Under the Robotic Knife: A Verifiable Controller for use of Multiple Robotic Arms in Surgery

Zachary Barnes
Department of Electrical and Computer Engineering
University of Pittsburgh
zach.a.barnes@gmail.com

***Abstract*— Surgical robotics are extremely complex and potentially life-saving pieces of technology. At the moment, they remain only in partial use and are often excceedingly expensive and unreliable in operation. Verification is proposed as a method of developing safer and more effective platforms. A robotic arm was analyzed and a controller was developed for using such an arm in a multi-arm robotic surgury platform. A system was designed and prooved to safely avoid static obstacles. A system was designed and verified to allow for safe operation of robotic arms using controller placed boundaries. A dynamic boundary system was designed and verified for to create autonmous collision avoidance. Lastly, A parabolic approximation method was explored, and efficiency and complexity trade-offs are discussed in depth.**

I. Introduction

Imagine a hospital of the near future. Every room, every piece of equipment, and, most importantly, every patient is continually monitored. Decisions about patient care are made in split seconds. Drugs are administered automatically. In the operating room, sophisticated robotic platforms perform complex procedures once considered dangerous and invasive. In this place, technology is relied upon to help people, to save lives. Cyber-physical systems (CPS), computing systems that interact in the physical world, are a crucial piece of enabling technology in turning this vision into reality.

The da Vinci surgical robot is such an example of a current cyber-physical system. Arguably, the da Vinci is one of the most complex robots on earth. It allows physicians beyond human control of surgeries [1]. Commercially, the da Vinci costs around 1.5 million dollars. Even at that astounding cost, the devices safety has been called into question, as there have been documented cases of malfunction and failures, many with injuries to a patient [2]. With such an example, it is easy to see why similar devices are not ubiquitous in the medical field. Thus the question looms, for the betterment of healthcare and the wellbeing of patients, how can the development safe medical devices be improved?

One proposed potential solution is a greater reliance on the use of formal verification in the design and analysis of such complex cyber-physical systems. Verification, in this sense, constitutes developing a system model and using proof rules and logic to prove that the controller provides safe operation. Here, verification will be used to develop and prove safety for a controller of a multi-limbed robotic system similar to the da Vinci robot.

Fig. 1. The da Vinci robot surgery platform.

II. Background

A. Differential Dynamic Logic (dL)

Using the language of hybrid programs in the environment of differential dynamic logic, allows for the description of a complex system in a simple manner. These hybrid programs are constructed out of formulas, logical statements that resolve either true or false, and programs, which provide functionality. The characteristic forms of programs and formulas are as follows:

$$Formulas ::= \not\phi | \phi \wedge \psi | \phi \vee \psi | \phi \rightarrow \psi | \forall x.\phi | \exists x.\phi | [\alpha]\phi$$

$$Programs ::= x := \theta | x := * | x' = \theta | ?H | \alpha; \beta | \alpha \cup \beta | \alpha^*$$

Equation 1 is an example of the typical format of a formula created to prove safety of a system [3].

$$pre \rightarrow [(ctrl; ODE)^*](safety) \tag{1}$$

This formula has the meaning that, if shown valid, all executions of the program inside of the brackets ([ ]), will satisfy the safety property [3]. The program utilizes a control decision (ctrl) that will discretely alter the execution of the program based on the current state of the system. Next, the system continuously evolves according to the set of differential equations (ODE) that govern the operation of the system. Lastly, according to the *, the program will repeat itself a nondeterministic number of times [3]. What is unique about hybrid programs is that it allows for the modeling of both discrete and continuous events, a crucial principle in CPS, in a sound framework [3]. This allows for the utilization of proof rules to transform a dL formula, like the statement in equation 1, to a set of decidable formulas in first order logic [3]. This will be done utilizing the Hybrid Systems theorem-prover KeYmaera [4].

B. Relatable Verification Studies

Formal verification has been studied in the context of surgical platforms before. One system that was analyzed was that of a robotic controller to assist in complex skull surgeries designed at Johns Hopkins University (JHU) [5]. The system used fixed boundaries predefined by the surgeon, which could not be crossed with a hand-held scalpel device [5].

From this study, it was shown that verification could reveal subtle bugs in controller software that are often overlooked in design. This proposed research furthers the results of the work at JHU by examining the particular dynamics and interactions of multiple robotic arms rather than a single hand-held scalpel. In this study, the results are more general, in a sense, as they will apply to any device that uses a robotic arm, even outside the field of surgery.

III. System Specifics

The robotic arms in use commercially are often exceedingly complex, and vary significantly in application. Thus, in order to present a general and feasible solution, an anthropomorphized robotic arm is considered. This arm, as its name suggests, is similar to the human arm in its joints. It has a shoulder, elbow, and wrist joint. Since the wrist joint is used for small minor hand motions, it will be considered as the end of the arm for the scope of this project.

This arm is used in the design of a verifiable controller for the multi-arm situations that are inherent in the da Vinci robot. In a three dimensional field, the shoulder allows for rotation about the y plane and its rotation effects both the wrist and the elbow joints. The elbow joint rotates about the shoulder and in a plane that corresponds to the current rotation of the shoulder. Its rotations in that plane affect the wrist, which is also allowed to rotate in the same plane and about the elbow joint and its rotations affect no other joints. A model of such a robot is shown below in Fig. 2:

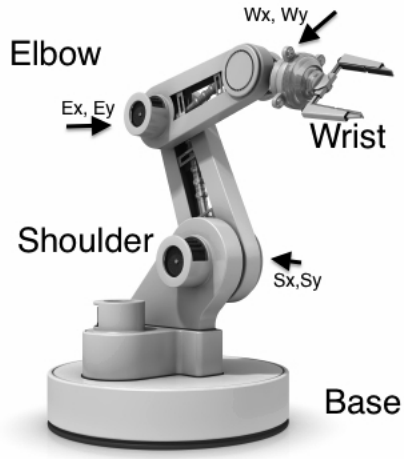

Fig. 2. Anthropomorphized Robotic Arm

The point S (Sx, Sy, Sz) will be used to represent the center for shoulder rotations. The point E (Ex, Ey, Ez) represents the center of rotation for the elbow joint, which the wrist rotates around. Lastly, the point W (Wx, Wy, Wz) represents the wrist joint, which is the end of the robot arm. The surgeon will move these arms by specifying distinct velocities for each arm. For the most part, smart controllers will be used to limit the operations to remain safe.

