31st Annual International Conference of the IEEE EMBS
Minneapolis, Minnesota, USA, September 2-6, 2009

Safety Design for Medical Robots

Peter Kazanzides

Abstract— **The use of robots in medicine is increasing, leading to the call for specific safety standards. This is a challenging endeavor, however, because the patient must usually be placed in the robot's workspace and the medical staff must frequently interact with the robot. Although specific safety standards for medical robots do not yet exist, there are several medical device standards and well-established principles of risk analysis and safety design that can and should be applied. This paper presents a tutorial overview of safety design for medical robots, starting with a discussion of high-level safety requirements, followed by methods for risk assessment (or hazard analysis) and a brief discussion of some sample safety strategies.**

I. Introduction

Robots were introduced to the manufacturing floor more than 40 years ago, and today there are accepted standards for industrial robot safety, such as ANSI R15.06-1999. In contrast, robots were first used in surgery about 20 years ago, and were not widely used in this field until the last decade. Although there has been prior work in medical robot safety (for example, see [1], [2], [3], [4], [5], [6]), there is not yet an accepted standard. This is likely due to the fundamental differences between the safety requirements for industrial robots and medical robots [7]. In an industrial setting, safety systems typically involve gates, pressure-sensitive mats, and flashing lights – devices designed to keep people out of the robot's workspace or to shut down the system if a person comes too close. This is especially important when the robot is capable of high speeds or torques. In an industrial robot, high speeds and torques are desirable because they reduce the cycle time, thereby increasing the robot's productivity. In addition, many industrial robots require super-human strength to perform their tasks. Unfortunately, these desirable attributes increase the potential danger to human beings.

In medicine, it is rarely possible to keep the robot away from people. This is especially true in surgical applications, where the robot is directly interacting with the patient's body and is often working alongside the Operating Room (OR) staff and within reach of life-sustaining medical equipment. Thus, we have the situation where people *must* be in the workspace of the robot and where one of them (the patient) is usually anesthetized and cannot escape if the robot misbehaves. Furthermore, the robot may be holding a dangerous instrument, such as a scalpel, and is supposed to actually *injure* the patient with this instrument. In this scenario, robot safety involves keeping the robot under control (no "robot runaway"), working safety within its environment (not bumping into the OR staff or other medical equipment), and "injuring" the patient in precisely the right places.

P. Kazanzides is with the Dept. of Computer Science, Johns Hopkins University, Baltimore, MD USA pkaz@jhu.edu

Although there are not yet any accepted standards for medical robot safety, it is neverthless prudent to apply good engineering design concepts and absolutely necessary to satisfy regulatory requirements and general medical device standards such as IEC 60601 and IEC 62304.

II. Safety Requirements

The first safety consideration is whether the system must be *fail-safe* or *fault-tolerant*. A *fail-safe* system is allowed to fail, as long as failure causes it to enter a safe state. In contrast, a *fault-tolerant* system must continue to operate even in the presence of failures. Obviously, it is much easier to design a fail-safe system than a fault-tolerant system. Fortunately for medical robot designers, a fail-safe system is often sufficient because the robot can generally be brought to a safe state by powering off the motors; the medical intervention can then be completed via the conventional (manual) method. Of course, this assumes the existence of a manual method – as robotics becomes more advanced and enables surgeries that are not currently possible, it may become necessary to develop more fault-tolerant systems.

The second consideration is the magnitude of error that can be tolerated before a safety response is initiated. This is important because in many cases, a safety violation cannot be detected until a specified threshold is exceeded. To illustrate this, consider that most robots consist of multiple joints, where the position of each joint is set by a feedback control system; for example, the feedback could consist of an encoder mounted on the motor shaft (Fig. 1). As will be discussed in the next section (Risk Assessment), a "robot runaway" condition could be caused by several factors, such as an encoder failure or an amplifier failure. A common method of control (i.e., safety feature) is to place a maximum value (threshold) on the error between each joint's commanded position and its measured (encoder) position. Due to the nature of feedback control, this error is almost never zero; thus, the error threshold must be set to some non-zero value, E (see Fig. 2). Furthermore, most control systems are digital, with a fixed sampling period ΔT, so it is possible for the robot to travel $V_{max}\Delta T$, where V_{max} is the maximum velocity, before the runaway condition is detected. Finally, the robot has mechanical inertia and cannot be stopped immediately, leading to an additional travel of ΔP_{off}. So, the maximum error is given by:

$$E_{max} = E + V_{max}\Delta T + \Delta P_{off} \quad (1)$$

With many current robot systems, it would not be unusual for E_{max} to be several millimeters. The question of whether

