JacksonBot – Design, Simulation and Optimal Control of an Action Painting Robot

Michael Raschke[1], Katja Mombaur[1,2], and Alexander Schubert[1]

[1] Interdisciplinary Center for Scientific Computing, University of Heidelberg
Im Neuenheimer Feld 368, 69120 Heidelberg, Germany
michael.raschke@gmx.de, kmombaur@uni-hd.de,
alexander.schubert@iwr.uni-heidelberg.de
[2] LAAS-CNRS, Université de Toulouse
7 av du Colonel Roche, 31077 Toulouse, France

Abstract. We present the robotics platform JacksonBot which is capable to produce paintings inspired by the Action Painting style of Jackson Pollock. A dynamically moving robot arm splashes color from a container at the end effector on the canvas. The paintings produced by this platform rely on a combination of the algorithmic generation of robot arm motions with random effects of the splashing color. The robot can be considered as a complex and powerful tool to generate art works programmed by a user. Desired end effector motions can be prescribed either by mathematical functions, by point sequences or by data glove motions. We have evaluated the effect of different shapes of input motions on the resulting painting. In order to compute the robot joint trajectories necessary to move along a desired end effector path, we use an optimal control based approach to solve the inverse kinematics problem.

Keywords: Action painting, robotic tool to generate art works, algorithmic motion generation, dynamic motion.

1 Introduction

In this paper we present JacksonBot - an algorithmic robotics platform established to explore the interdisciplinary research field between Graphical Arts, Robotics, Dynamical Motions and Mathematics. The painting style of the robot is inspired by the Action Paintings of the famous American artist Jackson Pollock (1912 - 1956) who dripped and splashed color on the canvas.

The purpose of this platform is not to generate a completely autonomous art robot but rather to set up a complex tool that an be used and easily programmed by a human artist to generate action paintings. With respect to classical computer generated art, the introduction of the robotic system as a tool between computer and art work represents an interesting addition due to its ability to move in a dynamical fashion.

There is a strong relationship between arts and motions. While the motion itself is the main focus in performing arts such as dancing, it also plays an important supporting role in fine arts, since the resulting artwork heavily depends

F. Huang and R.-C. Wang (Eds.): ArtsIT 2009, LNICST 30, pp. 120–127, 2010.

on the artists motion during its creation. This is particularly true for Action Paintings.

The robot presented in this paper relies on a combination of different techniques which are connected in an modular software and hardware environment. The platform consists of a robotic manipulator arm which is installed upside down in its working environment and carries a color container. This is completed by a color supply system, different input devices and a computation and visualization environment. The generation of robot arm motions is based on algorithmic techniques. Different possibilities to enter desired end effector motions are provided, such as specifying the motion by mathematical functions in the Cartesian space, or by a sequence of points. Alternatively motions can be freely prescribed by the user via a data glove. The computation of robot joint trajectories necessary to move along any of these desired end effector paths follows an optimal control based approach to solve the inverse kinematics problem, using efficient numerical techniques. These precise algorithmic components of the approach are complemented by more arbitrary and stochastic effects such as speed-related tracking errors, elastic oscillations of the robot arm and of the color container, and unpredictable behavior of the splashing color. These random effects make the paintings of our robotics platform more human-like than classical robotics art.

We have used the robot platform to investigate the effect of different shapes and types of input motions on the resulting painting. This also allows us to explore the relationship between mathematics and art and to find mathematical criteria or input functions that lead to a visually pleasing output in terms of the resulting painting. This can serve as a basis to implement more autonomous art robots.

Several authors have worked in the interdisciplinary field linking robots, art and motions. One of the first to combine robot motions and art was the Swiss artist Jean Tinguely [1], the founder of the so-called kinematic art, who became particularly famous for his kinematic fountain installations in Basel and Paris. The Strandbeest art works of the Dutch artist Theo Jansen [2] are immense mechanisms that move efficiently along beaches, only powered by wind energy, in a very similar way as passive-dynamic walking robots. During the last decade there has been a variety of robots producing some kind of visual arts. Several robots such as HOAP at EPFL [3] and [4] in Karlsruhe are capable to autonomously draw realistic portraits of people using on their vision systems. There are other robots that generate paintings with independently chosen styles and colors [5]. The annual exhibition "Artbots Robot Talent Show"[6] features many different art robots, e.g. the pneumatic robot gossamer-1 which splashes color powered by sound signals, and has some resemblance to the robot presented in this paper. There also is some relationship to the computer-controlled rolling balls or moving pens that create the Algorithmic Art of Hébert [7].

