Estimation of physical properties of an object through interaction

Nienke Driessen[1]

Supervisor(s): R. Venkatesha Prasad[1]**, Kees Kroep**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Nienke Driessen
Final project course: CSE3000 Research Project
Thesis committee: Dr. RangaRao Venkatesha Prasad, Kees Kroep, Dr. M. Weinmann

Abstract

Creating a local model of a remote environment is a way to reduce latency in tactile internet. This local model contains properties that are estimated by a non-intrusive estimation method. To prevent the model from deviating increasingly from reality, the estimates should be updated once an interaction begins. This research paper investigates how the physical properties of a remote object can be estimated. This is done by exerting forces and observing the resulting motion. The physical properties that are updated include the mass and center of mass. These values are calculated to converge toward the actual value. The mass can be estimated by observing the linear motion, for which the game engine Unity is used. For the center of mass estimation, it is advised to take a different approach, since the values received by Unity were not enough to base an estimate on. This paper also investigates the influence of update frequency on estimation accuracy and the inaccuracies that the Unity physics engine presents.

1 Introduction

This research will pertain to the area of Tactile Internet (TI). This field aims to enable interaction with humans and cyber-physical systems at a distance as if they were nearby. TI is useful in situations where human presence is hazardous or costly [8], for example, remote reparations or remote operations in healthcare. Since these remote operations are steered by a local controller, maintaining a perception of the force exerted by the operating side is crucial. Since the interactions are performed at a significant distance, the information sent between both parties always comes with latency. For precise interactions, the feedback must have little to no delay. This is because delay can result in the robot performing actions that result in unintended mishaps.

A solution for this latency problem is the use of a local model. This is rendered on the local ”control” side, estimating the remote environment where the actual operation takes place [4]. This is called model-mediated teleoperation (MMT). Figure 1 shows a schematic view of the system interaction between the local and the operating side. A human operator makes use of a local physics simulation to dictate or demonstrate actions. Then the remote device will imitate, or perform the intended actions. The physics simulation is initially based on a non-intrusive parameter estimation method. The remote environment is estimated based on properties detected by a combination of sensors.

However good the initial guess may be, it is never completely accurate. This results in the model deviating increasingly from reality, resulting in an unstable system. That is why this specific research is about the adjusting section. The task is updating the estimated physical properties based on interactions.

The research question that will be answered in this paper is the following:

How can the physical properties of an object be estimated through interaction?

The physical properties that will be estimated in this report are the object’s mass and the Center of Mass (CoM). Therefore this research question can be divided into two sub-questions.

- *How can the mass of an object be estimated through interaction?*
- *How can the CoM be estimated through interaction?*

To limit the scope of this bachelor thesis, both the local and remote parts of the MMT will be a virtual setup. This is called a digital twin methodology, which is an approach that involves creating a virtual representation or model of a physical object, system, or process [1].

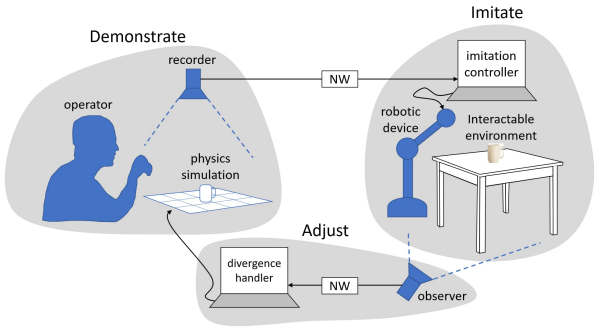

Figure 1: Schematic explanation of big picture system (Figure created by Kees Kroep).

The following contributions have been shown in this paper.

- Mass estimation method,
- An analysis of the influence of different start estimates on mass estimation,
- An analysis of the influence of the update frequency on mass estimation,
- Observations on the center of mass estimation.
- Observations of limitations of the Unity physics engine.

The organization of the paper is as follows: Chapter 2 outlines the work that has already been done in this field. Then a detailed explanation of the development process is described in Chapter 3. The experimental setup and results are discussed in Chapter 4. A reflection and discussion is held in Chapter 5. Chapter 6 discusses ethical concerns regarding this research. Future work recommendations will be discussed in Chapter 7. Finally, Chapter 8 covers the conclusions drawn from this research.

2 Related Works

A fair amount of research has been done on MMT systems. Tsumaki et al. distinguish between geometric modeling errors (errors in the position) and dynamic modeling errors (force and motion errors) [11]. Property estimation of a spring system is discussed by A. Shahdi and S. Sirouspour

[7]. Since a spring system is used, fewer degrees of freedom are possible than in the system explored in this paper.

The first work that considers a physics engine as a model for MMT is by Xu et al. [15]. It is stated by Li et al. [3] that Unity3D, an open-source visualization engine, has found extensive application in numerous digital twin-based systems [12]. This shows that unity is a useful tool for this field. However, only a limited number of studies within this domain have focused on the semantic-level modeling of digital twins [12].