A. Essential Properties

One of the key properties for a surgical robot is safety. Without safe operation, the potential boon of such robotics will be rejected. Having safety is then especially important for complex surgeries that may require robotic assistance, and without safe robotic controls, cannot be performed. Here, certainty is crucial.

The specific safety properties for each model as expressed in dL as in Eq. 1 will vary throughout each model. However, in general the two concerns that will be addressed are obstacle avoidance and arm intersection.

Obstacle avoidance can be thought of as the need for the surgeon to specify a particular object, namely a crucial artery, organ, or other important biological feature, that must remain untouched throughout the operation. Being able to enforce this is critical in performing surgeries near vital biological elements and to help guard against dangerous robotic operation. This feature also exists as a check on the surgeon to safeguard against any possible reason for misuse of the equipment, or emergency that may arise with the operator.

Arm intersection is the event in which two robotic arms collide with each other. As in the case of the da Vinci robot, many robotic systems require the use of multiple robotic arms. However, in a complex operation, the intersection of two arms could prove disastrous. The result of such an event could even be life-threatening for the patient. Even more, these events are less obvious than obstacle collision, and being able to perform operations while consciously avoiding every possible arm intersection is not feasible. Thus, this feature seeks to provide a way to dynamically prevent the collision of two arms during an operation to enhance the safety of the procedure, and ease the job of the surgeon who can focus purely on the surgical aspects.

Another important property is that of liveness, or relative functionality of the system. It is important that these controllers are not overly restrictive and thus do not allow the surgeon to perform the tasks needed. It is also important that arms do not fall into a position in which there are no possible safe movements, and thus become stuck. This could also be a disastrous outcome.

This project will be formally concerned with the obstacle avoidance and arm intersection, and keep the liveness of the system in consideration without exploring it in verification explicitly.

IV. Static Obstacle Avoidance

A. System

With the overwhelming complexity that is inherent in multi-armed robotic arm scenarios, reducing the problem to something manageable is key. The explicit nature of the motion in each direction is an example of this complexity. Ex(t) and Ey(t) will be both be unwieldy sinusoidals, about which reasoning directly is very difficult. Thus, roundabout arguements focusing on verifiable approximations are used.

First, I examine the robotic arm in a two dimensional space. Here, I assume that the shoulder joint remains stationary. Additionally, the obstacle to be avoided will also be a stationary point O (obx, oby) that is located anywhere on the two dimensional plane. This means that the obstacle, and the joints W, S, and E will all lie on this two dimensional plane.

Here, the system is designed to avoid collision with the static obstacle. This representation of this model can be seen in Figure 2 below:

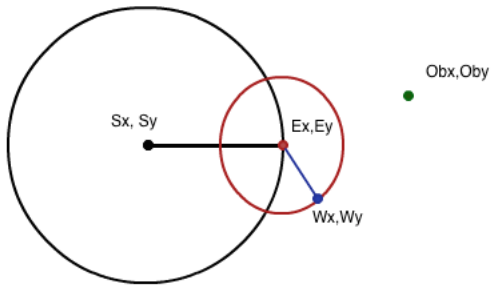

Fig. 3. 2D representation of the robotic arm with a static obstacle O.

B. Safety

In this representation, safety is the ability for the arm to avoid collision with the obstacle. That is, no point along the arm can collide with the obstacle. This means that both the line segment from the shoulder to the elbow joint, SE, and the line segment from the elbow to the wrist, EW, should not collide with the obstacle. This is done to ensure for the surgeon that the entire arm, not just the tip as the wrist, will not collide with

the obstacle. This property is expressed in dL in the following equation:

$$\forall c.(c >= 0 \land c <= 1 \rightarrow$$
$$(c * ex \neq obx \lor c * ey \neq oby)) \land$$
$$\forall c.(c >= 0 \land c <= 1 \rightarrow$$
$$(c * rwx + ex \neq obx \lor c * rwy + ey \neq oby))$$

This formula uses quantifiers over a constant C to specify every possible value along the vector from S to E and from E to S. These values are indeed every point along the line segments SE and EW. Then, safety specifies that for every one of these points, either the x-value of that point does not equal obx or the y-value at that point does not equal oby. Together, this conclusion shows that no point along those line segments is ever in the same position as the obstacle. That is, this safety property shows that the arm does not collide with the obstacle at any point. Note that any obstacle that is greater than the sum of the radii of the arms is trivially safe, and thus these cases are not focused on.

C. Dynamics

This arm is controlled by a velocity that is set non-deterministically in the beginning of the controller. This models the behavior of a user specifying how they wish the arm to rotate, such as a surgeon performing an operation. In this case, only the elbow is allowed to rotate and the wrist is held constant. The governing dynamics of this system are determined by its differential equations as seen in the equation below:

$$ex' = -ve * ey/re,$$
$$ey' = ve * ex/re,$$
$$t' = 1 \& t <= T$$

This dynamics specify circular motion about the point S, which for simplicity, is assumed to be the origin. Here, the model denotes a circular trajectory about S with constant linear velocity Ve at a radius Re about the origin. Here Ve controls the rate at which the arms can rotate, and is allowed to be positive or negative, and can only be a magnitude of Vm, the operating velocity of the system. This is a simplifying assumption made about the dynamics of the robotic system. In reality, the arm would have to accelerate with some value before arriving at the operating velocity of Vm, but this model assumes Vm to be small, and the acceleration to be large, and thus the time it takes for the arm to accelerate will be minimal. This assumption then allows for the discrete change of the variable for velocity from Vm to 0 to Vm, without having to worry about relative acceleration and deceleration. This greatly reduces the complexity in the controller stage.