The remainder of the paper is organized as follows. In section 2, we briefly present the robotics platform JacksonBot. Section 3 describes the formulation of the inverse kinematics problem to generate robot arm motions and its solution

via numerical optimal control. In section 4 we present three different paintings generated by JacksonBot. In section 5 we finally give some conclusions about past and future work.

2 The Robotics Platform

In this section, we briefly describe the hardware and software environment of the art robot platform JacksonBot which can be considered as a complex and powerful tool to generate art works guided by a user. It is a first test platform that has been designed with a very small budget in order to validate the concept. But since modularity and flexibility were important guidelines during the design process it will be easily possible for us in future work to replace e.g. the simple manipulator arm by a more sophisticated mobile robot with feedback, or to add other ways of desired motion input.

As shown in figure 1, the current platform uses a six DOF (degree of freedom) robot manipulator arm which is installed upside down on the ceiling of its protected working environment. The canvas is positioned on the floor below the robot. Every joint of the robot is driven by a servo motor and can be independently controlled by prescribing its angular history. Potentiometers were attached to the four main axes which allows to record executed trajectories and analyze tracking errors. An interface module connects the servo motors and the potentiometers to the PC via micro controllers. A small color container is attached to the end effector of the robot with the possibility to swing freely. A small pump guarantees a continuous refill of the color in the container through a flexible polymer hose.

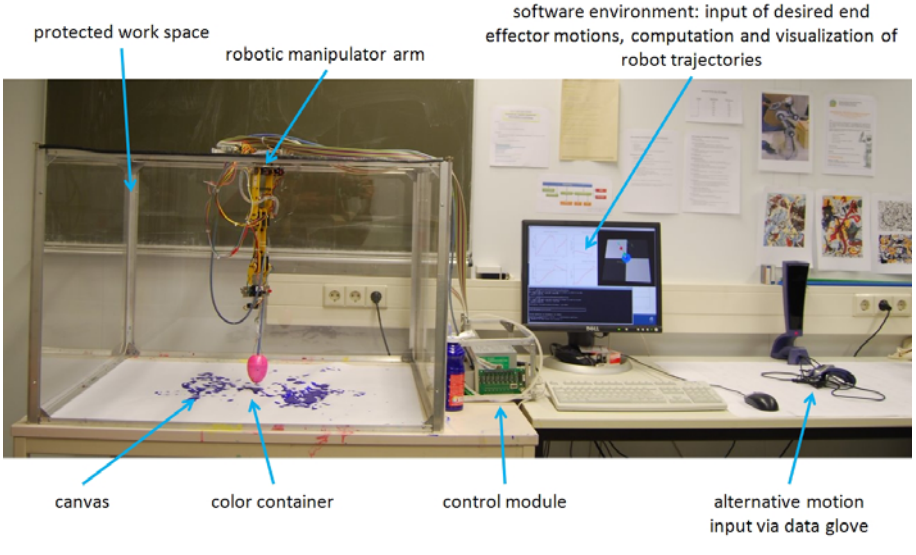

Fig. 1. Hardware and software environment of the JacksonBot art robot platform

Desired end effector motions can be prescribed either by mathematical functions, by point sequences or by data glove motions. A library of mathematical plot functions supports the user during the generation of desired trajectories. From simple shapes like lines, circles and other geometrical standard forms to very general curves, all types of trajectories can be programmed and added to the library. It is also possible to request the end effector to run through a defined sequence of points at given times. In addition, a P5 Data Glove allows a direct transfer of human hand movements to the robot arm which makes it possible to compare the mathematical descriptions of trajectories with free human-like motions.

Once a desired end effector motion is specified, it has to be translated to trajectories of the individual robot joints by solving an inverse kinematics problem. This issue is handled automatically on the PC via efficient numerical techniques, and we will describe this part in more detail in the next section. Once the resulting joint motions are determined, they can be prescribed to the servos via the micro controllers. In an OpenGL visualization (see figure 3) all desired end effector motions and the resulting computed motions of the robot arm can be displayed on the screen prior to execution on the real robot.

