Modeling and Simulation for Grounding a Mechatronic Test Environment for Inertial Measurement Units

Dan Tappan and Josh Czoski
Department of Computer Science, Eastern Washington University, Cheney, WA, USA

Abstract - *One of the most difficult aspects of learning to play violin is posture. Students practice endlessly in front of a teacher and mirror to ensure correct bow movement with respect to the violin. This work describes a hybrid modeling-and-simulation environment fabricated to evaluate the use of inexpensive inertial measurement units for a larger project to perform such monitoring automatically. It describes a mechatronic test rig that simulates complex bow movement for repeatable, controlled experiments. This physical hardware model introduces its own idiosyncrasies into the evaluation process and in turn requires evaluation by a virtual software model. The combined result is an objective comparative proof-of-concept framework for grounding (calibrating and tuning) the two models against each other and reality to tease out performance characteristics.*

Keywords: violin, data acquisition, inertial measurement unit, mechatronics, performance evaluation

1 Introduction

Learning to play violin is a long and challenging process. One of the greatest difficulties involves developing and maintaining appropriate posture to keep the bow oriented correctly with respect to the violin. During lessons, a teacher closely monitors this activity and indicates whenever there is a problem. Students on their own practice in front of a mirror to monitor themselves. Either way, the process is onerous. An automated system could be much more practical. The recent explosion of popularity in drones, virtual-reality gaming devices, and motion-based smartphone apps has had a profound effect on increasing the capabilities and availability of small, inexpensive inertial measure units (IMUs) that keep track of their position and orientation in three-dimensional space in real time. Attaching one to the violin and another to the bow provides their physical states. In theory, subtracting the two states would produce the relative state of the bow with respect to the violin and facilitate its evaluation within an acceptable range of motion.

In practice, however, this approach morphs into a much larger problem of using the IMUs appropriately and compensating for their many shortcomings. The focus of this paper is on how to evaluate the real-world performance of IMUs objectively for a larger project of this kind. The underlying approach involves conducting repeatable, controlled experiments as part of the scientific method. Directly measuring the IMUs on a real violin and bow exhibits neither property: the tests are not controlled because the violinist cannot be precisely sure of his or her actions, isolate and change them individually, or quantitatively compare them to a standard of correctness; nor are they repeatable because multiple attempts cannot produce the same results or in the same way because humans are not consistent enough.

To mitigate this limitation, a physical model served as a surrogate for the real violin and bow assembly. This mechatronic device combined solutions from computer science, electrical and mechanical engineering, and fabrication with commercial off-the-shelf parts to produce a physical simulation device that was much more amenable to controlled experiments on the real IMUs. However, its idiosyncrasies introduced their own problems, which led to the need to evaluate its own performance. The final result was a virtual model in software that was both convenient for high-speed experiments and arbitrarily accurate. In order to use the virtual model as a surrogate for the physical model, which in turn was a surrogate for the real assembly, the performance characteristics of all three needed to be identified, measured, modeled, tested, analyzed, and validated.

This paper addresses a proof-of-concept integrated environment of modeling, simulation, visualization, and analysis as an objective comparative framework for this heavily underdetermined, messy comparative problem. It involves a wide range of creative *what if* engineering thinking and doing. Specifically, it addresses calibrating and tuning (i.e., grounding) each of the models to each other, executing them, and comparing their performance. It partially uses a Monte Carlo approach to perform sensitivity analysis on the parameters to tease out their independent and interdependent contributions.

2 Background

The larger violin project aimed at determining in real time whether the violinist was manipulating the bow appropriately. It addressed posture and movement only, not how to play correctly in terms of musical notes and style. The definition of acceptable manipulation is complicated and not strictly necessary here. This paper focuses only on establishing the state of the bow and violin in space and

time as correctly and reliably as possible, not on their interrelationships to produce music. For context, however, the purpose of this information was to establish and monitor a complex three-dimensional region of acceptability based on two sources of state data in Figure 1a: the violin, and the left end of the bow opposite the violinist's hand. The violinist has a wide range of freedom in holding the instrument (even upside-down is physically possible); therefore, the acceptable state of the bow is relative to the state of the violin, and both are needed. Conveniently, the same solution applied to both, but for simplicity, the rest of this paper usually refers to the bow only. The bow was also subject to the most movement and demanding tests, so it makes the better representative given the space limitations.