978-1-4244-3296-7/09/$25.00 ©2009 IEEE

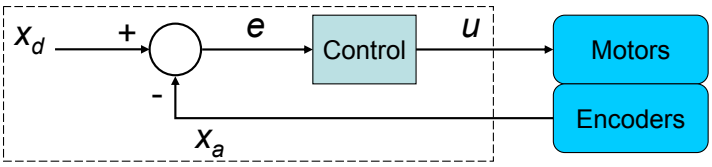

Fig. 1. Robot joint controller

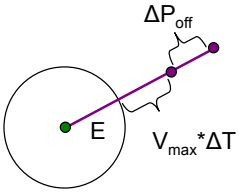

Fig. 2. Error analysis

or not this is acceptable depends on the application, so it is critical to involve domain experts (surgeons) in the safety design process. In some areas, such as orthopaedic joint replacement, a rare "glitch" of a few millimeters may be acceptable even if sub-millimeter accuracy is required for the overall system. In other areas, such as neurosurgery, this magnitude of error may not be tolerable. These safety considerations motivated some researchers to develop passive robots that are not capable of powered motion [8], [9].

III. Risk Assessment

Once the general safety parameters have been identified, it is necessary to perform a Risk Assessment (also called Hazard Analysis). This is one step of an overall Risk Management process, as required by ISO 14971 (Application of risk management to medical devices). Here, the most common tool is a Failure Modes Effects Analysis (FMEA) or, better yet, a Failure Modes Effects and Criticality Analysis (FMECA), both of which are covered by IEC 60812. Henceforth, the term FMEA will be used to refer to both methods. The FMEA is a bottom-up analysis, where the (potential) system failure is determined for each possible component failure [10]. Methods of control are devised to mitigate the hazards associated with these failures. The information is generally presented in a tabular format. The FMECA adds the "criticality" assessment, which consists of three numerical parameters: the severity (S), occurrence (O), and detectability (D) of the failure. A risk priority number (RPN) is computed from the product of these parameters; this determines whether additional methods of control are required. The FMEA/FMECA is a proactive analysis that should begin early in the design phase and evolve as hazards are identified and methods of control are developed. Essentially, it is an iterative process, starting with an analysis of the risks for the initial system design, followed by the addition of methods of control where necessary, and then a re-evaluation of the risks for the improved system, including risks associated with the methods of control.

As an illustrative example, consider again the robot joint controller shown in Fig. 1. The error, e, between the desired position x_d and the measured position x_a is computed and used to determine the control output u that drives the motor. An encoder failure will cause the system to measure a persistent steady-state error and therefore continue to drive the motor to attempt to reduce this error. An amplifier failure can cause it to apply an arbitrary voltage to the motor that is independent of the control signal u. The controller will sense the increasing error and adjust u to attempt to compensate, but this will have no effect. Finally, a processor failure could cause the robot motor to continue to move based on the last commanded voltage or current.

These failure modes are shown in the FMEA presented in Table I. The result in all of these cases is that the robot will move until it hits something (typically, the *Effect on System* is more descriptive and includes application-specific information, such as the potential harm to the patient). This is clearly unacceptable for a surgical robot, so methods of control are necessary. One obvious solution, shown in Table I, is to allow the control software to disable the motor power, via a relay, whenever the error, e, exceeds a specified threshold. Other methods of control include a redundant position sensor and a watchdog, which are described further in the following section.

Another risk assessment method is the Fault Tree Analysis (FTA), standarized in IEC 61025, which is a top-down approach that traces each system failure down to individual components. Thus, the FTA is most useful for after-the-fact analysis and is often presented graphically, using logic symbols. Figure 3 depicts an FTA that corresponds to the FMEA presented in Table I. The bottom row contains the *basic events* that, under certain conditions, can cause the *top event* (robot runaway) to occur. If the probabilities of failure are known, these can also be associated with each link. Note that the graphical representation of the FTA makes it abundantly clear that, in this case, the top event cannot occur unless at least two basic events occur (i.e., there is no single point of failure).

IV. Safety Design

This section presents some practical approaches to safety design for computer-assisted surgery systems, continuing with the example of the robot joint controller (Fig 1).

The risk analysis (FMEA in Table I and FTA in Fig. 3) identified three potential hazards, for which three methods of control were developed – these are illustrated in Fig. 4. In this example, it is assumed that the robot can always be brought to a fail-safe state by turning off the motor power.

The *redundant encoders with software check* addresses the hazard associated with an encoder failure. Here, the solution is to introduce a second encoder[1] and use software

[1]although the term encoder is used, any position sensor would suffice.

