Behind the screen it happens: from DSS to TSR

Kees M. van Hee
Department of Mathematics and Computer Science
Eindhoven University of Technology
k.m.v.hee@tue.nl

1. Background

In the eighties of the last century Jan Karel and I where very much interested in Decision Support Systems (DSS). We both had a background in operations research, Jan Karel in combinatorial models and I in stochastic models. We both had a strong interest in applying OR techniques in practice. Before 1980 computers where used by OR experts to find an optimal solution to some planning problem, but only the output was communicated to the planners. This was fine for strategic planning problems with a long planning horizon, but for operational planning the goals and constraints where to volatile. Due to the appearance of personal computers the concept of a Decision Support System was born: a system where the planner could update the goal and the constraints and find a new plan himself. This were in fact "do-it-yourself planning systems". Jan Karel and I where pioneers in this area. Jan Karel in his role as head of the OR group of CWI and I as managing director of the consultancy firm AKB at Rotterdam. We had a lot of experience in building DSS. We preferred the term Interactive Planning Systems (IPS) because the term DSS was misused too often according to our opinion.

In 1988 we were together involved in a DSS project at the International Institute for Applied System Analysis (IIASA) in Vienna. Jan Karel, Ko Anthonisse and I defined and published (see [1]) specifications for an IPS. It was a Resource-constrained Project Scheduling system. Later in 1994 Jan Karel and I became guest editor for a special issue of EJOR (see [2]) where the results of seven of these systems were evaluated. In the eighties each of us published several papers about DSS. One of the papers of Jan Karel e.a. (see [3]) I will never forget because of its title: Behind the Screen: DSS from an OR Point of View. I use the same ambiguity of "behind" in this paper. Paper [3] summarized the best practices of building DSS at that time. In the nineties we lost interest in DSS and we moved to other research areas.

In 2007 a common friend and a colleague in DSS, Jo van Nunen and I discussed the system of research funding: big programmes with appealing names collecting groups of scientific friends, who would do their own preferred research, independent of each other, under the umbrella of the programme. So we were thinking of a programme as appealing as a-man-on-the-moon, that could be explained in an elevator pitch, but that would contain a set of strongly related research and development projects with high potential economical value. Our solution of this problem is described in the rest of this paper. The elevator-pitch description is:

Make a system of robots that can perform daily life activities that a human can do in a normal environment, but controlled by a human at a distance.

Such a technology would have revolutionary consequences. For instance production activities can be performed at the location of the consumers or where the raw material is produced instead of moving the production to the location where labour costs are minimal.

In fact this "great idea" did not disappear and I started in 2009 a programme for Teleoperated Service Robots (TSR's). A TSR consists of two subsystems: a *slave robot* that acts in the real world and a *cockpit* from which the operator or user controls the slave robot. In our programme which takes only two years, we are building three prototypes of a TSR (called Rose-x, x=0,1,2) to perform experiments. The project is *experiment-driven*, which means that our specifications and designs are based on *use cases* and that we test them with prototypes in an *experience laboratory* to learn from experiments. The hypothesis is that you only discover problems and their solutions by performing experiments. Actually that hypothesis seems to hold. In the programme ten organizations from the Eindhoven region work closely together. One partner is ZuidZorg, which is the largest organization for home care in the Eindhoven region. They offer the experience laboratory. We have chosen home care as application domain because there is a great need for productivity improvement in this domain.
We try to use as much as possible available components and software. We specifically use open source Robot Operating System (ROS), originally developed at Stanford University and now in hands of the spin-off company Willow Garage (see [4]). The status of our project with movies of the experiments can be found in [5]).

TSR is a challenging field and there are several similarities with the DSS field of the past:

- The operator should plan the activities of the slave robot and be able to adapt an existing plan to new circumstances. There are hard mathematical problems and soft goals and constraints. For instance the operator is not able to specify exact coordinates of places to move to.
- Planning requires mathematical models, even optimization methods from OR.
- The user interaction is extremely important: the operator should obtain all relevant state information in an easy-to-read format, should be able to enter plans himself, to generate plans automatically, to evaluate plans and to update existing plans in execution.

So actually the cockpit contains a DSS! There is also an important difference: The slave robot acts in the real world: actions are not printed on paper or screen like in a DSS, but are sent to actuators that execute them. So there really happens something behind the screen. There is feed back from sensors, showing that the slave robot did not do exactly what was planned. So updating the plan is necessary. This all happens in real time, which means that the cockpit has to react fast. Since the slave robot is acting in a daily life environment there are several safety issues.