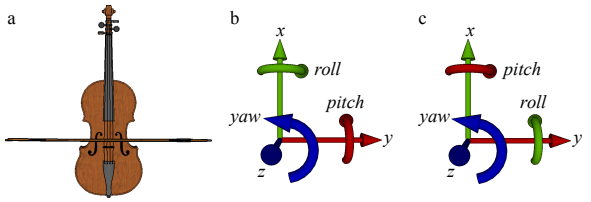

Figure 1: Violin with bow [1], violin and bow coordinate systems

State is defined in terms of three components in three-dimensional space that together compose a spatial model with six degrees of freedom (6DOF). The first is *position* as (x,y,z) relative to the initial position (0,0,0) from when the measurements started. It is not necessary to know the absolute start position in the real world (e.g., two meters from the wall, one from the floor), and in fact, these values have no specific units of distance. The second is *attitude* as *yaw*, *pitch*, and *roll* in degrees relative to the initial values of zero. The third is relative *time*, which contributes to computing the speed (change in state) and acceleration (change in speed).

The state of the violin in the coordinate system in Figure 1b serves as the reference against which to measure the state of the bow. The bow uses the system in Figure 1c because it is more intuitive to swap the roll and pitch axes to account for the perpendicular interaction. In other words, pitch for the bow in the right hand should be at a right angle to the pitch of the violin in the left hand. The desired bow yaw should be 90 degrees counterclockwise from the violin yaw about the z axis at the origin, which is where the bow and strings intersect. Bow pitch is the arcing movement as the bow passes over the strings. It is the angle of the bow relative to the angle of the violin on the plane formed by the x and y axes, ranging over roughly ±20 degrees.

Yaw deviation of the bow is what violinists at all levels strive to minimize. Pitch deviation is not an error because pitch must vary in order to interact with different strings. Roll deviation could be a consideration, but compared to yaw, it is minor. Determining the state on all axes, however, is necessary to solve the state of the complete system. The details of the math, physics, and engineering involved in moving a bow are out of scope, but their relationship to the larger project is worth mentioning for context. *Kinematics* is the study of geometry in motion without regard to the underlying mechanism (the *kinetics*), such as force, mass, and gravity. For example, pushing the base of the bow forward in line with it moves the other end a corresponding distance in the same direction. This event translates an action (the cause) into a reaction (the effect). In other words, the violinist must do the former so that the latter happens. *Inverse kinematics* in this context is the study of how to achieve this result given the many options. For example, the upper arm, elbow, forearm, and wrist can combine in many ways to produce the same action, and other actions can also lead to the same reaction. It is the teacher's job to make sure that the appropriate actions occur in the right way for the right reasons. This system considers only the kinematics of producing the result. In the bigger picture, the correct result is necessary to play a violin well, but it alone is not sufficient because technique also matters.

3 Architecture

The architecture consists of two complementary parts for reliably executing repeatable, controlled experiments: the physical model operates directly on hardware to simulate the role of the bow, and the virtual model is its software counterpart. Section 5 discusses how the two work together.

3.1 Physical model

The hardware is a computer-controlled electromechanical device consisting of a movable turret with a movable bow holding an IMU, collectively called the test rig.

3.1.1 Turret

The turret, a stock RobotGeek Roboturret in Figure 2a, provides a flat, open platform typically intended to hold a camera [6]. A separate hobby servo motor (2b) drives the rotational movement of its two degrees of freedom, pan and tilt, which respectively correspond to yaw and pitch for the bow. The turret is a self-contained unit with its own power supply, an Arduino Duemilanove embedded controller (2c), and a thumb-sized joystick [7]. It is strong and fast enough to simulate the angular movement of the bow, but it does have notable limitations, covered in Section 3.2.