TABLE I
SAMPLE FAILURE MODES EFFECTS ANALYSIS (FMEA)

Failure Mode	Effect on System	Cause	Method of Control
Incorrect feedback	Incorrect robot motion	Encoder failure	Redundant encoders with software check
Uncontrolled motor current	Incorrect robot motion	Power amplifier failure	Tracking error software check
Robot continues previous motion	Incorrect robot motion	Processor failure	Watchdog to disable power

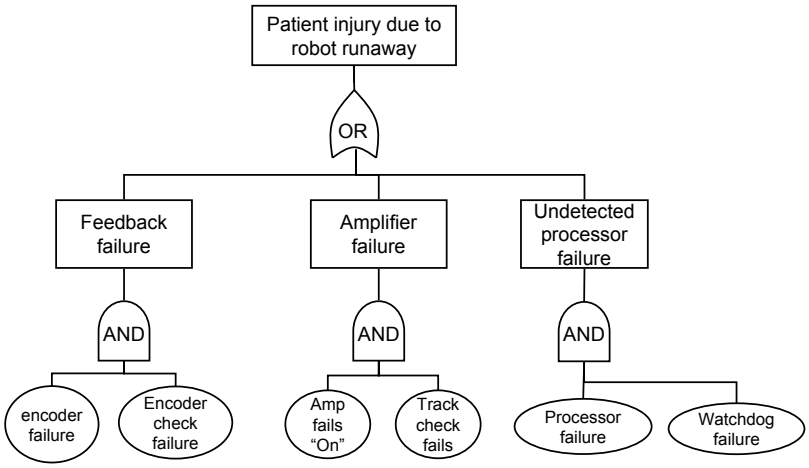

Fig. 3. Sample Fault Tree Analysis (FTA)

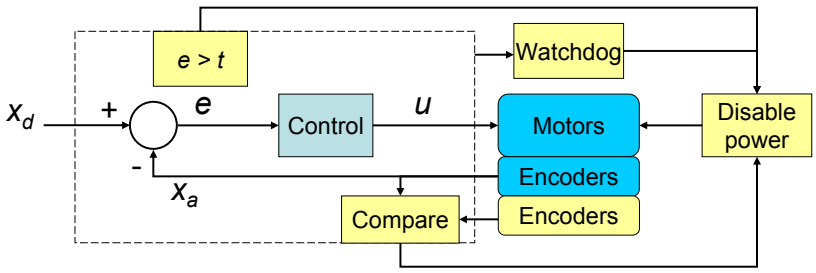

Fig. 4. Robot joint controller with safety features

to compare the readings from the two encoders. Practical considerations dictate the need for a tolerance to account for factors such as mechanical compliance between the sensors and differences in sensor resolution and time of data acquisition. If the software detects a difference between the encoders that is greater than this threshold, it powers off the robot motors. Note that although redundant encoders remove one single point of failure, it is still necessary to avoid a single point of failure in the implementation. For example, if both sensors are placed on the motor shaft, they cannot account for errors in the joint transmission (e.g., due to a slipped belt).

The *tracking error software check* was briefly described in the previous section. It is a check in the control software that ensures that the error between the commanded position x_d and the measured position x_a remains within the specified threshold, t; otherwise, the software powers off the robot motors. Again, practical considerations influence the choice of the threshold, t. Typically, a robot is moved along a trajectory by periodically making small adjustments to the desired position x_d and relying on the joint controller to reduce the error (i.e., to "chase" the point). Thus, it is necessary to set the threshold at least as high as the maximum incremental change of desired position, which corresponds to the maximum commanded velocity. This threshold can be quite large, thereby reducing the effectiveness of this safety check. One practical solution is to alter the threshold based on the robot's current mode of operation. The simplest example is to define two thresholds: a "loose" (large) threshold when the robot is moving and a "tight" (small) threshold when it is not.

The *watchdog to disable power* is a well-known method

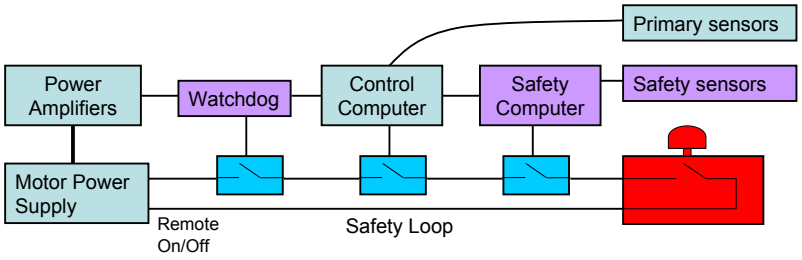

Fig. 5. Implementation of a "safety loop"

that guards against processor and software failures [11]. Generally, it is an external hardware device that must be periodically refreshed; otherwise, it disables motor power. Many medical devices rely on software for proper operation as well as for many safety features (such as the two described above), so it is critical to assure that the software is actually executing.