In addition, W is not allowed to rotate and is held stationary; its final value determined as a sum of W=E+RW. This models a situation in which the W is not effected by the rotation of E other than its final position. That is, W retains its relative position on its circular path throughout the evolution of E. By the evolution domain, t is held to be less that T, and thus the system will be allowed to evolve at most T seconds.

D. Controller

In this initial examination of the obstacle problem, the controller seeks to restrict the range of motion of the robot to ensure that the arm does not collide with the obstacle. In order to do so, the arm is restricted to operation only in the region where it remains to one side of the obstacle. For simplicity in this analysis, only one position, that of $obx > 0$ was examined. The controller than keeps the arm (point E and W) on the left side of the obstacle. The decision for the controller can be seen in the following equation:

```
if(
!((ve>=0 &
    obx-ex-ve*T>0|
    ve<0 & obx-ex+ve*T>0)&
  (ve>=0 &
    obx-wx-ve*T>0|
    ve<0 & obx-wx+ve*T>0)))
  then ve := 0
fi;
```

That is, the arm will have its velocity set to 0 if it crosses over on to the right of T seconds. This is an example of a time triggered controller, where the system will be tested for allowed operation every T seconds, a product of the limit on execution in the evolution domain that $t <\leq T$. Therefore, logically the controller checks whether the arm will remain safe after heading in T seconds in the direction of the obstacle at a velocity of T seconds. Thus, if it is still less after T seconds, the arm can be sure that it can allow one evolution before it must check again.

E. Proof Methods

Many steps were taken in order to prove safety for this relatively simple controller and system.The first was to generalize the problem by showing that $E < O$

and $W < O$ prove safety. Then the system needed to be shown to hold these properties. Next, I used a loop invariant to show that in each repetition of the program under the *, the key properties held true. This invariant included many constants and, importantly, the fact, which is the conclusion, that $E < O$ and $W < O$. This invariant was easy to show proved safety.

Next, the proof branches with the control decisions and the initial assignments of the value of Ve. Three important branches of the proof are created. The first is that $Ve = 0$, and this is either by assignment by the user, or by the failing of the controller. In either condition, safety is readily shown by proving that $E < O$ and $W < O$ is invariant along the evolution of the differential equation. Intuitively, this approach means if the arm is safely behind the obstacle, than if it doesnt move, it remains safe.

The next branch is if $Ve > 0$, and is reached by passing the controller. In order to prove this branch, a solution to the motion of E in undertaken by cutting into the differential equation that $Ex^2 + Ey^2 = Re^2$ which is easy to show through a differential invariant. With this, it is then easy to show through differential weakening that that Ey ¡= Re, a simple but important result. Next, the fact that $Ex_0 + Ve * t \geq Ex$ is cut in as a property. This can then be easily shown as true using a differential invariant using the fact that $Ey \leq Re$. This simply means that traveling at Ve in the x-direction will always go farther than by traveling around a circle of the same speed. With this, the control decision that it is safe after T, and the fact that $t \leq T$, it is relatively easy to show that this branch is safe.

The last branch proceeds identically to the one before it except that this time $Ve < 0$, and thus the cur needs to be that $Ex_0 - Ve * t \geq Ex$. Through differential weakening both prove.

Note that since $W = E + RW$ and since RW does not change, the cuts for E are the only significant part to proving that $W < O$.

F. Results

Although this problem did prove easily in terms of arguments it still took some force. The problem is that the arguments tend to repeat themselves with only very minute changes. Also note that the region of operation that is limited by this controller is quite large. That is, any and all regions greater than the obstacle are off limits. This is indeed very restrictive for a system that might need to have functionality near to an obstacle, such as spinal surgery.

One simple proposed improvement is to take into account the y direction as well. Then, the controller would have to keep the arm either to the left or below an obstacle in the positive quadrant. This would additionally branch the proof making it least twice as many arguments needed. An example of this controller can be seen below:

```
if(
  !((ve>=0 &
      (obx-ex-ve*T > 0 | oby-ey-ve*T >0)|
ve<0 &
      (obx-ex+ve*T > 0|oby-ey+ve*T >0))&
    (ve>=0 &
      (obx-wx-ve*T > 0| oby-wy-ve*T >0)|
    ve<0 &
      (obx-wx+ve*T > 0|oby-wy+ve*T >0))))
then ve := 0
fi;
```

Additionally, note that this problem was only in consideration of one of four regions for placement of an obstacle. That is, this argument for proof of the entire system would take at least four times as many cases into account, and would cause a great amount of branching. Lastly, allowing for W to rotate would also have to take as many cases into account as well.

Ultimately, while successful, this approach to avoidance would be very unwieldy and complicated based on the sheer number of cases that would need to be considered. In order to create a system that needs a function with potentially numerous and differently shaped obstacles, a more elegant solution is needed, hopefully a less restrictive one.

V. Barrier Avoidance

In expanding to tackle avoidance of arm intersection as well as potentially differently shaped obstacles, a new solution was needed. Here, the problem is an innate complexity that comes along with having multiple arms that rotate. Spelling out all possible potential situations that might lead to danger is unfeasible.