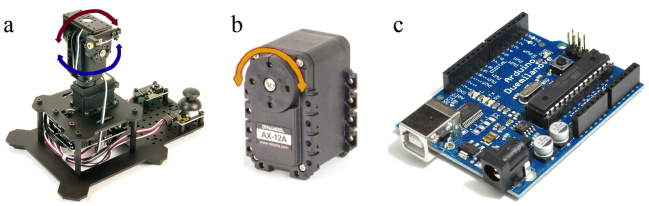

Figure 2: Turret, servo, and controller [6,6,7]

3.1.2 Bow

The drive mechanism in Figure 3a for the simulated bow occupies the platform of the turret. This component required design and fabrication from scratch. It consists of a geared motor with a pinion gear that meshes against the rack on a lightweight, extruded square aluminum shaft 50cm long. The motor driver is a JRK 12v12 (3b), which accepts a wide range of convenient parameters, such as acceleration and deceleration profiles [8].

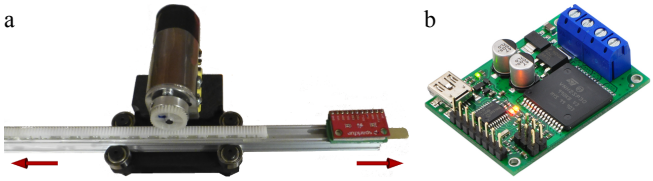

Figure 3: Bow drive mechanism and motor controller [8]

Rotating the motor and thus the pinion gear causes the shaft to move linearly through a low-resistance channel guided by bearings. This action must be precise because the shaft has to move to a specified position at a specified rate. The hardware has two complementary positioning strategies. The first is an open-loop control system that counts how many rotations of the motor occur in 1/48th increments. The motor has a built-in quadrature encoder for this purpose, which interfaces directly with the motor controller. Each partial rotation, or step, translates into a corresponding linear movement of theoretically around 0.06mm. However, the messy operating environment causes minor counting errors that compound over time. For example, moving the bow back and forth the same number of steps many times does not return to exactly the original position. This error is minor, but in combination with many other errors inherent throughout this work, this solution alone is not acceptable.

To mitigate this limitation, a simple closed-loop control system uses an optical sensor to determine when a black dot at the back center of the shaft passes the drive mechanism, which means that the bow is back in its home position. This signal resets the step count to negate any counting errors. In fact, it is so effective that the front of the shaft includes similar dots at 5mm increments as reference points for the image processing discussed in Section 5.2. The back dot also serves to initialize the bow to the same starting position for each test. The Arduino can also initialize it to other positions in code, or the user can manipulate the joystick.

3.1.3 Sensors

Reliably knowing the position and attitude of the bow and violin (real or simulated) is essential. Three types of sensors determine these values. In general terms, an *accelerometer* measures change in position from the previous measurement; a *gyroscope*, change in yaw, pitch, and roll; and a *magnetometer*, absolute yaw (i.e., compass heading). They collectively form an inertial measurement unit and typically reside on a single chip, here an MPU-9150 in Figure 4a [9].

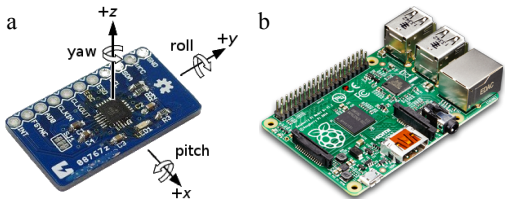

Figure 4: Inertial measurement unit and Raspberry Pi [9,10]

In reality, computing these values reliably is far more complex. Despite the apparent convenience of this IMU, it is in practice quite an unreliable device with messy output and many inherent errors. Higher-quality devices were available at acceptably higher cost, but they were larger and heavier, which was prohibitive for use on the end of a long bow in motion. (The supplier also billed this product as "the world's first" 9DOF device with complex onboard digital motion processing, which sounded highly promising [9].) Mitigating these errors in the larger project involved complex signal processing, primarily Kalman filtering, which is out of scope here [2]. Nevertheless, even with such processing, the results were hardly ideal, which is the subject of Section 6.