Several studies have been done regarding estimating mass and CoM. A common CoM estimation method relies on the weighted segmentation method [5]. Here the CoM is calculated using the center of mass for each body segment and summing them to estimate the center of mass for the whole body. Since in this report, both objects are assumed to be equal in shape, and the initial CoM location estimate will probably be based on this tactic, this method can not be used in this study. No works have been found about estimating mass and center of mass solely by measuring the reaction on forces. Setterfield et al. proposed a method of detecting the CoM of a rotating object in free space by observing the trajectory [6]. In this paper, the object is not free in space.

A lot of latency compensation research has been done in the field of cloud-based games [9]. This is compensated by time warp to compensate for latency and reduce network bit rates. This is however not relevant for this research but can be useful in the future.

A teleoperation system that is robust to modeling errors is proposed by Tsumaki et al. [11]. This article discusses the model-jump effect. This occurs when a model needs to be updated during the interaction between the controller and the operator. The controller can for instance experience unexpected changes in predicted forces, which results in unexpected motion. This can result in an unwieldy interaction and trigger a jump in the remote movement. This can cause instability in the model and eventually result in dangerous behavior. [13] A simple way of dealing with the model-jump effect is presented by Xu et al. [15]. In this paper, the object geometry updates are said to only be applied when the operator is free in space to avoid the model-jump effect. When the difference between the local and real models is too significant, the controller is asked to command the operator back to free space and stop the exploration. Xu et al. also explain how they partition the differences in the two models in two fields, namely geometry and physical properties [16]. An update is necessary if at least one of the two differs above a certain threshold. Perception-based model updating is mentioned by Xu et al., where only values that can result in significant differences in perception will be sent back to the controller [16]. This way only relevant changes will be handled to reduce computational load.

3 Methodology

The task at hand is developing a system that can estimate the mass and CoM of the remote cube by converging the properties of the local cube to approximate the unknown properties of the remote cube. This will be done by progressively refining the local system until convergence. Estimates will be derived for both mass and the center of mass location by analyzing the resulting motion created by applied forces. Figure 2 shows the general concept of the system.

The initial values for the local mass and center of mass will in reality be determined by the non-intrusive mass and the mass center prediction. For this study, realistic values will be assumed.

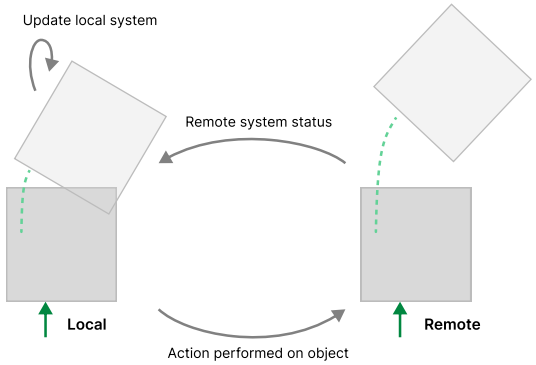

Figure 2: Update cycle of the system. The force to be exerted is sent to the remote object. At a specified rate the local side will receive information about the resulting remote state. Using this information, the local model will be updated.

3.1 The Unity setup

For the scope of this project, the actions will be limited to a 2 dimensional (2D) system, looking from a top view. This is to reduce the degrees of freedom and thus to reduce the complexity.

Even though Unity supports both 2D and 3D scenes, it was decided to use Unity3D, since realistic friction with a surface below is essential to a realistic interaction. In 2d, realism can be faked to some extent by adding linear and rotational resistance, but this would unnecessarily complicate the problem.

As shown in Figure 3, a scene containing two objects will be used. For this study the objects are cubes. One is representing the local object and one is the remote object. The two cubes do not interfere with each other, but for visualization purposes, they are placed at the same location. The task is to get the local cube to behave exactly like the remote cube.

Both cubes get equal forces acting upon them. The movement of the cubes will be constrained to 2D, limiting their linear movements solely within the x-z plane. The rotational motion will be limited to the y-axis.

The local system receives information on the state of the remote cube at a specified rate. This will consist of the following four items

- Position,
- Rotation,
- Linear velocity (v),
- Rotational velocity (ω).

These values were chosen because it is realistic that they could be measured in a physical setup. The update rate is

chosen to be 60 updates per second (60Hz) since it is generally considered a desirable update rate for smooth and fluid visual experiences. Even though this is not actual latency, the system does have to work with receiving information every x ms, and recalculating the states at each step, as it would in an actual situation with latency. The performance of the system using different update rates will be assessed in Chapter 4.

In each update instance, The location, rotation, v, and ω of the local model are all updated to match the remote model, to minimize the difference in movement and the number of unexpected changes. The new mass calculation is then solely based on the current mass, and the observation is not interfered with by the velocity resulting from the previous masses.