3 Optimal Control Based Solution of Inverse Kinematics to Generate Robot Motions

The inverse kinematics problem for a robot manipulator arm consists in determining the combination of joint angles that results in a prescribed end effector position[1]. Depending on the DOF of the robot arm and the requested situation this inverse kinematics problem may have a unique solution, multiple solutions or no solution. The problem of moving the end effector along a prescribed trajectory could be solved by studying the inverse kinematics independently for a sequence of positions along this trajectory.

However, we choose a different approach based on the solution of an optimal control problem which handles the full trajectory at once instead of pointwise. This has the following two advantages:

- In the case of redundant solutions for individual time points it avoids jumps between different families of solutions from one time point to the next, since appropriate measures guarantee that the overall joint trajectories are as smooth as possible and that the change in joint angle per time is as small as possible.
- In the case of no existing solution of the inverse kinematics problem (since the requested positions are outside the work space) it is still possible to compute the best possible solution within the workspace since the cost function requires a best possible approximation, and not a zero distance.

[1] Note that in some cases it also may be interesting to additionally prescribe end effector orientations. In the case of the JacksonBot platform with its freely swinging color container it however only makes sense to prescribe end effector position histories and no orientations (which might be different e.g. for a brush or a pencil).

We do not need to explicitely formulate the inverse kinematics equations, but can use a model of the forward kinematics of the robot end effector positions

$$z_E(t) = f(\phi(t), p), \quad z_E \in \mathbb{R}. \tag{1}$$

as a function of its joint angles ϕ and the robot parameters p such as segment lengths. The function $f(\cdot)$ essentially describes the sequential rotations and translations induced along the kinematic chain up to the end effector.

We consider the angular velocities as input or control functions u_i of the model which are linked to the joint angles ϕ_i - the state variables of the robot arm - by the simple differential equations

$$\dot{\phi}_i = u_i \quad , i = 1, ...6. \tag{2}$$

As optimization cost function, we use a minimization of deviation from the desired reference end effector positions, evaluated at m discrete points. We add a regularizing term to punish sudden changes of the joint angles, i.e. large values of the angular velocities. This results in the cost function

$$\min_{u,\phi} \Psi = \sum_{i}^{m} (z_E(\phi(t_i), p) - z_{E,ref}(t_i))^2 + \gamma \int_0^T u^T u dt. \tag{3}$$

