Autonomous Chess-Playing

Karsten Will
Department of Computer Science
TU Darmstadt
`karstenwill@t-online.de`

Abstract

The dream of building a robot to help people is very old. One of the importent tasks is playing chess. A match of a chess master against a computer is reported in every newspaper and everybody is taling aboout. This paper gives a close look at autonomous chess playing and presents the winner of the AAAI-10 Small-Scale Manipulation Challenge, a robot manipulator system called Gambit. Chapter 1 deals with the historical background and the fascination of autonomous chess playing long time ago. Chapter 2 describes the competitors of Gambit at the AAAI-10 Small-Scale Manipulation Challenge in the year 2010 and presents the competition. The hardware and the software of Gambit are described in Chapter 3.
At the end of the paper Chapter 4 deals with further application possibilities, using a robot manipulator system to support the housework, for example.

1 Historical background - the turk

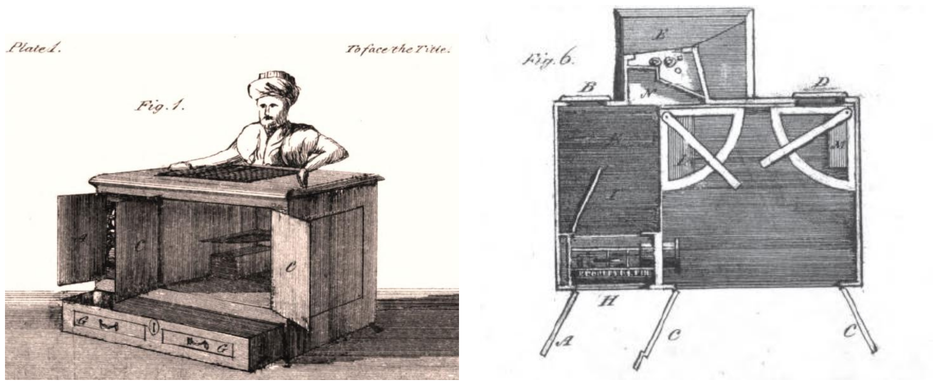


Figure 1: [2] Interior and exterior view of the turk

In the year 1809, Napoleon Bonaparte was beaten by a chess computer in only 24 moves. The founder Wolfgang von Kempelen narrated, only the very complex combination of many gears would be the secret of success. Behind a wooden box, a puppet, wearing the clothes of an turkish pasha, was sitting. Opening the two doors of this box everybody was able to see many rolls, levers and gears. At the beginning of every chess match von Kempelen winded the puppet and it was playing a game by itself. At the day of von Kempelens deth the function of the turk was still a secret. The last owner, Edgar Allan Poe, revealed the true situation in the year 1840: in the interior of the box was enough space for one person, so the turk was only a magic trick. Nevertheless this story shows the fascination autonomous chess playing was able to attract even many years ago. This is one reasonwhy playing chess is a very good example for building robots.

2 AAAI-10 Small-Scale Manipulation Challenge

The idea of the tournament was to close the gap between the fact, that programs are able to beat all human chess players and the fact that it's still a human's task to move the pieces on the board. The following chapter presents the rules and competitors of AAAI-10.

2.1 The tournament

The rules of the tournament were very simple. The robots could be placed anywhere around the board. There were used unstandardized chess boards, but with standardized pieces.

There was no size limitation for the robots and they could be controlled by a second computer off-board. Every competitor had two minutes to make its move and a game ended after 10 moves or if a robot failed to make a legal move three times. The pieces had to be placed in a legal field and every other piece (e.g. the beaten) had to be removed.

Points were given for every successful move and every captured piece. For every piece that had to be removed by the referee the robots were penalized.

Every robot could earn some extra points for detecting of misplaced pieces or the verbally announcing of their one and opponent's moves. More details can be found in [4].

2.2 The competitors

This section presents the competitors of AAAI-10. It contains a description of the construction and a part of the software per robot. Each subsection references an article for further information about the robots.