2. What are Teleoperated Service Robots?

Telemanipulation is probably the oldest form of robotics (see [6]). Telemanipulation enables a person, called operator, to act remotely as if the operator was on the spot, by copying the manipulations of the operator at a distance. Telemanipulation is in fact restricted to grasping, moving and releasing physical objects remotely. In this way it is for instance possible to write a letter at a distance by grasping a pen and paper, fixing the paper and moving the pen over the paper. A natural extension of telemanipulation is to transform the human movements, by extending or shrinking the distances, or to increase or decrease the forces executed by the operator. In this way the operator can not only copy his actions at a *distance*, he is also able to act at a different *scale*, e.g. with higher precision, or at larger reach or with more force. Probably the oldest form of telemanipulation is a pantograph, with was used to copy images at a different scale. Typical examples of telemanipulation are found in aerospace and in medical surgery.

Telemanipulation is one of the basic functions of a TSR. But a TSR has more advanced functionality. In order to compete with a human on the spot, the TSR system should be able to perform tasks fast which means that the operator should be able to give a simple command to perform a complex task. In fact a TSR needs a lot of autonomy in task performance. Before we study a TSR in more detail, we remark that the TSR field differs

fundamentally from the field of *industry robots*. The field of industry robots is mature and the most well-known applications are in the automotive industry. Compared to the TSR the industry robots look much more advanced. The big difference is that industry robots are operating in a completely controlled environment that is often designed for them. To program them, only the *kinematics* of the system are important: the control is completely determined by the coordinates of a position of the robot or its arm. For example the robot arm moves fast (often in an optimal way) exactly to a given position. An industry robot can perform an arbitrary sequence of complex tasks as many times as we like. But a TSR moves in an unknown and an adapted environment and the operator is not able to give coordinates.
For example consider the movement to a door that must be opened: the operator sees the door via a camera on a screen and he has to give a command to move to the door, to grip the door handle, to move that handle downwards and to pull (or push) the door. This (complex!) task can not be commanded by providing coordinates. There could be obstacles in the room that were not there when a similar task was performed before, so we can not replay the same set of instructions from the past, as is done with industry robots. So industry robots are *numerically* controlled by a *program* for very specific tasks in a completely known environment and they operate *autonomously*, while a TSR acts in a completely unknown environment, on the fly controlled by an operator, who provides *non-numerical* commands. Therefore the field of TSR differs fundamentally from the more classical field of industry robots.

A TSR consists of a cockpit and a slave robot. The slave robot consists of three components: a *mobile platform*, a set of *arms* (one, two or even more) each one equipped with a *gripper* and a *vision system*. The cockpit is an integrated set of devices that enables the operator to control the slave robot. Seen from a different perspective the cockpit is also a robot, but one with a "human behind the screen" or formulated differently a "human in the loop". In fact, the slave robot is like in telemanipulation, only able to grasp, move and release physical objects.
In a basic TSR the operator has to demonstrate the actions to be executed by the service precisely, maybe at a different scale. In advanced TSR's the operator has a high-level *command language* in which he can order a complex task for the service robot with a simple command. Such a command can be given to the master by means of advanced input devices such as gloves, joysticks with haptic feedback or by voice recognition. An even more advanced TSR is able to learn behaviour from past behaviour, *programming by example,* and by operator training which is in fact *supervised learning*.

We assume however that our service robot acts in unknown environments, so the system has no map of the environment and there is neither an external navigation system with beacons, nor cameras in the environment that pass information to the TSR. We also do not assume "common knowledge" in our system, so the system does not know that a door turns on hinges and how it should be opened with a door handle.
We use the term *service robot* because the TSR replaces the human by performing tasks supposed to be done by humans, sometimes in environments where humans do not like to work, such as in dangerous environments.
There is another class of robots, called *humanoids,* that differs from TSR. Humanoids are used to *imitate* humans, while TSR's are used to *replace* humans. A humanoid should look like a human and it should be able to imitate emotional contact with humans, e.g. by producing a smile, while TSR's only try to take over human actions at a distance and possibly at a different scale, with the least possible effort of the operator.
We expect that TSR's need to have some autonomous behaviour for specific domains. On the other hand it seems to be an illusion to be able to program autonomous robots to perform all types of tasks a human can do. One reason for this is that a human can perform many tasks governed by the unconsciousness, like moving the pedals of a bicycle or applauding with two hands. This is tacit knowledge. Trying to make this knowledge explicit and transfer it into programs for an autonomous robot, seems to be infeasible. Therefore human control will be necessary for all service robots that act in an unknown environment. Hence we foresee that the gap between autonomous robots (like industry robots and humanoids) and TSR's will disappear in the long run: it will always be needed to control robots with a master.