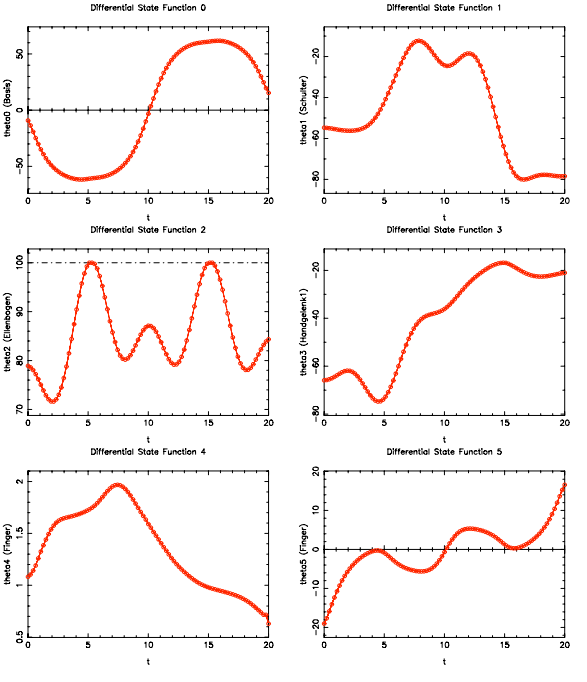

Fig. 2. Joint angle histories computed to follow the figure eight base function

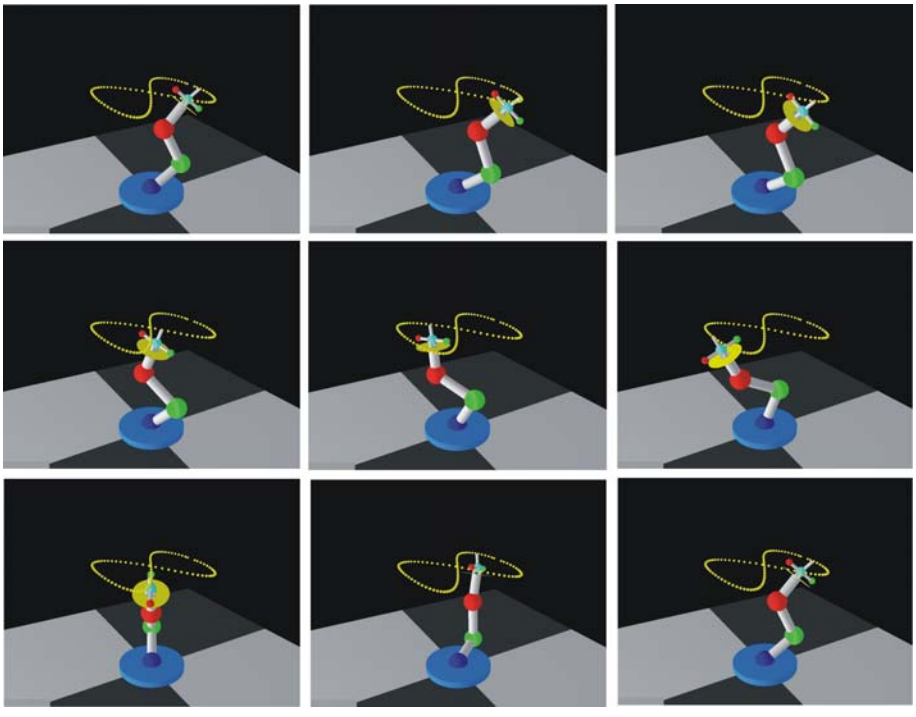

Fig. 3. Visualization of the resulting robot arm motion corresponding to figure 2

We solve this problem using the optimization techniques implemented in the optimal control code MUSCOD developed at IWR in Heidelberg [8,9]. The MUSCOD core algorithm is based on a direct method for the solution of the optimal control problem, i.e. a discretization of control variables, and a multiple shooting state parameterization. The technique which allows a fast and efficient solution of the described problems, is integrated to be called autonomously within the JacksonBot software environment.

In order to illustrate this approach, we briefly show results of an example elementary robot motion along a desired trajectory. In this case, the task is to move along a figure eight with a given length and width. This is one of the elementary plot functions contained in the mathematical library. Figure 2 shows the optimization results for all individual joint motions. In figure 3 the OpenGL visualization of the resulting robot and end effector motion is shown.

4 Example Paintings Generated by JacksonBot

In this section, we show three examples of paintings generated by the robot. Paintings are usually produced by a sequence of different motions using different colors and different strategies for defining motions.

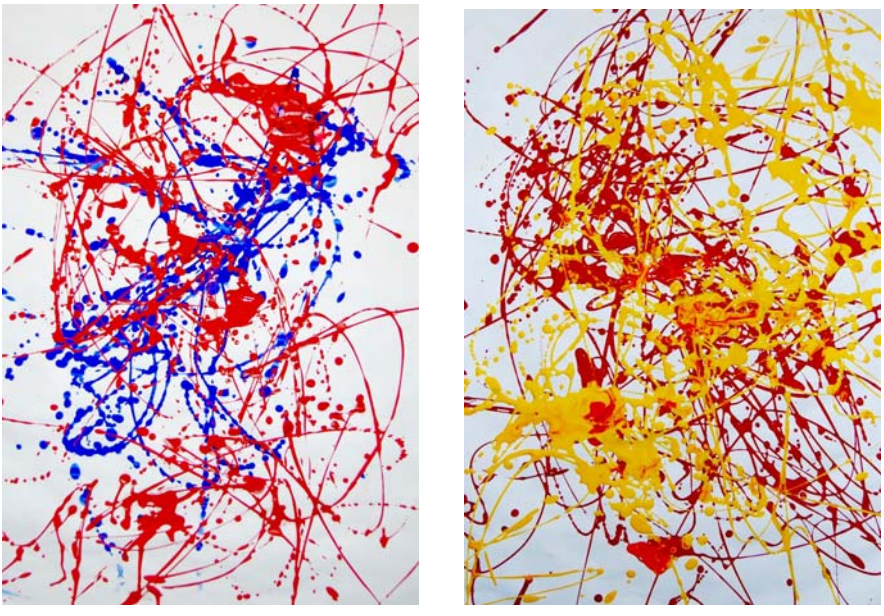

Fig. 4. Two paintings generated by dynamic motions passing through series of points