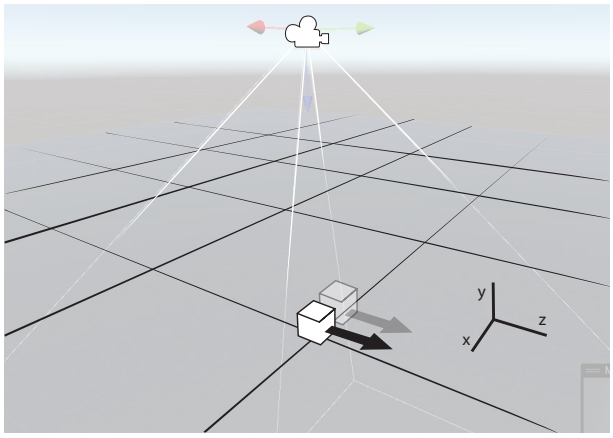

Figure 3: Explanation of unity scene setup.

Since the mass and CoM estimation algorithm can grow complex quickly, several assumptions have been made.

System assumptions

1. The mesh estimation method works perfectly. Therefore the shape of both the local and remote objects, the meshes, are equal.
2. The objects can only move in 2D space.
3. The objects exist of one single shape, a uniform mesh. No compound objects are used.
4. The force exerted is a point pressure.

The mass estimation algorithm went through a number of iterations as shown in Figure 4 in order to create a stable system with good-quality code on which tests can be performed.

3.2 Approach structure

First, a mass estimation method will be developed, with equal center of mass locations. Then, the center of mass estimation will be done with the identical mass. Therefore the task will be divided into the following sub-tasks to strategically solve the objective.

1. Mass estimation
   (a) Mass estimation based on linear forces
   (b) Mass estimation based on linear and rotational forces

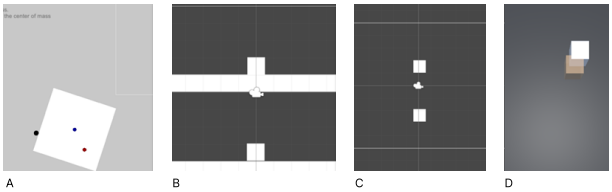

Figure 4: All iterations of the mass estimation setup. A) is the first iteration, where a side view was assumed. A point pressure could be dragged to tip over the cube. B) Shows the first occurrence of a local and a remote cube. Both were moved using impulses in stead of continuous forces. C) Is the first top-view version of the system, though only one update per force applied could be calculated. D) is the final 3 dimensional version where the mass is continuously updated, with forces able to act for a number of seconds.

2. CoM estimation
   (a) CoM estimation with difference only in one axis.

3.3 Mass estimation method based on linear movement

The first objective is a descending method for mass estimation with only linear forces on the center of mass. This means that no rotation is possible. Based on the received values of the remote behavior, The mass m of an object can be determined through only linear motion. By knowing the linear momentum p and the linear velocity v [2] the equation for linear momentum,

$$p = mv, \tag{1}$$

can be used. Eq. (1), the function for linear momentum p is used instead of the function for net force F because even though the force acting upon the object is known, the net force is not calculable. This is because the force resulting from friction is not calculable. Friction in Unity is not necessarily realistic.

> *"The friction model used by the Nvidia PhysX engine is tuned for performance and stability of simulation, and does not necessarily present a close approximation of real-world physics." [10]*

In particular, it is said that for situations where two contact surfaces are larger than a single point, friction is calculated as the objects having two contact points[10]. This is the case for this study.

Although the remote linear momentum can not exactly be known, it can be estimated. That is why a descent method is chosen in stead of directly calculating the mass solely based on physics equations. Since the mass and velocity of the local object are known, the local linear momentum can be calculated. Then, with the remote velocity and assuming an increasingly similar momentum, the remote mass can be approximated.

Table 1 shows the calculation that is initiated each update instance,

$$m_{\text{new local}} = \frac{m_{\text{local}} v_{\text{local}}}{v_{\text{remote}}}. \tag{2}$$

Table 1: Mass calculation and update sequence. The new mass is calculated based on received remote velocity.

Step	Formula	Explanation
1	$p_{\text{local}} = m_{\text{local}} v_{\text{local}}$	Calculate local linear momentum
2	$m_{\text{remote}} = \frac{p_{\text{local}}}{v_{\text{remote}}}$	Calculate remote mass with local p and remote v
3	$m_{\text{local}} \leftarrow m_{\text{remote}}$	Update local mass

Eq. (2) shows the resulting mass equation. This shows that if the local velocity is larger than the remote velocity, the mass will increase and vice versa. Since this way the local mass will converge to the remote mass, we can assume the momentum will become more similar as well. Therefore a descent method based on physics equations will be used as an approximation method.