Therefore, this system instead utilizes barriers to separate the regions of operation between the robotic arms or the arm and any obstacle. The idea being that if a controller can set a boundary between the arm and potential danger, and then enforce that it does not cross the boundary, thus it will remain safe.

A. System

This system is very similar to the static obstacle system. That is, this model is also confined to two

dimensions, the x and y plane. This means that the joints W, S, and E will lie on this two-dimensional plane in addition to the controller generated boundary. This model can be seen in an overview in the following figure:

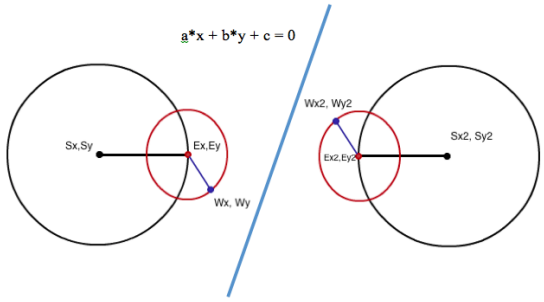

Fig. 4. Two Robotic arms with a barrier seperating their respecitve regions of operation. The barrier is of the form ax+by+c=0.

B. Safety

For this system, the concept of safety is slightly different than before. Here, it ensures that the robotic arm must fully stay on its side of the boundary that was created at the beginning of each iteration of the program. That is, the distance from the line to each joint (S, E, and W) must not change sign during the evolution. An example of unsafe behavior would be a arm that starts above the line and some part of the arm ends the evolution below the line. For this problem, the case where the arm starts above the line was used, but the argument is entirely reflexive for the case when the arm starts below the line. The safety property for this model can be seen in the equation below:

$$\begin{gathered}\forall x. \\ (\forall n.(n >= 0 \wedge n <= 1 \wedge x = sx + (n * (ex - sx)) \\ \rightarrow sy + n * (ey - sy) > (a * x + c)/-b)) \\ \wedge \forall x. \\ (\forall n.(n >= 0 \wedge n <= 1 \wedge x = ex + (n * (wx - ex)) \\ \rightarrow ey + n * (wy - ey) > (a * x + c)/-b))\end{gathered}$$

This safety property uses once again quantifiers along the vectors from S to E and E to W to reason about every point along the line segments that are the arms. This property also uses the equation

$$D(x_0, y_0) = \frac{a * x_0 + b * y_0 + c}{(a^2 + b^2)^{(1/2)}}$$

to represent the shortest distance from the line to the point in consideration. A positive value for this distance means that the point is above the line and a negative value that the point is below the line. Here, it is denoted that the point is above the line if that for every value of x along the line segments, that the value of y along those line segments is greater than the value on the line (ie above the line).

Once again, the point of this safety property is to show how two arms (or an arm and an obstacle) will not collide by showing that their regions of operation are distinct. That is, by showing the controller can keep an arm on one side of the barrier, then the use of this controller with two arms utilizing the same boundary and keeping to opposite sides of that barrier can be used to prove that the arms will not collide, and thus remain in safe operation.

C. Dynamics

The dynamics of this model is very similar to that of the static obstacle. Like the previous problem, these dynamics allow for the non-deterministic assignment of velocities to the E joint and to the W joint. These are represented as Ve and Vw respectively and they are limited to operation at Vm, the positive operating velocity. Vm would also be an acceptable choice for velocity, but for simplicity, this problem only examines the use of a positive velocity because the arguments for the -Vm are identical. These dynamics can be seen in the figure below:

$$\begin{gathered}\{ex' = -ve * ey/re, \\ ey' = ve * ex/re, \\ rwx' = -vw * rwy/rw, \\ rwy' = vw * rwx/rw, \\ t' = 1 \\ \&t <= T\}\end{gathered}$$

While E operates as it has before, with circular motion about the point S. Now, to represent W as circular motion about the point E, a more roundabout approach is taken. This approach is to instead rotate a point R about the origin in the exact same path as W would, in order to represent W as components. That is, to represent W as the sum of E and R. As it will be seen in the controller, both E and W can move, but only one at time. This is an approach used that eases the strain on the controller and decreases the complexity of the problem without a reduction on functionality.

D. Controller

The object of this controller is also to restrict the motion of the arm to prevent intersection by setting a barrier do denote specific regions of operation. However, picking one barrier, and enforcing it throughout time, while safe, would drastically reduce the operation of the system. That is why this controller allows for the picking of a new barrier at each time step. This barrier can be any line of the form $ax + by + c = 0$, and must also be valid in separating the regions of operation of the arm. This control can be seen in the following program utilizing non-deterministic assignment.

```
a := *; b := *; c := *; ?(b>0);
k := *; ?(k = 1/(a^2+b^2)^(1/2));
?(
k*(a*sx+b*sy+c) > 0 &
k*(a*ex+b*ey+c) > 0 &
k*(a*wx+b*wy+c) > 0);
```

With this boundary in place, of which the arm is strictly above, the controller then must enforce that the arm will remain in this region for one time step of T. In order to do this, the controller then checks to see if the arm moved at a velocity (Ve, or Vw) towards the barrier for T, that the distance from each joint (S, E, and W) would remain positive (or that the arm would stay above the barrier). If this is not the case, then the velocity of the arm is set to 0. These decisions can be seen in the program below:

```
if(!(k*(a*ex+b*ey+c)-ve*T > 0 &
 k*(a*wx+b*wy+c)-ve*T > 0))
     then ve := 0
  fi;
  if(!(ve=0 & k*(a*wx+b*wy+c)-vw*T > 0))
    then vw := 0
  fi;
  t := 0;
```

Additionally, an illustration of the behavior of the controller and this reasoning can be seen in Fig. 5 and Fig. 6.