Both paintings in figure 4 were generated by specifying sequences of points to be passed at given time points. They represent a combination of very dynamic fast movements and slower ones. The dynamic movements produce more continuous lines, the slower ones leave behind more pronounced drops of color. Since the mechanics of the current robotics arm do not allow to equally access all areas, the canvas was manually rotated once during the creation of the painting.

In figure 5 the red circle was generated by a circle plot function. In the middle of the picture a blue eight can be identified. The galaxy-like shape on the left

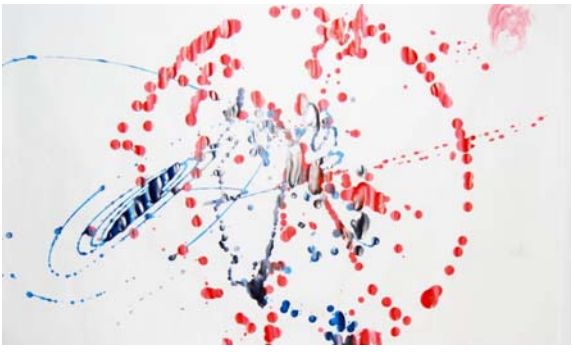

Fig. 5. Painting produced based on elementary mathematical plot functions and visible random effects

side was caused by a rapid movement of the end effector to this position over the canvas with a sudden stop. Because of the sudden stop the color container started to swing in a chaotic fashion. This is a good example for a combination of a determined, algorithmic with a chaotic and not precomputed, but random movement.

5 Conclusion

The presented robot platform JacksonBot is capable to automatically generate action paintings requiring only a few input functions or parameters. In its current state it can therefore be considered as a complex tool for the generation of art under the direction of a human user. It is possible to extend this approach and to continue from this *automatic* generation of art towards a truly *autonomous* generation of art by a robot alone without any interaction of a human. In order to do this it will be necessary to better understand the process of art generation as well as art evaluation by humans and to incorporate this knowledge in a model to be used by the robot. We have studied several different input modes and functions describing the motions but will continue this study further including evaluations of the resulting paintings by test persons. Based on these findings the robot would then be capable to autonomously select the most appropriate motions. In addition we plan to exchange the robot arm by a more sophisticated robot hardware such as a small humanoid robot that is capable to execute faster motions in a larger workspace and that also is equipped with visual sensing allowing an autonomous visual evaluation of the painting during the creation.

References

1. Wikipedia: Jean Tinguely (2009), `http://en.wikipedia.org/wiki/Jean_Tinguely`
2. Theo Jansen: Strandbeest (2009), `http://www.strandbeest.com`
3. Calinon, S., Epiney, J., Billard, A.: A Humanoid Robot Drawing Human Portraits. In: IEEE-RAS International Conference on Humanoid Robots (2005)
4. Gommel, M., Haitz, M., Zappe, J.: Autoportrait- Portaitzeichnungen aus der Hand eines Roboters (2002), `http://www.robotlab.de`
5. Van Arman, P.: Zanelle, the painting art robot, www.zanelle.com
6. ArtBots: The Robot Talent Show, `http://www.artbots.org`
7. Hébert, J.-P.: Mac-controlled Art (2009), `http://www.apple.com/science/profiles/hebert/`
8. Bock, H.G., Plitt, K.J.: A multiple shooting algorithm for direct solution of optimal control problems. In: Proceedings 9th IFAC World Congress Budapest, pp. 243–247. Pergamon Press, Oxford (1984)
9. Leineweber, D.B., Bauer, I., Bock, H.G., Schlöder, J.P.: An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization - part I: theoretical aspects. Comput. Chem. Engng. 27, 157–166 (2003)