3. What is the economical impact?

There are in principle many application domains for TSR's:

- *Care* for elderly and disabled people. Here we can have two modes of application: the person who needs the care is operating the system himself, or there is another caretaker at a distance who controls the system. Of course it is possible to switch between these two modes. An example of the first mode is some user that is not able to walk and the service robot picks up the news paper from the mail box. An example of the second mode is that a caretaker at a distance helps in waking up a person with a cup of tea.
- *Cure* such as surgery has two potential modes of use. One is telemedicine, where the surgeon is at a distance, for instance on a cruise ship without a surgeon but with a service robot on board. Also the high

precision TSR is important for care, for instance in heart, eye or brain surgery.

- *Maintenance* of equipment or installations requires service engineers to be on the spot. Preventive maintenance can be planned, but in many cases equipment breakdowns require fast repair and for these cases the travelling time to the spot is a bottleneck that can be diminished using TSR. Also maintaining equipment in difficult places, like clean rooms, and in dangerous environments can be improved using TSR.
- *Security* in buildings or compounds is done mainly by means of personal inspection and security cameras. There are great opportunities for TSR here: both to observe objects from unforeseen positions as well as for preventing people to do something or even to arrest them. Also the disarming of explosives is an example in this class of applications. (Today's robots are only used to inspect explosive bombs.)
- *Manufacturing* is done these days at locations where labour is relatively cheap and not at the location where the final products are needed or where the raw materials are won. With TSR it is in principle possible to have factories "manned" with TSR and operators at a great distance. In a 24 hours economy we could have operators on several locations on earth, each working in its own day time. Also in places where manufacturing is done in an unpleasant environment, we could use TSR, e.g. in a slaughterhouse. A TSR is able to work on a different scale, which might be a specific advantage here.
- *Agriculture* is often very laborious and the margins are so small that guest workers are imported from low-labour-cost countries. In principle it is possible to make a dedicated machine for each kind of harvesting, but economically that is infeasible. With TSR it is possible to let them stay in their own living environment and let them work at a distance, like in manufacturing. Since the TSR are general purpose devices, they can be used for other tasks by other people after the harvesting period. So TSR rental could be a profitable business.

TSR's are a special class or robots, different from industry robots and humanoids which both are autonomous robots. One of the unique features of TSR's is that the operator is in the loop which enlarges the scope of applications dramatically compared to autonomous robots. The operator in the loop does not mean that the operator has to be at the same location as the slave. In fact by teleoperations we have *decoupled* the location where a human resides and the location where he performs a task. This is a fundamental forward in technology, comparable with the decoupling of the location where energy is produced and where energy is consumed: electricity is produced in a power plant and we are using it in our homes. This

decoupling in energy production and consumption created an industrial revolution. We expect that the decoupling of the location of creating an activity and the location where the activity is performed could create a similar *industrial revolution*.
Another important feature of TSR's is acting at a different *scale* as the operator. So at a larger scale the TSR might have longer arms, move faster or it can carry more weight. At a smaller scale the TSR can manipulate objects at the micro level with high precision.
Since a TSR is a system that can in principle act in all situations where a human can act, TSR's are multi-purpose devices. This means that they are in principle suitable for *mass production*, which implies that they can become affordable for a large variety of applications.
We envision new industrial activities when TSR's will really make a breakthrough. First of all there will be companies that produce the hardware components of TSR's, such as motion platforms, arms, vision systems, and interface devices for the master. Secondly a lot of software has to be built and maintained. Thirdly the integration of all components has to be done by system integrators. So they will together produce the complete TSR systems. Then there will arise a new type of *services* that might be called consultancy. This concerns the deployment of TSR's in the various application domains. This requires training of users (operators), adapting business processes and also programming the TSR's for specific tasks. Last but not least there will be companies that maintain TSR's and companies to let out TSR's.

4. Challenges

In this programme there are two kinds of challenges: scientific challenges and engineering challenges. Since many techniques and components are available today, one of the main challenges is to integrate them effectively and efficiently in order to reach our goal. Combining existing pieces into a new product is a creative activity. In fact authors arrange existing words in a new way to produce a novel and chemical engineers arrange existing atoms to create new molecules.
We emphasize the *scientific* challenges here. For the state-of-the-art of techniques see [5]. We discovered by our experiments open problems. We list some of them.