E. Proof Methods

In order to prove safety for this controller, it was crucial to isolate what is needed to assure that the arm will stay in its given region of operation.

The first step is in constructing a loop invariant that stays true throughout every evolution of the program and is also strong enough to imply the conclusion.

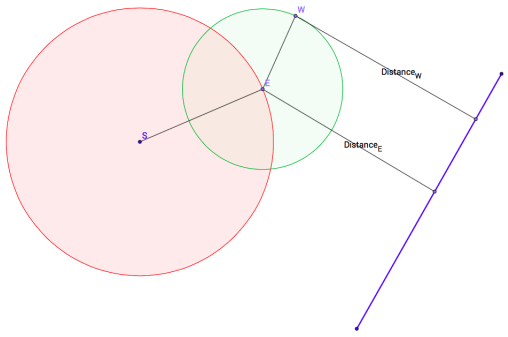

Fig. 5. Distances from the joints E and W initally

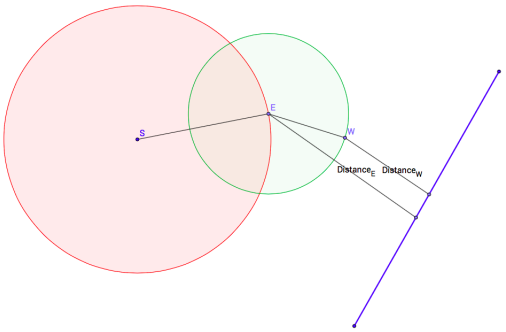

Fig. 6. Distances from the joints E and W at a later point in time

The invariant that was concluded can be seen in the following code:

```
re > 0 & ex^2 + ey^2 = re^2 &
rw > 0 & rwx^2 + rwy^2 = rw^2 &
b>0 & k = 1/(a^2+b^2)^(1/2) &
wx = ex + rwx & wy = ey + rwy &
k*(a*sx+b*sy+c) > 0 &
k*(a*ex+b*ey+c) > 0 &
k*(a*wx+b*wy+c) > 0
```

This invariant was initially valid, and also satisfied the use case for the important fact that the distance form each point to the line remained positive throughout execution. Now, getting this to close took some significant massaging of the variables, mostly in dropping out k which is a constant that stands for $1/(a^2 + b^2)$, that serves to significantly muddle the use of quantifier elimination. Now this can be done relatively easily by cutting in that the points are above the line, which can be proved knowing that the distance to the line is positive.

Now, the more difficult portion of the proof was to prove that this invariant is valid. In order to do

this, a series of compositions and assignments were completed that are a result of all of the assignments, nondeterministic and otherwise, that take place in the beginning of the controller on each step. Then, the important conclusions must be massaged out of the differential equations primarily using differential cuts, differential weakening, and differential invariant. Once again, this is because the solutions to these differential equations governing circular motion are not distinct and reasoning about them is extremely complex and difficult as they are of the form of sinusoids. Instead, this approach allows for determining results without stepping up the complexity.

Here, the important solution is that the distance from each point to the line is strictly positive. Through the assignments, the proof branches into four distinct elements. The first is the case when both Ve and Vw are 0. This is easy to prove since the differential equations do not alter the current states. The next case is when both Ve and Vw are positive. This case is also easy to prove in the fact that the controller does not allow this state to occur and thus the preconditions fail and the branch is shown to be trivially true.

Next, is the branch where Ve is positive, and Vw is 0. This is the case when only the elbow joint is rotating. The key argument here is that the rate at which E is moving is less than the assumption made in the controller. That is, evolving along the circular path is slower than moving at Ve toward the barrier. This is proven first by cutting in to the differential equation the fact that the arm follows circular motion at a radius of Re. With this solution it is must be shown that the following expression is true:

$$(aey + bex)^2 \leq (a^2 + b^2) * re^2$$

This formula was derived knowing the fact that at any point along the path of a circle, the sum of the x and y components are less than or equal to root 2 of the radius, that is $sin(t) + cos(t) <= 2^{(1/2)}$. It can be shown that the sum of sin and cos constructs another sinusoid that has amplitude of root2. Thus, the sum of two sinusoids with the amplitudes of a and b can be shown be $(a^2 + b^2)^1/2$. This result is squared to give the equation above.

This result is extremely important in proving that evolving along the circular path is slower than moving at Ve toward the barrier. To show this, the current distance from the line to from the point is recorded using an auxiliary variable D0, and then the fact that $D0 - Ve * t \leq D$ is cut into the differential equation. Using a differential invariant, this statement resolves to the fact that $1 \geq k/re * (aey + bex)$. Then using the result from equation X, the problem reduces to $1 \geq 1$ which is true. Since W is a sum of E and R, which in this case is constant, than this result was shown to apply to W as well. With these results, differential weakening is able to prove the final conclusion that $D > 0$ with the controller statement that $D0 - Ve * T > 0$. This was difficult for the arithmetic mainly because of sheer number of different variable at play. Thus, in order to close, Skolem abbreviation was used for many formulas to reduce the problem in complexity and allow it to reason about the logic of the statement without confusion about the variables.

The last case where Ve is 0 and Vw is positive proceeds in an almost identical fashion except reasoning about E is not necessary, as it does not move.

F. Results

The results from this line of inquiry into the use of dynamic barriers was positive. Here, barriers were successful as a means of ensuring safety for these arms. What is even more promising is the versatility they provide. By developing controllers that are able to effectively handle and place barriers to motion, all manner of different safe operations become available. Multiple barriers could be used to show that the arm will not collide with any number or shape of obstacles. The initial approach by using a point was limited in application, whereas the use of barriers can prevent accidental collision with lines, circles, squares, or any type of non-uniform shape. This is crucial in medical applications because avoiding organs, arteries, specific brain tissue, or even the spine is imperative and these objects are not reliably approximated with any uniform shape.