3.4 Mass estimation method based on rotational movement

The rotational equivalent to linear momentum is angular momentum L. To determine the moment of inertia I of an object through only angular motion, we can use

$$L = I\omega, \tag{3}$$

where ω is the angular velocity. In a similar manner to how mass determines the force needed for a desired acceleration, rotational inertia represents the quantity that determines the torque necessary to achieve a desired angular acceleration around a rotational axis [2]. Generally, the moment of inertia is defined as

$$I = mr^2, \tag{4}$$

where m is the sum of the mass's products, and r is the distance from the rotation's axis. Since I also depends on the mass of the object, the same strategy can be used that has been used for the linear forces.

$$m_{\text{new}} = \frac{m_{\text{local}}\omega_{\text{local}}}{\omega_{\text{remote}}}. \tag{5}$$

3.5 The combination of linear and rotational estimation

To observe whether combining the two estimation techniques could bring an improvement, the performance of linear and rotational mass estimation was compared. This was done by applying the same force sequence and noting the relative error of each mass estimation for each update.

Figure 5 and 6 show that the mass estimation based on observing linear movements shows more accuracy and more stability. It needs to be noted, however, that this is the result of a single session. Although over several sessions, the trend

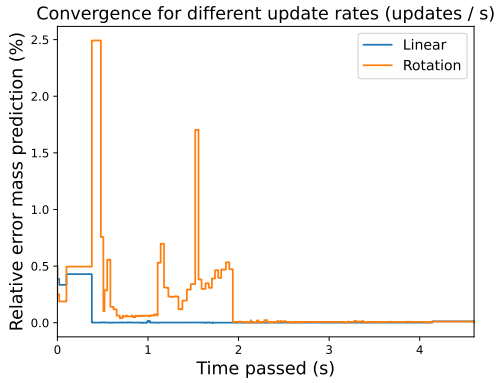

Figure 5: Performance linear vs rotational mass estimation.

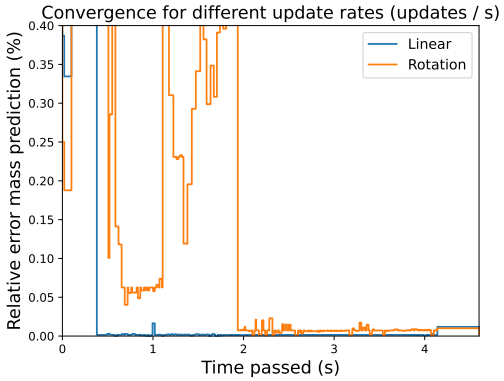

Figure 6: Performance linear vs rotational mass estimation, y-axis limited to 0.4.

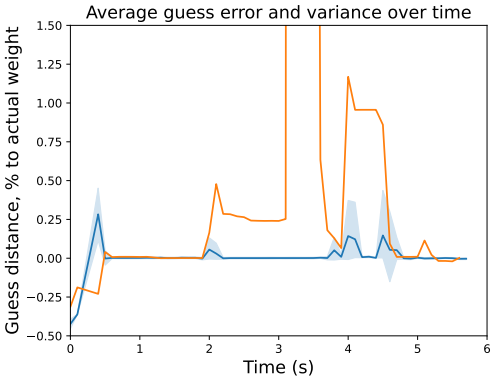

Figure 7: Performance linear vs rotational mass estimation. The distance to the actual weight, in percentage plotted against the time in seconds. The Translucent area shows the distance to the furthest guess at that timestamp.

consistently is that linear is more stable and precise than rotation. This can be seen in Figure 7, where over 5 sessions an average line is drawn, including the variance. It was deliberately chosen not to take the absolute error, but the relative error. This was to see whether the guesses are more likely to

overestimate or underestimate. Figure 7 shows that the large peaks are mostly over-estimations.

The probable cause for this is the fact that in linear movement you can directly calculate the mass, whereas in the rotational movement, we can calculate the inertia tensor. This relates to the mass but also to other factors, such as the force and shape of the object.

Another factor is that friction in unity does not accurately represent how a cube would behave in the real world. Linear movement could be less affected by this difference.

In summary, the remote mass can be estimated separately through both linear and rotational observation. Even though the original idea was to combine the two for stability, this will not be done. Linear momentum on itself is already accurate enough to estimate the remote mass. The rotational movement can be used to observe the inertia and estimate the center of mass location.

3.6 Center of mass estimation

In Unity, the mass of an object is uniformly distributed across the volume of the mesh. The way to make objects with a difference in mass distribution can only be obtained by creating compound objects. Since Assumption 1 in section 3.1 states that the local object is the same as the mesh of the remote object, and Assumption 3 states that no compound meshes are used, the only changeable 'factor' that can explain a difference in the moment of inertia is the center of mass. Therefore a descent method can be created that descends the center of mass to the remote value using the difference in the moment of inertia.

To simplify the problem at first, only one estimate update will be performed. Only forces in the x or z directions will be used. The updated estimate should be closer to the actual CoM.

The hypothesis is that based on the reaction of a cube, *amount of difference* tells us the factor of how far away the CoM point in the plane is orthogonal to the force ($f_{\text{difference}}$).