The two IMUs simply acquired state data. They did not process or store anything. For this part, they communicated with a Raspberry Pi, a small, inexpensive, yet powerful single-board computer in Figure 4b [10]. The programming language for processing the data was Python. While not the fastest for number crunching, it was adequate for the requirements. Moreover, it includes convenient libraries for communicating with the test rig over the I^2C and USB buses and GPIO (general-purpose input/output) pins.

3.1.4 Architectural overview

Figure 5 shows the architectural overview of the main components of the system and their communication. A laptop serves as the base station to provide the user with a convenient interface into the other components, which do not have a keyboard or display of their own. It also collects the raw data during tests.

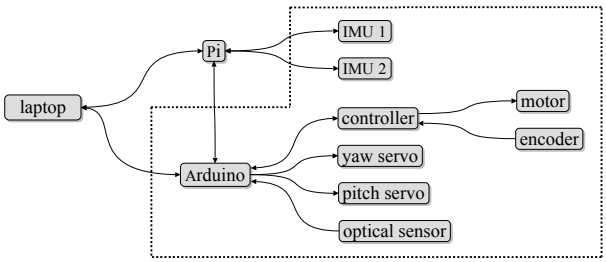

Figure 5: Architectural overview

Figure 6 shows the test rig, which contains the components within the dashed box above. The IMU is on the right end of the bow.

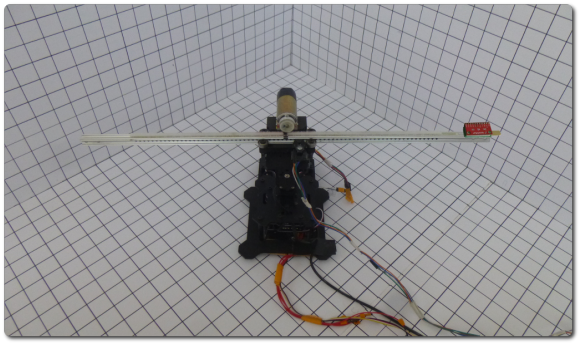

Figure 6: Test rig in calibration chamber

3.2 Virtual model

The physical model is a surrogate for the real-world bow, which does not reliably lend itself to controlled experiments. The virtual model in turn is a surrogate for the physical model, which exhibits a large number of issues that undermine its use as the only means of testing the IMU on the bow. Section 5 covers in detail how the three variants function together better in a modeling-and-simulation environment than any one could alone.

The virtual model is a simplified computational engineering representation of the turret and bow, as well as of the physical mechanisms that connect and move them. The turret has inputs for yaw and pitch, and the bow for speed and distance of motor rotation. The virtual model in this form is perfect, which would be the ideal goal for the physical model, too. However, the latter exhibits a wide range of errors and limitations owing to its real-world nature and the quality of the components and construction. In order to use the virtual model in place of the physical, it must model these errors to some configurable degree. Trial-and-error experimentation through simulation played this role.

All substantive errors are related to interconnections and movement. For instance, the yaw and pitch axes of the turret pivot through their respective servo motors. These low-cost hobby units have a noticeable amount of rotational backlash, or slop, in their ability to hold a specified angle: it can vary by plus or minus several degrees, depending on how much force is applied. The bow suffers from a similar problem in multiple respects because the motor has its own internal gears with backlash, and the pinion gear does not mesh perfectly with the rack. As a result, the bow can vary in position along its length by plus or minus a millimeter or more. The bow also has sideways slop because the bearings that hold it in position have some freedom. Reducing the tolerance improves this performance, but it introduces another problem because the friction becomes much higher, and the bow motor does not behave as uniformly. Everything in engineering design is a compromise. The goal here was not to produce the best solution for a specific set of test cases, but rather a generalized proof of concept that fleshed out areas to investigate further.