- *Computer vision.* The operator has to navigate the slave robot through an environment and he has to position the gripper to grasp, shift or release objects. The operator can see by means of cameras on the slave robot where the slave is moving and in particular where the arm with the gripper is moving. It can also see objects in the environment. But the problem is that the camera images are projected on a 2-dimensional screen and so we do not see depth, which is a serious handicap in

grasping objects. It is possible to use stereo cameras and to display an image such that with special goggles it is possible to see a 3 dimensional picture. However it is not very practical to use these goggles and the precision is also not very good: if you see that the gripper is right on top of the object that should be grasped, you are not sure that this also in reality. There is a huge amount of algorithms for computer vision mainly for autonomous systems. So there are algorithms to make a 3-dimensional computer image of an environment using stereo cameras, maybe augmented with laser scanners. But these images are not easy to interpret by human operators. So the challenge here is to combine the direct 2-dimensional images with the computed 3-dimensional images in order to give the operator fast and accurate insight in the position of the slave in its environment. The combination of real and computed images is called *augmented reality*.

- *Simultaneous control*. For industry robots there are all kinds of control algorithms to move to grippers to the right position. But our first problem is that we do not have precise positioning information and in many cases we have feedback via the operator. A more challenging problem is that for certain tasks we need to control both the arm and the mobile platform to perform a certain task. For example for opening a door it could be necessary to move the arm the gripper of which is holding the door handle and simultaneously to move the platform backwards to give the door the space to open. Another example of simultaneous control is when we have to move a plate (for instance with some cups on it) with two hands. Then we do not want to move both arms (grippers) independently, but we want to direct the centre of the plate to a particular position and the control algorithms should take care of the fact that the plate moves exactly horizontally in de chosen direction. In both cases we have to combine the control of two different systems simultaneously.
- *Command language and user interaction*. The operator should be able to give his commands in an easy to learn and efficient way. The first challenge is to determine the right level of commands, which is actually the same as the right level of task complexity and the parameters for the command. The second challenge is to determine the right user interaction device to express the command and its parameters. Here we can profit from the developments in the world of computer games. A particular aspect of the commands is that they have not only input parameters but they may also produce output, which is actually the feedback from the slave robot. Of course we have visual feedback but we can also have *haptic* feedback where the operator can feel forces. We

are looking for a good (to avoid the term "optimal") combination of these elements in order to let the TSR be as fast as a human on the spot.

- *Software architecture*. We are developing a *component-based* software architecture that makes it easy to add new functions, even at run time. So when we develop for instance a new method for navigation then it should be easy to add this function to the system such that it can work in combination with the other functions. So this is the required *plug-and-play* functionality. Further we would like the software to be applicable for different configurations of the slave robot. For instance if we add an extra arm or we replace an arm by a very long or very strong arm then the adaptation of the software should be minimal. In this way we can use the software for a whole family of TSR's. So this is the required *configurability* functionality. Last but not least we require that the software architecture enables the formal proof of safety of the robots system. It is of course essential that the TSR will never damage objects or hurt persons when it is in operation.
- *Learning*. We discussed already the need for learning capabilities. For industry robots that is easy: it is just a capture-and-replay functionality. But for TSR's this is much more difficult. Consider the task of picking balls form the table and putting them in a basket. The operator is controlling this task with five red balls on the table. What to do if the next time the five balls are in a bit different position? And what to do if there are six red balls or five red ones and a blue one? The big challenge here is not to determine *how* the TSR should learn but *what* exactly should be learnt.
- *Object recognition*. It must be possible to give the TSR the command to find a coffee cup. This means that it should be able to recognize a shape that fits the characteristics of a coffee cup. There is a lot of research in this area today, but it will take some time to let the results be useful in the TSR. We may expect that in the future most daily life objects will have a 3-dimensional image in a public library. Since most objects are designed with CAD (Computer Aided Design) systems, the data are already available.

There are all kinds of *engineering* challenges. One of them is the *speed* and *reliability* of the communication between the master and the slave. Another one is to design the slave and the master in such a way that it is relatively easy to make a *family* of TSR's with different scale properties (bigger, stronger, smaller, more precise). This requires a component structure that allows assembling different variants of components in an easy way. Of course the greatest challenge is to make the TSR for a reasonable price!

5. References

Note: Jo van Nunen passed away suddenly in 2010. Parts of this text appeared also in my contribution to his remembrance book.

1. Anthonisse J.M., Van Hee K.M. and Lenstra J.K. (1988) Resource-constrained project scheduling: An international exercise in DSS development. Decision Support Systems 4.
2. Van Hee, K.M. and Lenstra J.K. (1994) A comparative study in DSS development. European Journal of Operations Research vol. 79, nr 2.
3. Anthonisse J.M., Lenstra J.K. and Savelsbergh M.W.P (1988) Behind the screen: DSS from an OR point of view. Decision Support Systems 4.
4. Willow Garage (2008), http://www.willowgarage.org.
5. Rose (2010), http://www.robot-rose.nl.
6. Siciliano, B. and Khatib, O. (2008) Handbook of robotics. Springer-Verlag Heidelberg. ISBN: 978-3-540-30301-5.