Then the *difference of the rotation* tells which side the estimate should move towards (f_{dir}). Figure 8 shows a schematic view of the hypothesis.

The only difference that can confidently be calculated of the CoM is in the orthogonal plane. Therefore only forces in the x and z directions should be enough the have the cube converge to the correct value. The equation to calculate the new CoM is

$$\text{CoM}_{\text{new}} = \text{CoM}_{\text{previous}} + \frac{f_{\text{dir}} f_{\text{difference}}}{r_{\text{cube}}}, \tag{6}$$

where r_{cube} indicates half the size of the cube. $f_{\text{difference}}$ is a value between 0 and 1. Where 0 means no difference at all, and 1 means the most difference, while the CoM is still inside the object. The first task is to get the center of mass to move in the correct direction based on f_{dir} and the correct amount based on $f_{\text{difference}}$, starting from the middle of the cube. Once this works, the equation can be changed to handle multiple updates. Figure 9 shows the setup for the CoM estimation.

Regrettably, no underlying logical pattern could be discerned in the values received from the Unity system. Various factors were analyzed, including rotational velocity, rotation, torque, acceleration, and inertia tensor. Eq. (3) was used to analyze the movement. Yet no discernible coherence was found. The observed values displayed a wide range, without

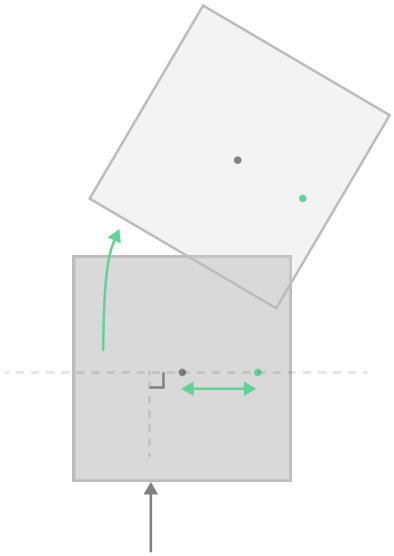

Figure 8: Schematic view of CoM estimation hypothesis.

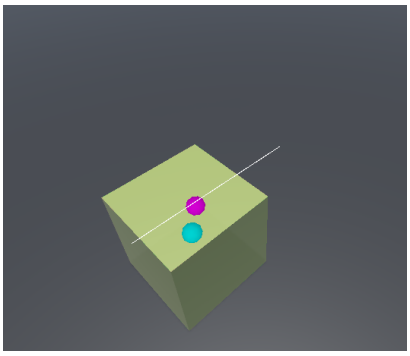

Figure 9: Test setup center of mass estimation. The white line indicates the location and direction of the force. The pink sphere shows the location of the CoM of the remote object, while the blue sphere indicates the location of the CoM of the local object. The force will rotate to remain in the same direction relative to the cube's rotation.

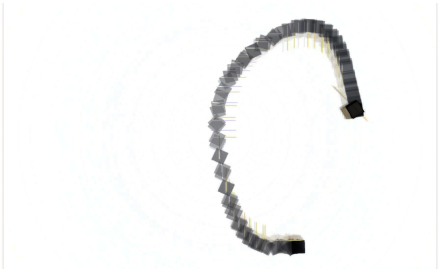

Figure 10: Long exposure photo of the force movement.

any direct correlation with the applied force or difference in movement. Furthermore, no apparent relation was observed between these values and the distance between the centers of mass along the orthogonal line of force.

4 Results

Several factors need to be tested to get a realistic view of the estimation method's performance and accuracy. First, the mass estimation will be evaluated. Second, CoM estimation will be tested.

4.1 The experimental setup

It is important that all test runs, where factors are compared, are based on the same forces and movements. Therefore an order of forces has been defined on which all tests are based. This section will explain the test sequence.

The sequence contains different forces that result in linear and rotational movements over time. Figure 11 shows the order and the type of forces that are exerted on the cube. The force magnitude is constant throughout all forces. Using Unity's coroutines, tasks are spread across several frames.

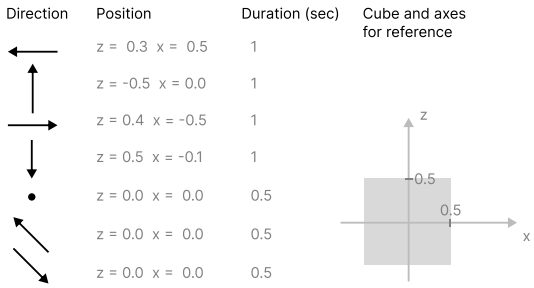

Figure 11: Explanation of sequence of forces performed during test sequence.

Figure 12: Long exposure photo of the test force movement. The cube starts at the bottom right and moves rotating along the path as a result of the forces applied.

Figure 12 shows a long exposure photo of the movement of the cube throughout the test sequence, to get a better insight into the movement. The force sequence is explained by Figure 11. The video of the test run is available on YouTube via *this link*.