While the contribution of any of these errors alone is relatively small, they amplify over the length of the bow. For example, one degree of error at its maximum extension (normally avoided) equates to roughly a centimeter of perpendicular distance error that the IMU at the end sees. Even worse, some errors compound. For example, the yaw axis of the turret holds the servo for the pitch axis, so pitch measurements suffer from both errors.

A completely realistic virtual model would be overwhelmingly complex and difficult to define, test, and evaluate. Therefore, this abstraction and simplification ignores contributions from vibration and resonance (compounded interacting vibrations), as well as balance and stress-related factors. For example, the bow at its maximum extension exerts far more bending moment (twisting) at the bearings than it does in balance at its minimum extension. Similarly, electromagnetic effects on the magnetometers from the motors were problematic, but effectively impossible to model.

4 Visualization

A laptop logs the data from the controllers and sensors in quantitative form in terms of the expected and actual states (i.e., position and attitude), as well as time. The nature of this work lends itself to interpreting and evaluating the results in visual form. The visualization stage thus provides a variety of perspectives that help determine performance qualitatively, which tends to be more intuitive.

4.1 Two-dimensional static visualization

Static visualization involves displaying the results after a test is complete. Although it should be possible to present them dynamically in real time with additional software, the amount of data is large, and the changes are generally too quick and subtle for a person to follow in detail anyway. Instead, the output exports natively to Excel, as in Figure 7, where any manner of post-analysis is then possible.

time	px1	py1	pz1	ay1	ap1	ar1	px2	py2	pz2	ay2	ap2	ar2	epx	epy	epz	eay	eap	ear
0.00	0.073	0.089	0.159	1.563	0.014	0.115	0.659	0.217	-0.051	0.000	-0.177	0.126	1.981	0.018	0.248	-0.508	0.066	-0.037
0.10	0.010	-0.056	-0.045	-1.855	0.495	0.354	-0.122	-0.116	0.018	-0.085	0.067	-0.015	0.038	-0.034	0.349	0.251	-0.174	-0.036
0.20	0.064	0.096	0.106	-1.116	-0.064	-0.322	0.080	-0.031	0.020	-0.052	0.113	0.185	0.124	0.245	-0.406	-0.525	0.295	0.036
0.30	0.002	0.162	0.176	1.985	0.892	3.854	0.346	0.058	-0.043	-0.066	0.154	0.087	-0.748	0.908	1.022	0.049	0.278	0.000
0.40	0.081	0.118	0.247	1.716	-0.040	2.173	-0.561	0.066	-0.009	0.010	0.152	0.126	1.541	-0.059	-0.521	-0.341	0.050	-0.026
0.50	0.118	0.259	0.213	0.690	-0.275	-0.544	0.681	-0.036	0.006	-0.044	0.031	0.329	1.817	-0.227	0.151	-0.104	0.038	0.044
0.60	0.033	0.058	0.236	-1.293	-0.701	1.242	0.514	0.212	0.050	0.095	0.122	0.427	0.472	0.378	1.257	0.508	0.270	0.076
0.70	0.103	0.061	0.407	0.657	0.088	-0.936	-0.199	0.209	0.050	0.168	0.126	0.349	-1.532	-0.229	2.319	0.001	0.124	0.066
0.80	0.029	0.015	0.528	0.348	-0.778	0.154	0.370	0.089	-0.040	-0.010	0.120	0.188	-1.595	-0.009	1.531	0.272	0.337	0.160
0.90	0.090	0.038	0.388	1.593	-0.328	1.447	0.374	0.051	0.173	0.146	0.110	0.186	1.730	0.662	1.104	0.372	0.060	0.071
1.00	0.017	0.141	0.554	-0.247	0.260	0.972	0.974	0.141	0.155	0.097	0.101	0.529	1.804	0.394	1.151	-0.080	-0.230	0.166

Figure 7: Excel numerical data

Ordinary two-dimensional graphs in countless configurations and combinations can provide a wealth of insight into the results. Figure 8 shows a notional example from a one-second test on attitude (at 10ms intervals). The

paired lines show how the expected (smooth) and actual (jagged) results varied over time.