A common theme in the above safety features is the ability to bring the robot system to a fail-safe state by turning off the motor power. Obviously, it is best to avoid a single point of failure in the "Disable Power" block shown in Fig. 4. Thus, a typical implementation is to use a "safety loop," such as the one shown in Fig. 5, which consists of three relays in series with an *emergency power off* switch (e.g., the familiar red mushroom cap). This particular implementation relies on a motor power supply that only enables power when its two *remote power on* terminals are shorted. Thus, motor power can be turned off by opening any relay or switch in the safety loop.

As a final point, it is important to realize that a redundant system is not truly redundant unless individual failures can be detected. For example, assume in the above example that when a processor or watchdog detects a safety violation, it opens its relay and also sends a "power off" message to its peers, who then respond by opening their relays. This redundant action is not a bad idea, but it can mask the failure of up to two relays. If failure of redundant components is not detected, the result is a system with a single point of failure. For example, assume that the first two relays in Fig. 5 have failed in the closed (shorted) state. The system will appear to operate correctly because the one remaining relay will disable motor power whenever a safety violation is detected. That one relay has become a single point of failure.

V. Conclusions

Although safety considerations are important for both surgical and industrial robots, the problems and solutions are different. In an industrial setting, safety frequently involves preventing human beings from entering the robot's workspace. In the surgical setting, an anesthetized patient is inside the workspace and often physically attached to the robot. Members of the surgical team also enter the workspace and physically interact with the robot. Thus, safety systems must prevent injury even in the event of component failures or human error. The functional requirements for medical robots (e.g., proximity to patient and medical staff) are somewhat at odds with the requirement for a high level of safety. This makes it difficult to develop general safety standards that can be applied to all types of medical robots.

In the mean time, it is prudent to follow well-known standards and methods for designing safe medical devices, which include performing a risk analysis (e.g., FMEA/FMECA) and identifying and eliminating single points of failure. This paper presented a brief overview of these topics, with an illustrative example that considered and addressed certain hazards associated with a robot joint controller. This work is certainly not an exhaustive treatise on the subject; in fact, some critical issues, such as software validation and human factors design to avoid unsafe usage, have not been addressed at all. It is hoped that some day these standards will emerge, so that mankind can safely reap the promised rewards of medical robotics.

References

[1] R. Taylor, *et al.*, "Taming the bull: safety in a precise surgical robot," in *IEEE Intl. Conf. on Adv. Robotics (ICAR)*, Pisa, Italy, Jun 1991, pp. 865–870.

[2] P. Cain, P. Kazanzides, J. Zuhars, B. Mittelstadt, and H. Paul, "Safety considerations in a surgical robot," in *Biomedical Sciences Instrum. 29: Proc. 30th Annual Rocky Mountain Bioengineering Symp.*, San Antonio, TX, Apr 1993.

[3] B. Davies, "A discussion of safety issues for medical robots," in *Computer Integrated Surgery: Technology and Clinical Applications*, R. Taylor, S. Lavallée, G. Burdea, and R. Mösges, Eds. MIT Press, 1995, pp. 287–296.

[4] W. Ng and C. Tan, "On safety enhancements for medical robots," *Reliability Engin. and System Safety*, vol. 54, no. 1, pp. 35–45, 1996.

[5] U. Laible, T. Bürger, and G. Pritschow, "A fail-safe dual channel robot control for surgery applications," *Safety Science*, vol. 42, pp. 423–436, 2004.

[6] K. Fodero, H. H. King, M. J. Lum, C. Bland, J. Rosen, M. Sinanan, and B. Hannaford, "Control system architecture for a minimally invasive surgical robot," in *Medicine Meets Virtual Reality (MMVR) 14*, Long Beach, CA, Jan 2006, pp. 157–159.

[7] P. Kazanzides, B. Mittelstadt, J. Zuhars, P. Cain, and H. Paul, "Surgical and industrial robots: Comparison and case study," in *Proc. 1993 Int. Robots and Vision Automation Conf.*, Detroit, MI, Apr 1993.

[8] M. Peshkin, J. Colgate, W. Wannasuphoprasit, C. Moore, R. Gillespie, and P. Akella, "Cobot architecture," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 4, pp. 377–390, Aug 2001.

[9] O. Schneider and J. Troccaz, "A six-degree-of-freedom passive arm with dynamic constraints (PADyC) for cardiac surgery applications: Preliminary experiments," *Computer Aided Surgery*, vol. 6, no. 6, pp. 340–351, 2001.

[10] R. E. McDermott, R. J. Mikulak, and M. R. Beauregard, *The Basics of FMEA*. Quality Resources, 1996.

[11] R. Kilmer, H. McCain, M. Juberts, and S. Legowik, "Watchdog safety computer design and implementation," in *RI/SME Robots 8 Conference*, Jun 1984.