4.2 Performance of the mass estimation method

Mass estimation performance with different initial estimate error

First, the mass estimation performance based on linear movement will be evaluated. Test runs were performed starting at different start values, where a mass of value 1 in Unity is assumed to be 1 kilogram (kg). The start values ranged from 0.5 kg to 6 kg. Figure 14 shows that the poorest performing values are the values with a much lower initial estimate, compared to the actual mass. These values both converge at a slightly higher error rate and have more noise than the estimates starting at higher values. It is also noteworthy that lower values have high peaks in mass estimation error at the beginning, which occur less in the higher values. Overall, if the initial estimates are kept above 50% of the actual mass, the error will remain below 0.02% of the mass of 3 kg.

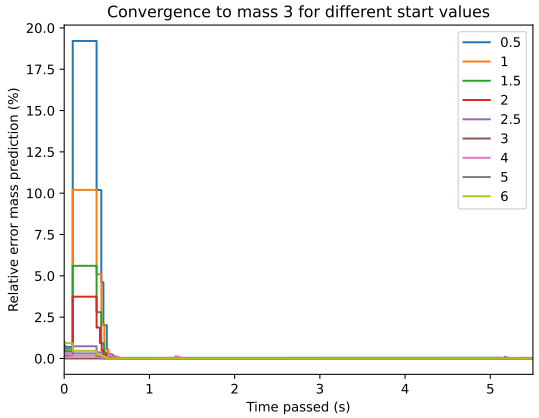

Figure 13: Convergence of different start values. The mass start values are kg.

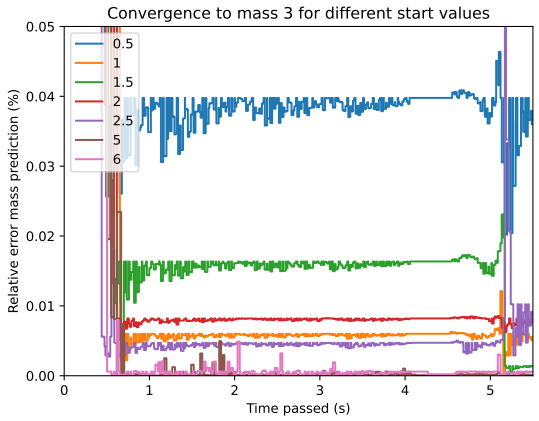

Figure 14: Convergence of different start values, y limit at 0.05. The mass start values are in kg.

Mass estimation performance sensitivity to update frequency

It is useful to know what the effect of update frequency is on the performance of the mass estimation to know at what latency this method still works. A number of different latency's have been tested, where a mass initial estimate of 1kg converges to the actual mass which is 2kg. Figure 16 shows that only once the update rate

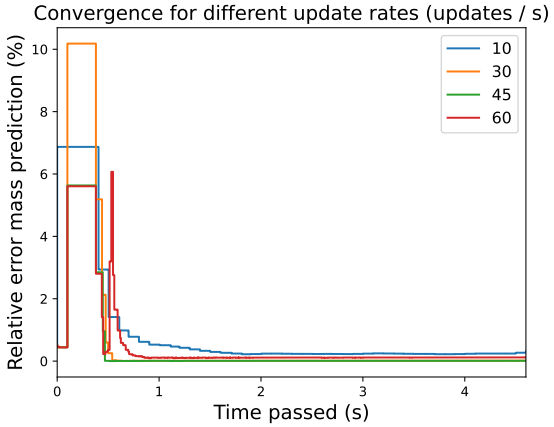

Figure 15: Convergence of different update rates.

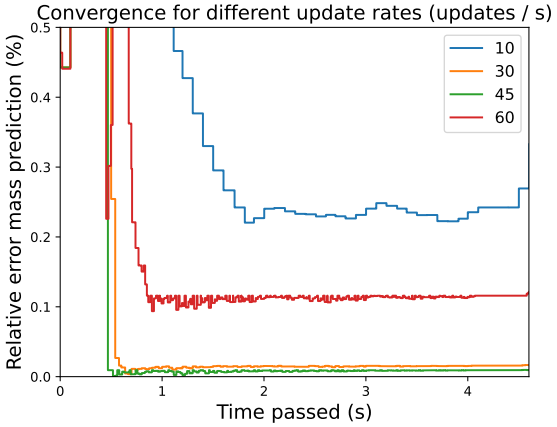

Figure 16: Convergence of different update rates, y limit at 0.5.

An interesting note is that while the "10 Hz" rate performs slightly less, 30 and 45 have a higher performance. A theory as to why this occurs relates to the fact that Unity also by default runs 60 FPS. In Unity, the FPS is determined by the performance of your game and the capabilities of the hardware running it. Unity has two update functions that perform at different rates.