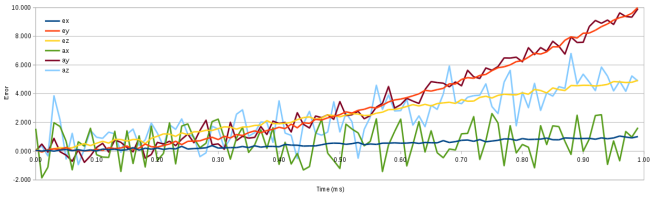

Figure 8: Excel graphical data

4.2 Three-dimensional dynamic visualization

Dynamic visualization involves displaying the results in real time as a test executes or afterwards statically in replay mode. The Java OpenGL 3D viewer in Figure 9 shows the mechanical configuration at any point in time from any perspective [4]. Its code is highly configurable and extensible to any special-purpose analysis. It can also render a variety of metadata to depict information that is not possible in the Excel visualization.

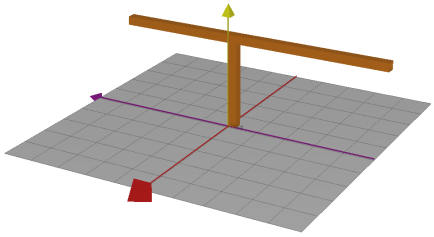

Figure 9: 3D viewer

5 Experimental framework

The overarching goal of this work was to use both physical and virtual modeling and simulation as a better integrated test environment for the IMUs than the real violin and bow could provide. In particular, it was essential to be able to run a wide range of tests for good breadth of coverage, as well as more instances of each for depth in statistical processing and analysis. The advantages of the physical and virtual models over the real bow in this respect are substantial. However, the value of their collective results depends on how well all three representations agree. There is no simple way to perform these comparisons because each representation has its own idiosyncrasies. For example, the real bow held by the violinist is, of course, the most accurate at being itself, but it is also the least reliable in accommodating a specific test. It is also completely useless for collecting multiple samples from the same test because the violinist cannot exactly repeat the same actions. In other words, the violinist would contribute the most errors and overshadow the errors in the IMU itself, which are the true interest. The physical model improves on these limitations, but it introduces inconsistencies that are not present in the real bow. Finally, the virtual model attempts to be a surrogate for both to tease out the performance characteristics of the entire system piece by piece; i.e., which components contribute which errors and by how much. Configuring the models to reflect reality involved substantial trial and error.

The approach was to compare the performance of the bow IMU on the same tests across various combinations of the physical and virtual models. All tests used the same methodology of mapping inputs to outputs and measuring error as expected versus actual output. Specifically, in the perfect world, the same input (cause) would map to the same output (effect) every time. In practice, however, a multitude of factors introduced many errors that could not be isolated directly. The following combinations were the first attempt at building an objective comparative framework, which is still a work in progress.

5.1 IMU against physical model (C–A)

The first type of experimental comparison took the obvious route and aimed to determine how well the data from the IMU on the bow in Figure 3a corresponded to where the test rig believed it actually was. For example, in the perfect case, the bow would go to the expected state of position (*x*,*y*,*z*) and attitude (*yaw*,*pitch*), and the IMU would indeed report exactly this actual state; i.e., no error. In reality, however, there are three types of states in play: (A) where the rig believes it put the IMU, (B) where the IMU actually is, and (C) where the IMU believes it is. Therefore, there are errors likely in the comparisons between states A and B, B and C, and A and C. The difference between A (input) and C (output) is supposed to reflect the IMU error, but in fact, it really includes errors from all three. Some of these errors are additive (i.e., two wrongs make a bigger wrong), and some are subtractive (i.e., two wrongs make a smaller wrong or accidentally even a right).

This system is heavily underdetermined, and no amount of comparison can completely overcome having more unknowns than knowns, as well as a lack of confidence in the truth of the knowns. Establishing (as much as possible) which states contribute which errors and how they combine is the purpose of the next four similar (and admittedly confusing) types of experimental comparisons. To reiterate, this C–A test above intended to measure the IMU performance by comparing state C and A, but it actually does C and B because A and B are not the same due to errors in the rig. (Comparisons are reflexive, so A against B is the same as B against A.)