Additionally, it has been argued that these barriers can set the boundary between two arms, but with multiple barriers, the number of arms that can safely operate in a confined space can increase dramatically. This approach could very well be scaled up to include the four robotic arms that are utilized in the da Vinci system, or possibly even surpass this number. The tradeoff here is that adding more arms increasingly restricts the regions in which they can operate. These issues would have to be weighed on a case by case basis on what is the optimal number of arms for the situation.

It also important to note that in this approach there is a tradeoff as well in simplicity and efficiency in the

controller decisions. By using the approach detailed above, in which the controller does not allow motion if the point would cross the barrier after moving T seconds in that direction, the controller decreases its simplicity but it also increases how conservative it is in restricting motion. In the first iteration of this controller using barriers, reliance on simplicity is useful in proving the potential for this approach. Later sections will discuss an approach on how to improve on the efficiency.

Additionally, one must consider the relative realism that this model presents. While the controller is indeed relatively succinct and would not be too difficult to implement, having solid approximations about the locations of the joints in space might be difficult. Additionally, the assumption that the joint W is not dependent of the rotation of E is a simplification that is rarely true in the real world. A solution to this is proposed in the following advanced model.

G. Enhanced Barrier Avoidance

This model differs from the previous model in the fact that it is designed to take the previous joints rotation into account during evolution. That is, it means that when E rotates, the W joint must rotate with it as would happen in reality. The following figure illustrates this concept:

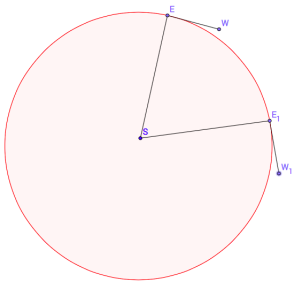

Fig. 7. Location of points E and W initally, and then again after rotation at E1 and W1. Demonstrates expected rotation of W dependant on E.

In order to model this behavior, one could use differential equations exclusively. However, this would require complex calculations to establish valid motion, and with these complex differential equations comes even more complex differential invariants. Thus, such an approach would greatly increase the complexity of the system. Instead, the approach taken resembles that of the previous model of separate evolution and a final summation to correct the model. An examination of the dynamics reveals that throughout an execution, the angle between E and W should remain constant.

In order to model this discretely, direction vectors for E and for W must be introduces into the dynamics. These changed dynamics can be seen below:

```
{ex' = ve*dex,
 ey' = ve*dey,
 rwx' = vw*dex,
 rwy' = vw*dwy,
 dex'= -ve*dey/re,
 dey'= ve*dex/re,
 dwx'= -vw*dwy/rw,
 dwy'= vw*dwx/rw,
 t' = 1
 & t <= T}
```

Here, the direction vector changes according to the speed and the radius, and then is used in the differential equation for the point to create how the point evolves based on these direction vectors. The following invariants can be easily proved for this new set of dynamics:

```
ex^2 + ey^2 = re^2 &
rwx^2 + rwy^2 = rw^2 &
ex=re*dey &
ey= -re*dex &
dey^2+dex^2=1 &
dwx^2+dwy^2=1 &
rwx= rw*dwy &
rwy= -rw*dwx
```

Examining these vectors in this model reveals that the angle between then also must be constant. Thus, in using the dot and cross product one can create a rotation matrix for the angle of rotation between the two direction vectors. Thus, after E evolves that the direction vectors change, the rotation matrix is used to generate the proper direction vectors for W. Then using the equations for invariants, I can piece together the location of the W as E plus R. Where R now is now the position based on previous rotation as calculated by the rotation matrix and added to the independent rotation that is undergone by the joint. These equations are summarized below.

```
cos := dwx*dex+dwy*dey;
sin := dwx*dey-dwy*dex;
```

which is calculated at the beginning of the time step. And at the end of the time step the rotal is calculated,

```
wx:=ex+rwx+rw*(dex*cos+dey*sin);
wy:=ey+rwy-rw*(-dex*sin+dey*cos)
```

Thus using this approach allows for a discrete representation for the joint W, which has a complex trajectory. The positive for this is that it models the actual behavior more closely and works to reduce the differential complexity. However, the downside to this approach is that the position for W is only calculated after the differential equation and thus limits the use of any evolution domain restriction on W.

Yet, even with the enhanced model, which more closely describes real world behavior, there is a need to add in more functionality into the system. This is explored as part of the dynamic barrier model.

VI. Dynamic Barrier

A. System

The previous model allowed for a barrier to be added at the beginning of each time step and then allowed motion so long as it did not cross the barrier. This does indeed allow for safe operation of the system but adds in no additional features or functionality. Take for example the surgeon moving arm1 towards an important element for surgery, but this is where the barrier lies so the arm stops. Then the surgeon must move arm2 away, reposition the barrier, and then begin the procedure again. Such annoyances are distracting and time draining for the surgeon. Thus, this model proposes a dynamically adjusting barrier, and autonomous control of the arm to keep the arm away from the barrier but also clear room to allow the other arm to move. An illustration of this concept can be seen in the figure below:

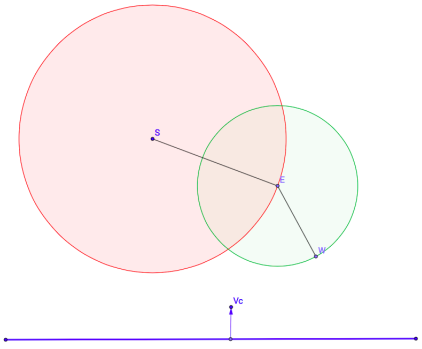

Fig. 8. Robotic arm model with a moving barrier (with velocity Vc) below the arm.