1. **Update**: This is called every frame, typically 60 times per second. This is primarily used for regular game updates and input handling.
2. **FixedUpdate**: This is also called every frame but is synchronized with the physics engine's time step. By default, this is at a fixed rate of 50 FPS. This is primarily used for handling physics-related calculations.

Since using 60Hz to update the mass update system as well, it can happen that this interferes with both Update and FixedUpdate. It is noticeable that the 30 and 45 update rate both have less noise. It should be noted that the artificial delay that we used the update rate for, is not actually a substitute for latency, since we receive values of the current state of the remote system, instead of the state of a certain delay previously. There the so-called model-jump effect could have a greater impact at lower frequencies.

5 Discussion

As there has been no existing methodology for estimating mass in this manner, there is currently no basis for comparison regarding concrete results.

Since the system underwent a lot of versions, it was easy to iterate on the design. That way the impractical decisions could be undone or improved. The unity training course that I took before this project helped me to write better-quality code. This ensured that I did not make unnecessary mistakes and gave me insight into how Unity deals with certain functions.

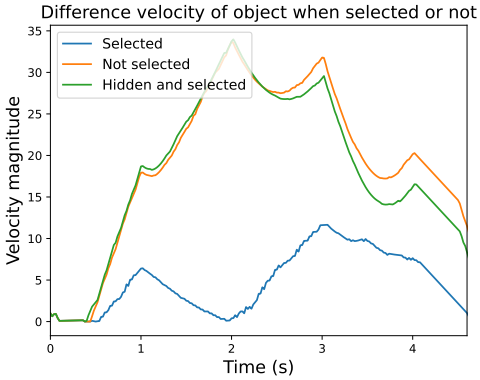

Figure 17: Significant difference in velocity based on whether the object was selected in the editor at the time of the run.

5.1 Unity physics

One improvement I will take into account next time is doing more research on how Unity works with physics. There are a number of factors that influence how the object reacts to forces. This created some unforeseen difficulties. One example is the fact that when the object was selected in the object inspector during the run of a test, the object reacted differently when the object was not selected. The velocity of the object became significantly smaller in the selected run. This can be seen in Figure 17. This strange behavior ended up being the result of the inspector being present. In Unity, the values inside the inspector can be changed during the run of the system. If the inspector is open, the system checks whether the values are changed in each frame. This might influence the behavior of the object.

5.2 Reproducibility

Even though Unity's physics engine is deterministic, each run results in a slightly different outcome. This could be because of the different frame rates. Just like physical experiments, multiple runs are necessary to collect reliable data.

6 Responsible Research

One of the primary ethical concerns in model-mediated teleoperation is system stability. A stable system is crucial for maintaining a safe and reliable environment for both the human operator and the remote object. An unstable system may lead to unexpected movements or even failures, which can endanger the operator and disrupt the task at hand. Since the current mass estimation can have sudden disruptions in predicted mass, these should be diminished. This can be done by adding a certainty value or making sure that the system is calibrated beforehand and no changes in mass can be done during the operation. Furthermore, since no humans were involved in this study, there is little ethical risk related to humans.

7 Future work

Certainty factor

The first addition that would have a great impact on the stability is a certainty factor. The system will keep track of the certainty of a certain mass and CoM. Once this certainty level is high enough, the mass will not be updated anymore. Then the mass could also be the median of all estimates. And thus the risk of disrupting a good system because of measurement errors will be minimized. However, even though it is very robust to measurement noise, it can lose flexibility. For example, if the friction gets a bit less or more during an operation, if you don't keep updating the system can get unrealistic.

Calibration

The second suggestion is to have a "calibration mode" that will occur before the operation. Then the certainty factor can be used to determine which forces need to be exerted to efficiently eliminate any uncertainty. Then the system will not or only slightly be updated during the operation.

Also for the CoM determination, if a calibration mode would be used, the CoM could be determined very efficiently. Then other, more efficient methods can be used such as the plumb line method [14]. By lifting the object twice, or three times in 3D, at a different anchor point and comparing the plumb line intersections, the CoM can be determined much more efficiently. This is shown in Figure 18.

Compound objects

Since changing the center of An interesting idea is to investigate the creation of compound objects to estimate an object's mass distribution.

The model jump effect

As discussed in Section 2 about related works, the model-jump effect is an interesting problem that accompanies MMT and tactile internet. It would be interesting to look into the perceived effect of the updates, and how updating the values more gradually can improve interaction but make the system less stable.

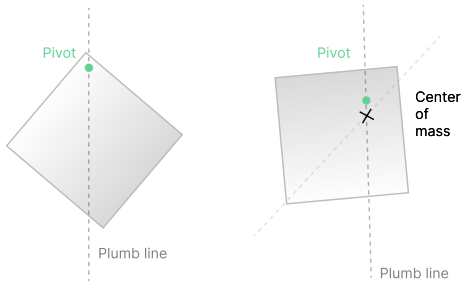

Figure 18: Plumb line method for estimating CoM.