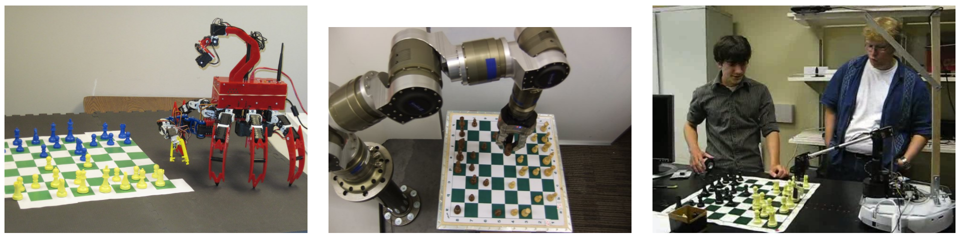


Figure 2: [4] Chiara robot, Golem robot and AL5D

2.2.1 Chiara robot

In the left picture you can see the Chiara robot, developed at the Carnegie Mellon University. The chiara is a hexadop robot: it has six legs for holonomic motion. A movable camera is mounted on top and a planar three-link arm in the front. The initial localization is done in three steps. First the average positions of all green, yellow and blue objects in front of the robot are determined. Now the robot has an approximation in which direction the chess board is oriented. After this step the camera is centered on the bottom left corner of the chess board. The third step is to extract the exact location of the bottom left corner to know the position of the chess board and the bottom boarder of the board to know the exact orientation.

To show the software, the detection of the board square occupancy is explained. This is done in two steps: the bottom of each chess piece is identified and its position saved. Additionally the positions of the board squares are extracted. Combining these two information is enought to determine the exact square occupancy.

[11] contains further information.

2.2.2 Golem

The center of Figure 4 is a picture of Georgia Tech's Golem. It's a 7-DoF arm with an three-fingered hand. Motion Grammer, a new representation kit for robot decision making, is used for the software.

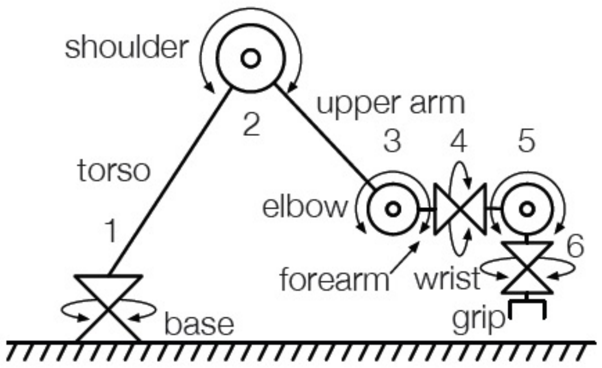


Figure 3: [1] schematic design of Gambit

The Motion Grammar represents the operation of a robotic system in a context-free grammar. The grammar is then used to generate the motion paser which drives the robot.

Further information on Motion Grammer is given in [12].

2.2.3 AL5D

The right picture in Figure 4 shows AL5D. As you can read in [5], it's the cheapest competitor. The production costs are less than $ 700, because it's an off the shelf robot with a hobby robotics arm kit. Localization of the chess board is done with a standard camera of an android phone The robot was controlled by an ubuntu-netbook and was able to drive around the chess board in order to find the best position for gripping and placing the chess pieces.

Only a little information is available on ALD5, mos of it in [5].

3 Gambit

3.1 Introduction

The simplicity of physical board games is the reason, why they are a popular problem domain for robot research. Chapter 1 depicts that autonomous chess playing is a very old task. That is the reason, why playing chess is the testbed for the Gambit system. Gambit is able to play with every kind of chess set on a large number of boards. No instrumentation or specific modeling of the pieces is required. After the opponent has made his move, Gambit detects the kind of it.

The position of the board has not to be fixed: Gambits detection system is able to track it all the time.

3.2 Mechanical Design

Gambit is a completely new designed robot. The 6-DoF arm with a parallel jaw gripper was newly developed for Gambit.

DoF is an acronym for *degree-of-freedom* and a DoF in the manipulation space is equal to a joint in the robot arm (see Figure 3). Thus, it is not possible to reach all positions with all orientations in a 3D space when using less than 6 joints. Having more than 6 joints offers more possibilities to reach a target position.

As you can see in Figure 3 the three revolute joints 1,2 and 3 are similiar to human torso, shoulder and elbow, having the function of the position control. On the other hand the joints 4,5 and 6 are used for orientation control. Gripping the chess pieces is performed with a gripper jaw with a rubber finger tip. The structure of Gambit is milled from aluminium and the costs are about $18.000.

The PrimeSense depth camera, mounted on the torso, has a working range of 0.5 m to 5 m, it's camera supplies the RGB-code and the depth for each pixel. It is mounted on torso to assert the minimum operating range of 0.5 m from the camera to the chessboard which would be undercut when mounting the camera e.g. on the shoulder.

To achieve precision during gripping oder placing a chess-piece a small camera, originally designed for the Apple MacBook, is assembled in the gripper.

3.3 Perception and Manipulation

This section describes how perception and manipulation of Gambit are implemented.

3.3.1 Locating and Checking the Chessboard

The first task is locating the chessboard and continuously updating its status. This job includes on the one finding the board and transforming it to the standard position on the other.

