplets. *Lab on a Chip*, 10(10):1293–1301, 2010.

[6] Xize Niu, Mengying Zhang, Suili Peng, Weijia Wen, and Ping Sheng. Real-time detection, control, and sorting of microfluidic droplets. *Biomicrofluidics*, 1(4):044101, 2007.

[7] Michael D Armani, Satej V Chaudhary, Roland Probst, and Benjamin Shapiro. Using feedback control of microflows to independently steer multiple particles. *Microelectromechanical Systems, Journal of*, 15(4):945–956, 2006.

[8] Slawomir Jakiela, Tomasz S Kaminski, Olgierd Cybulski, Douglas B Weibel, and Piotr Garstecki. Bacterial growth and adaptation in microdroplet chemostats. *Angewandte Chemie*, 125(34):9076–9079, 2013.

[9] Julia Khandurina, Timothy E McKnight, Stephen C Jacobson, Larry C Waters, Robert S Foote, and J Michael Ramsey. Integrated system for rapid pcr-based dna analysis in microfluidic devices. *Analytical Chemistry*, 72(13):2995–3000, 2000.

[10] Chung-Cheng Lee, Guodong Sui, Arkadij Elizarov, Chengyi Jenny Shu, Young-Shik Shin, Alek N Dooley, Jiang Huang, Antoine Daridon, Paul Wyatt, David Stout, et al. Multistep synthesis of a radiolabeled imaging probe using integrated microfluidics. *Science*, 310(5755):1793–1796, 2005.

[11] Timothy A Crowley and Vincent Pizziconi. Isolation of plasma from whole blood using planar microfilters for lab-on-a-chip applications. *Lab on a Chip*, 5(9):922–929, 2005.

[12] Henk Wensink, Fernando Benito-Lopez, Dorothee C Hermes, Willem Verboom, Han JGE Gardeniers, David N Reinhoudt, and Albert van den Berg. Measuring reaction kinetics in a lab-on-a-chip by microcoil nmr. *Lab on a Chip*, 5(3):280–284, 2005.

[13] Tohid Fatanat Didar and Maryam Tabrizian. Adhesion based detection, sorting and enrichment of cells in microfluidic lab-on-chip devices. *Lab on a Chip*, 10(22):3043–3053, 2010.

[14] Mira T Guo, Assaf Rotem, John A Heyman, and David A Weitz. Droplet microfluidics for high-throughput biological assays. *Lab on a Chip*, 12(12):2146–2155, 2012.

[15] Jenifer Clausell-Tormos, Diana Lieber, Jean-Christophe Baret, Abdeslam El-Harrak, Oliver J Miller, Lucas Frenz, Joshua Blouwolff, Katherine J Humphry, Sarah Köster, Honey Duan, et al. Droplet-based microfluidic platforms for the encapsulation and screening of mammalian cells and multicellular organisms. *Chemistry & biology*, 15(5):427–437, 2008.

[16] Linfen Yu, Michael CW Chen, and Karen C Cheung. Droplet-based microfluidic system for multicellular tumor spheroid formation and anticancer drug testing. *Lab on a Chip*, 10(18):2424–2432, 2010.

[17] Helen Song, Delai L Chen, and Rustem F Ismagilov. Reactions in droplets in microfluidic channels. *Angewandte chemie international edition*, 45(44):7336–7356, 2006.

[18] MG Pollack, AD Shenderov, and RB Fair. Electrowetting-based actuation of droplets for integrated microfluidics. *Lab on a Chip*, 2(2):96–101, 2002.

[19] Elizabeth A Ottesen, Jong Wook Hong, Stephen R Quake, and Jared R Leadbetter. Microfluidic digital pcr enables multigene analysis of individual environmental bacteria. *science*, 314(5804):1464–1467, 2006.

[20] Brandon Kuczenski, Philip R LeDuc, and William C Messner. Pressure-driven spatiotemporal control of the laminar flow interface in a microfluidic network. *Lab on a Chip*, 7(5):647–649, 2007.

[21] Wen Zeng, Songjing Li, and Zuwen Wang. Closed-loop feedback control of droplet formation in a t-junction microdroplet generator. *Sensors and Actuators A: Physical*, 233:542–547, 2015.

[22] Anish Shenoy, Melikhan Tanyeri, and Charles M Schroeder. Characterizing the performance of the hydrodynamic trap using a control-based approach. *Microfluidics and Nanofluidics*, 18(5-6):1055–1066, 2015.

[23] Nikon. *Inverted Microscope ECLIPSE Ti-E Ti-E/B Instructions*. Nikon, 2010.

[24] Dong Qin, Younan Xia, and George M Whitesides. Soft lithography for micro-and nanoscale patterning. *Nature protocols*, 5(3):491–502, 2010.

[25] Chris Paige. Properties of numerical algorithms related to computing controllability. *IEEE Transactions on Automatic Control*, 26(1):130–138, 1981.

Appendix A

Matlab Script

A.1 Pump System Identification

Matlab scripts and pump test data resides on:
`https://github.com/DaveSketchySpeedway/uEVA/tree/master/pump_sysid`

A.2 Modelling and Controller Design

Matlab scripts and Simulink models are located at:
`https://github.com/DaveSketchySpeedway/uEVA/tree/master/model_ctrl`

Appendix B

C++ Source Code

B.1 RoboDrop

The source code for RoboDrop and instruction for compilation is hosted on GitHub:
`https://github.com/DaveSketchySpeedway/uEVA/tree/master/RoboDrop02`

Appendix C

Experiment video with Single T-Junction

C.1 Video Clip

This appendix is a video file. The file name of this video file is “robodrop1t.avi”.

The footage is a screen-capture showcasing the RoboDrop software in operation. Microscope images were processed in real time to detect droplets and interfaces displacements. The processed images were augmented with the displacement information and displayed in real time. Furthermore, the controllers were activated in RoboDrop, allowing droplets to be manipulated arbitrarily by the user.

If you accessed this thesis from a source other than the University of Waterloo, you may not have access to this file. You may access it by searching for this thesis on `https://uwspace.uwaterloo.ca/UWSpace`

‘

Appendix D

Experiment Video with Double T-Junction

D.1 Video Clip

This appendix is a video file. The file name of this video file is “robodrop2thd.avi”.

The footage is a screen-capture showing the RoboDrop software operating in a channel network that has 2 T-junctions. Is shows the user manipulating droplets, performing generation, splitting, merging, and sorting operations. The real-time telemetry feature of RoboDrop was also showcased, where pressure values commanded by the controllers were plotted as the experiment was carried out.

If you accessed this thesis from a source other than the University of Waterloo, you may not have access to this file. You may access it by searching for this thesis on `https://uwspace.uwaterloo.ca/UWSpace`