This model specifically handles the case of a moving towards an arm directly above it, and sets the motion according for it to remain safe.

B. Safety

As in the previous models, safety here is keeping the arm above the barrier. That is, every point along the line segment SE and EW must at all times remain above the barrier. This is formalized in dL as seen below:

$$\forall n.(n >= 0 \land n <= 1 \rightarrow$$
$$a * (sx + n * (ex - sx)) +$$
$$b * (sy + n * (ey - sy)) + c > 0) \land$$
$$\forall n.(n >= 0 \land n <= 1 \rightarrow$$
$$a * (ex + n * (wx - ex)) +$$
$$b * (ey + n * (wy - ey)) + c > 0)$$

C. Dynamics

This model is very similar to that of a non-moving barrier, except for a few key differences. The first is the fact that the W joint is not allowed to rotate, and safe operation is achieved on moving E and having W follow. Additionally, the line is allowed to move linearly, and this is achieved by allowing c in $ax + by + c = 0$ to evolve in the evolution according to Vc. The overview of the dynamics can be seen below:

```
{ex' = -ve*ey/re,
 ey' = ve*ex/re,
 rwx' = -vw*rwy/rw,
 rwy' = vw*rwx/rw,
 c' = -vc,
 t' = 1
 & t <= T, (ey <=0 | wy <=0)}
```

Also, it is important to note that there is an addition to the evolution domain which stops motion once both W and E are at a distance away from the line greater than S. This prevents the arms from moving into a safe region and continuing on that path back out into danger.

D. Controller

The controller is tasked with picking a suitable velocity for E to rotate with so that it will avoid an impending collision with the barrier moving towards it. Thus, first the controller allows for the nondeterministic assignment of a velocity Vc for the barrier. It then checks to make sure that this velocity is not too large and will keep the barrier below the S joint. This has to be checked because the barrier, even though its allowed to move, must still remain shy of the S joint as it is the base of the arm and can not move.

Next, a value is constructed for Ve such that,

$$Vc = b * Ex * ve/re.$$

This assignment to velocity makes sure that that the arm keeps up with the barrier.

E. Proof Methods

This model is proved in a singular case as a proof of concept, where the barrier is a horizontal line and E is to the right of S. This is the example shown in Fig. 8.

As in previous cases, the proof proceeds first by choosing a suitable loop invariant that must remain true in each and every time step of the program. The key element to this invariant is the fact that the distance to E and to W remains greater than 0. This is the same as in the non-moving barrier. The proof then branches into when $Vc = 0$ and to $Vc > 0$. When $Vc = 0$ the proof becomes trivial as neither the barrier nor E nor W are moving and thus the distance remains greater than 0.

The other branch relies on a simple argument about this distance. First, using the facts about the Ey garnered from the evolution domain, it can be showed that along the evolution of Ex, that Ex is increasing. This is done through an auxiliary variable ex0 and then cutting in and proving by invariant that $ex0 <= ex$. With this fact, the statement that the distance from E to the barrier is greater than 0 can be cut in and proved by way of differential invariant, which holds because the of the relationship between Vc and Ve. Ultimately, this proves that d', or the rate that the distance changes is 0. This fact will close this branch with a little extra work that wasnt completed in the time scope of the project.

F. Results

The ability for an arm to autonomously adjust to a barrier towards it is very powerful. It would allow for the operator to easily clear room for another arm to operate. However, this result is important for the fact that the argument is reflexive. That is, you can turn yourself around an imagine that you are controlling the robot that is below the barrier, and set a Ve and use the same approach to set a corresponding Vc that will assure that the barrier will move away at the same rate that the arm is moving towards it. Thus, this becomes a solution to a dynamically moving arm and an autonomous response by the barrier. If this result is put together with the result that was explored in this model, the created system is a simple way in which one arm can autonomously avoid another arm. Special attention would have to be paid in all of the cases in which the autonomously moving arm can no longer safely move or if the barrier has reached the other arms base. This problem, however, is easily handled by a statement at the end of controller that sets the user operated velocity to be 0 if any of the checks fail and Vc cannot move.

This approach in selecting a velocity that will, at all times, make sure that the arm is moving in a direction away from the barrier at the same rate that the barrier is moving towards it is indeed safe. However, such an approach comes with many problems and warnings. The first is that the proof attempt was using a singular case of the system, and thus the overall system would be case based and more complex. More so, significant problems arise in efficiency and in implementation. This method works well when Ex is about Re and thus, most of its motion is in the y direction. But when Ex approaches 0, the necessary velocity blows up to a very large number. This is not only inefficient, but also very impractical in implementation, and generally unsafe.

Therefore, even though the result is important and promising, the proof method needs a better argument that is not so strong. A different, and likely more useful argument, is to set a velocity that will at the very least keep the arm above the boundary after one time step. This is a different approach than previously, because it does not seek to prove that the distance between the line and the barrier is constant, but just remains positive. That is, a d could be negative, but after T seconds, that the total distance is still positive. This a less restrictive argument, but requires a good time based approximation of the circular path. This estimation is explored with parabolic approximation.

VII. Parabolic Approximation and Efficiency

A. Introduction

In many of the previously detailed models, particularly the dynamic barrier one, there is a need to fully examine how restrictive the controllers are, and to develop a more efficient approximation to the circular motion inherent in these robotic arms. This is meant to create systems that are more function, as they restrict less motion, as well as more realistic, that they make reasonable decisions in control.

B. Model

One potential solution that was examined was that of estimating the path of the arm in a parabolic fashion. The potential benefits to this approach is that it allows for the reasoning about the position of the arm according to time T but also allows for a path that resembles more closely that of the circular motion.