3 Dimensions

Since in 2 dimensions, there is significantly less complexity in estimating properties. The mass estimation will likely not increase in complexity, but especially the CoM estimation will present more problems. Other factors like tilting and tipping over will come into play. Then most of all will it be useful to have a calibration mode.

8 Conclusion

In conclusion, this research has successfully developed a method for estimating the physical properties of a remote object through interaction. In answer of sub-question "*How can the mass of an object be estimated through interaction?*" The established technique achieves stable mass estimation with an error rate of less than 0.01%. The mass estimation system also shows solid performance at lower update frequencies, although it is important to note that the introduction of a delay can lead to an increase in the model jump effect at lower frequencies.

Furthermore, the accuracy of the initial guess has been found to have minimal impact on the convergence time and eventual accuracy of the estimation. However, lower estimates may result in early peaks during the estimation process. The type of forces exerted on the system has a more significant influence on the convergence time.

To ensure stability and reliability in mass estimation, it is advisable to initiate slight movements of the objects before engaging in any high-risk operations. The introduction of a certainty value proves beneficial in achieving convergence towards a stable and dependable mass estimate.

The Center of Mass prediction method has proven to be more complex than anticipated. While a descent method can theoretically be developed, Unity is optimized for performance and not necessarily for realism in physics aspects, thus relying solely on differences in rotational velocity values and physics-based calculations is challenging. As a conclusion for sub-question "*How can the CoM be estimated through interaction?*", further research is recommended in this direction to enhance our understanding and address this limitation.

References

[1] P. Aivaliotis, K. Georgoulias, Z. Arkouli, and S. Makris. Methodology for enabling digital twin using advanced

physics-based modelling in predictive maintenance. *Procedia CIRP*, 81:417–422, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[2] Hyperphysics. Rotational-linear parallels, moment of inertia. http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html. Accessed Jun. 16, 2023.

[3] Xin Li, Bin He, Zhipeng Wang, Yanmin Zhou, Gang Li, and Rong Jiang. Semantic-enhanced digital twin system for robot–environment interaction monitoring. *IEEE Transactions on Instrumentation and Measurement*, 70:1–13, 2021.

[4] Carolina Passenberg, Angelika Peer, and Martin Buss. Model-mediated teleoperation for multi-operator multi-robot systems. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4263–4268, 2010.

[5] Gabriel Ploof, Bassam Alqahtani, Farwan Alghamdi, Garret Flynn, and Cai Xia Yang. Center of mass estimation using motion capture system. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 287–292. IEEE, 2017.

[6] Timothy P Setterfield, David W Miller, John J Leonard, and Alvar Saenz-Otero. Mapping and determining the center of mass of a rotating object using a moving observer. *The International Journal of Robotics Research*, 37(1):83–103, 2018.

[7] Ali Shahdi and Shahin Sirouspour. Model-based decentralized control of time-delay teleoperation systems. *The International Journal of Robotics Research*, 28(3):376–394, 2009.

[8] Jingzhou Song, Yukun Ding, Zhihao Shang, and Ji Liang. Model-mediated teleoperation with improved stability. *International Journal of Advanced Robotic Systems*, 15:172988141876113, 03 2018.

[9] Jiawei Sun and Mark Claypool. Evaluating streaming and latency compensation in a cloud-based game. In *Proceedings of the 15th IARIA Advanced International Conference on Telecommunications (AICT)*, 2019.

[10] Unity Technologies. Physic material. https://docs.unity3d.com/520/Documentation/Manual/class-PhysicMaterial.html. Accessed Jun. 14, 2023.

[11] Yuichi Tsumaki and Masaru Uchiyama. A model-based space teleoperation system with robustness against modeling errors. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pages 1594–1599. IEEE, 1997.

[12] Kung-Jeng Wang, Ying Hao Lee, and Septianda Angelica. Digital twin design for real-time monitoring – a case study of die cutting machine. *International Journal of Production Research*, 59:1–15, 09 2020.

[13] Bert Willaert, Hendrik Van Brussel, and Günter Niemeyer. Stability of model-mediated teleoperation: Discussion and experiments. In *Haptics: Perception, Devices, Mobility, and Communication: International Conference, EuroHaptics 2012, Tampere, Finland, June 13-15, 2012. Proceedings, Part I*, pages 625–636. Springer, 2012.

[14] Science World. Finding the centre of gravity. https://www.scienceworld.ca/resource/finding-centre-gravity/. Accessed Jun. 15, 2023.

[15] Xiao Xu, Burak Cizmeci, Anas Al-Nuaimi, and Eckehard Steinbach. Point cloud-based model-mediated teleoperation with dynamic and perception-based model updating. *IEEE Transactions on Instrumentation and Measurement*, 63(11):2558–2569, 2014.

[16] Xiao Xu, Burak Cizmeci, Clemens Schuwerk, and Eckehard Steinbach. Model-mediated teleoperation: Toward stable and transparent teleoperation systems. *IEEE Access*, 4:425–449, 2016.