5.2 Physical model against ground truth (A–B)

The second type of experimental comparison aimed to determine the performance of the physical model against the best representation of reality — the ground truth. In other words, this comparison was of state A against B to discover errors in the rig. These tests used the calibration chamber in

Figure 6 in combination with three digital cameras to provide front, side, and top perspectives. The grid on the background permitted accurate visual measurement of the actual state of the IMU by hand. An LED on the test rig indicated when the test started and ended so the three streams could be synchronized. The tests were the same as earlier: command the bow to an expected state, measure its actual state, and report the error.

While this approach produced the best results on the true performance of the test rig, it was totally impractical for real tests. The image processing was a very tedious manual effort of isolating the IMU against the background grid in all three perspectives, translating the coordinates, then calculating the corresponding position and attitude. Furthermore, only the start and end states were available this way, limiting this approach to static tests only.

5.3 IMU against ground truth (C–B)

The third approach compared the IMU (C) against the ground truth (B). These tests used the same conditions as those in Section 5.2 and actually occurred at the same time. For the same reasons, they were utterly impractical for real-time tests, but they did provide more insight into the nature of the errors throughout the system.

5.4 Virtual model against ground truth (D–B)

The fourth approach compared the virtual model (D) against the ground truth (B). This process involved grounding the virtual model to match the physical model, including its errors. Each error source in Section 3.2 is actually a range from minimum to maximum with a probability distribution (typical uniform or Gaussian). Endless trial and error resulted in values that produced the same general behavior as the physical model on the limited number of data points captured.

5.5 Virtual model against physical model (D–A)

The fifth and final approach compared the virtual model (D) against the physical model (A). The values painstakingly processed in Section 5.2 served as the training data, where the actual results could be tweaked until they reasonably matched the expected results. The correspondence was generally good because it is not a fair measure of performance to know both the questions and the answers in advance. The true measure is how well the virtual model performs on tests that it has not yet seen, which Section 6 addresses.

5.6 Recap

The previous subsections capture five of the six possible comparisons in Table 1. *PM* and *VM* stand for physical and virtual model, respectively. The qualifier *believed* refers to where the device reports itself to be, whereas *actual* is where it truly is. (*PM actual* corresponds to the omitted *IMU actual* because they are connected at the same location and would have the same values.) Comparison C–D is not an option at this point because it would require modeling the IMU and its own errors, which is far outside the scope of this work.

Table 1: Summary of comparisons

Section	Types		Description
5.2	A	B	PM believed vs. PM actual
5.1	A	C	PM believed vs. IMU believed
5.5	A	D	PM believed vs. VM actual
5.3	B	C	PM actual vs. IMU believed
5.4	B	D	PM actual vs. VM actual
	C	D	IMU believed vs. VM actual

6 Results and discussion

There was nothing elegant about the tests: they were pure brute force. This approach was actually convenient, however, because it mitigated the curse of dimensionality, which rapidly expands the test space into an intractable number of combinations as the number and range of test parameters increase [5]. Looping over the combinations in the virtual model was by intent very fast. The physical model, on the other hand, was relatively slow (and self-destructive over time as the rig wore out), but it was still immeasurably more effective than a violinist attempting such tests repeatedly.

The first category of tests involved static snapshots of the final state of the IMU after all kinematic actions had completed. Each test of the physical model involved averaging 10 independent runs, and on the virtual model, 100. (Strictly speaking, the number of runs should be the same, but the physical model would not have survived a higher value, and the probability-based virtual model would have suffered from a lower one.) Each run started from the same initial conditions. Angle parameters increased by 10 degrees per test. Bow extension increased by 10 centimeters, as measured from the pinion gear to the IMU.