A specific case was examined in which the arm was to the left of S and had a positive velocity. I also assume that there is a barrier beneath the arm. Then in order to prove the functionality of this approximation, the result desired was that of $Yapprox <= Yactual$, where Y approx. is a parabola of the form $(Xapprox)^2/re + re = Yapprox$. This is illustrated in the figure below.

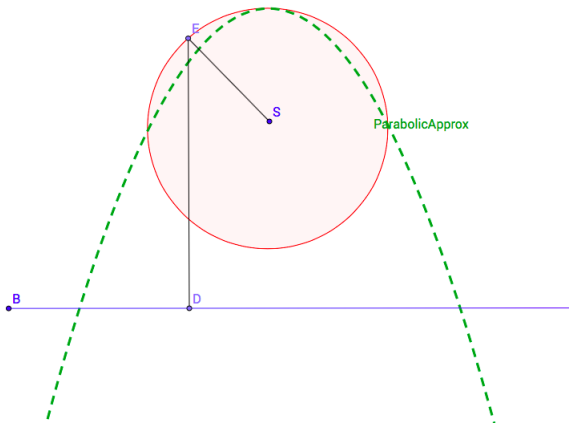

Fig. 9. Illustration of one potential means of using parabolic estimation for circular motion.

Xapprox is modeled as $Xapprox = X0 - Ve * t$ and using a differential cut and invariant shown that $Xapprox <= X$. Together, with cuts about the relative motion of the arm, It was shown that in this case $Yapprox <= Yactual$.

C. Study

In order to understand how significant, if any, the improvement in efficiency is, a study was undertaken on the relative effectiveness of each controller.

D. Results

The proof results do suggest that parabolas, with x-components approximated in a linear fashion, can be used an approximation of the arm. The potential benefit from this an improvement in efficiency. However, there is a tradeoff in complexity, and using this parabolic estimation would indeed increase the system complexity.

An adoption of this new approximation would indeed change the details of all of the previous proofs. However, the approach and overall argument would remain the same and the controller would instead use statements that are based on a relative position on the parabola rather than assuming the arm moves linearly towards the barrier.

VIII. Conclusion

Complexity, in hindsight, turns out to be very difficult to avoid. In the beginning of this project, a lot of time was sunk into coming up with coming up with a clever algorithm that would make two robots safe in a 2D space. However, with multiple rotating joints, and with many possible cases for an unsafe event, simplifying assumptions are imperative.

Only once the project was stepped all the way down to its most simple form, was progress made. It was in the slow upwards progression that the nuances of the situation became clearer. Initially, I had intended to construct proof arguments separately for avoiding obstacles and avoiding arm intersection. In working on the projects first few iterations, it became obvious that the using overt specification with multiple joints requires too many cases to be manageable. It was then that the projects focus pivoted.

After that point, I wanted to try and make a systematic and simple method that could control the arms without relying on too many specifications. That is when the boon of using controller-defined boundaries became apparent. However, even simple proofs were very time-consuming, and left a lot to be wanting in the exploration of more efficient controllers and in better models.

In the end, most of the goals in the proposal were achieved for the two dimensional model. The three dimensional model was not explored mainly for time constraints and interest in the 2D properties yet to be proved. However, these goals were achieved indirectly through a barrier system, and can be used to reason that the overall system would be valid and safe.

IX. Deliverables

This project has produced the following items:

- static_obstacle.key (with proof) and static _obstacle_2
  Static avoidance of an obstacle in 2D using linear approximations.
- barrier_avoidance.key (with proof)
  This model also deals with the same robotic arm configuration but this controller is developed to allow two arms to operate in the same space. This is done by the controller setting a line of the form ax+by+c=0 at every time step. A linear reduction of this distance is used as an argument for keeping the arm above the line at all times. An identical arm would use the same argument to

show that it remains below the barrier at all times, thus allowing for safe operation of multiple limbs.

- dynamic_barrier.key (with proof)
  Here, the idea is to provide functionality to the system by allowing the controller to move the barrier toward the arm, and have the arm autonomously set the velocity needed to remain above the arm. This is done by assuring the direction away from the barrier is at least equal to the velocity of the barrier.
- parabola_approx.key (with proof)
  In an examination of the conservative nature of the linear reduction, a parabolic estimation of the particles path was undergone. Here, it is shown that a parabola constructed at the point furthest away from the barrier can be used to estimate the distance traveled by the arm.
- advanced_barrier.key
  This model better represents the interactions of the joints. Here, a rotation matrix which remains invariant between the direction vectors for each arm and is used to provide relative motion for W as E rotates.
- full_safety.key
  This model expresses in dL how the conclusions reached between two arms staying on either side of a barrier can be used to show that there will be no intersections between the arms.

REFERENCES

[1] Bodner, J., et al. "First experiences with the da Vinci operating robot in thoracic surgery." European Journal of Cardio-thoracic surgery 25.5 (2004): 844-851.

[2] Zorn, Kevin C., et al. "Da Vinci robot error and failure rates: single institution experience on a single three-arm robot unit of more than 700 consecutive robot-assisted laparoscopic radical prostatectomies." Journal of Endourology 21.11 (2007): 1341-1344.

[3] Platzer, Andre. "Differential dynamic logic for hybrid systems." Journal of Automated Reasoning 41.2 (2008): 143-189.

[4] Platzer, Andre, and Jan-David Quesel. "KeYmaera: A hybrid theorem prover for hybrid systems (system description)." Automated Reasoning. Springer Berlin Heidelberg, 2008. 171-178.

[5] Kouskoulas, Yanni, Andre Platzer, and Peter Kazanzides. "Formal methods for robotic system control software." Johns Hopkins APL technical digest 32.2 (2013): 490.

[6] Fu, King S., Rafael C. Gonzalez, and CS George Lee. Robotics. McGraw-Hill, New York, 1987.