Two subcategories considered parameters independently and in combination. The independent tests were:

1. yaw 0º and extension 10cm, pitch −30º to +30º; 7 tests
2. pitch 0º and extension 10cm, yaw −45º to +45º; 10 tests
3. yaw and pitch 0º, extension 10cm to 40cm; 4 tests

The combinational tests were:

4. yaw 0º, pitch −30º to +30º, extension 10cm to 40cm; 28 tests
5. yaw −45º to +45º, pitch 0º, extension 10cm to 40cm; 40 tests
6. yaw −45º to +45º, pitch −30º to +30º, extension 10cm to 40cm; 280 tests

As this paper is about using modeling and simulation in support of other work, this discussion primarily addresses the methodology, not the actual results per se. Czoski [3] covers the IMU performance in great detail. Space limitations also prevent further analysis. Tests 1–6 were static tests because they reported the final state only. The independent variants (1–3) were acceptable, whereas the combinational ones (4–6) were very inconsistent because of amplified errors. The second set of tests, 7–12, were respectively 1–6 again, but with intermediate states sampled at 10ms intervals. Figure 8 is a small representative example (based on Test 8) that shows how the expected versus actual states translated to error measurements. This graph considers only accuracy (i.e., how closely they agree); precision (how repeatable they are) and variance (spread, in terms of standard deviation) are also useful, along with many other statistical measures.

Unfortunately, while the virtual model corresponds reasonably well to the physical model in the static and dynamic independent tests and static combinational tests, it does not come close to reflecting the chaotic operating characteristics of the dynamic combinational tests. These tests are unfortunately the most representative of a violin and bow in actual use. It is questionable whether improving the virtual model would even be worthwhile because the physical model is so problematic.

7 Future work

For more meaningful and useful results, a better test rig is undeniably necessary. As a proof of concept, this one served its purpose, but it introduced far too many problems that unnecessarily complicated all aspects of this work. Likewise, better IMUs are needed. It is also likely that a second one on the other end of the bow might help correlate the raw data to mitigate some of the errors. Similarly, a more practical approach to establishing the ground truth via automated image processing could improve some of the convoluted inferences on A through D. Finally, full motion capture on a violinist could provide even more potential for appropriate grounding.

8 Conclusion

The goal of doing engineering on the cheap with commercial off-the-shelf components was reasonable, but no amount of basic modeling and simulation appeared to be on a promising track to compensate adequately for the many combinations of inherent errors throughout the system. Isolating a single type of error was indeed achievable, but in this messy, highly underdetermined environment, the final results collectively were unfit for actual use. Nevertheless, as a proof of concept, this work overall demonstrated that an integrated framework of modeling, simulation, visualization, and analysis successfully supports repeatable, controlled experiments in the otherwise intractable realm of real-time data collection and processing for complex violin movement. The widespread use of IMUs in countless other applications could benefit from this approach, as well.

9 References

[1] Adapted from Google Sketchup Warehouse, 3dwarehouse.sketchup.com, last accessed Mar. 17, 2016.

[2] Caron, F., E. Duflos, D. Pomorski, and P. Vanheeghe. *GPS/IMU data fusion using multisensor Kalman filtering: introduction of contextual aspects*. Information Fusion, vol. 7, no. 2, pp. 221–230, June 2006.

[3] Czoski, J. *A Violin Practice Tool Using 9-Axis Sensor Fusion*. Masters thesis, Eastern Washington University, 2015.

[4] Tappan, D. *A Pedagogy-Oriented Modeling and Simulation Environment for AI Scenarios*. WorldComp International Conference on Artificial Intelligence, Las Vegas, NV, July 13–16, 2009.

[5] Thomopoulos, N. *Essentials of Monte Carlo Simulation: Statistical Methods for Building Simulation Models*. Springer: New York, 2012.

[6] www.trossenrobotics.com, last accessed Mar. 20, 2016.

[7] www.arduino.cc, last accessed Mar. 20, 2016.

[8] www.pololu.com, last accessed Mar. 20, 2016.

[9] www.sparkfun.com, last accessed Mar. 20, 2016.

[10] www.raspberrypi.org, last accessed Mar. 20, 2016.