Ignoring the depth information of the PrimeSense camera a 2D-image of the bord is taken to locate it and to identify the chessboard pattern. After this, the depth information of the points is registered. The 2D-points are rendered into an 3D-image using RANSAC.This is robust to partial occlusions of corner points. The output of RANSAC is a plane in which the board lies. The exact location of the board is determined by adapting an 8x8 cells chessboard template to the 3D-points.

The next task is checking the board occupancy. In order to determine the occupancy of a single square, Gambit uses the point clouds above the surface of the board. A 2D projection of all pieces on the board is built by using an isometric projection of these points. The next step is clustering these points: one cluster represents an individual piece. Each cluster is mapped to one cell in the 8x8 occupancy grid defining the chess board.

If an obstacle, e.g. a hand, is detected above the board, the determination of the occupancy is paused.

The color of the pieces is determined in two steps. Before the game, the medians of the point clouds on both sides are calculated. These medians define the color black for one side and white for the other. During the game Gambit determines the best classification, black or white, for every point cloud by computing the distance to the medians from the first step.

The board state $(s_i, c_i, p_i)_t, i = 1, ..., 64$ is defined as a tuple of

- $s \in \{0, 1\}$ the binary occupancy variable; 1 if the field is occupied, 0 if not
- $c \in \{white, black\}$ the piece color
- $p \in \{K, Q, P, R, B, N\}$ for King, Queen, Pawn, Rook, Bishop and kNight, the piece type.

After defining the board state, the board occupancy and the color differences before and after a move are checked. If GNU chess does not confirm the validity of the change, the opponent is asked for correction.

3.3.2 Chess Piece Detection and recognition

As explained in section 3.3.1, Gambit demands information about the initial state of the chess board to be able to identify the chess pieces. However, this is the only restriction for Gambit to join a chess match at any stage.

Figure 4 depicts the steps of the algorithm responsible for the recognition of the specific chess pieces. Its input data is a 640x480 image of the palm-camera.

The *square detector* is used to find a specific square in the image of the palm-camera. It implements the histograms of oriented gradients (HOG) features method.

The idea behind HOG is that appearance and shape of an object can be described by the distribution of local intensity gradients or edge directions.

Figure 6 shows the steps necessary to extract features from an input image. The first step is dividing the picture into small cells. For every cell a 1D-histogramm of the gradients is accumulated using

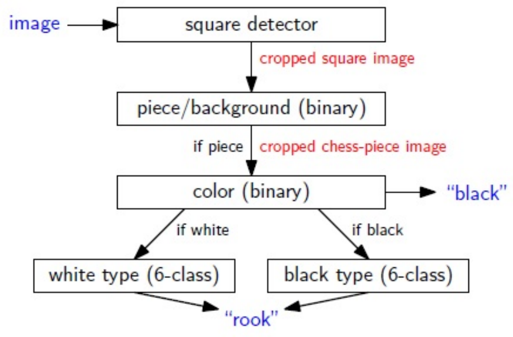


Figure 4: [1] classifier hierarchy for piece recognation

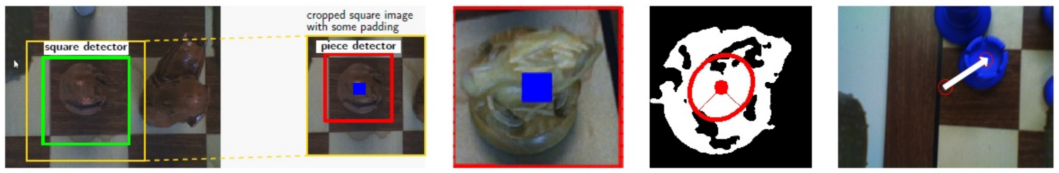


Figure 5: [1] (1,2) Chess piece, detected by the palm-camera. (3) SVM training set. (4) Principal axes. (5) Translational correction.

the gradients of the entries of a cell. The histogram entries form the representation of the picture. To achieve a better invariance to illumination, shadowing, etc., it is useful to normalize the cells before using them. To be robust to rotations, 20 square templates are generated for testing by rotating an example square. To detect a square, a score function computes values for every position in the picture. A square is detected if the function exceeds a threshold while scanning the image with a sliding window. The detected squares are bordered with a bounding box as the left picture in Figure 6 depicts.

The variety of chess boards available implies a variety of square sizes Gambit has to deal with. Additionaly its manipulator is able to approach different heights. Therefore Gambit has to be able to adapt the size of the camera image to fit the testing squares. To determine the necessary scale, a pyramid is created out of the images by scaling it with the factors $0.8 \cdot (1.05)^i, \quad i = 1, ..., 10$. The square detector is applied to each image in the pyramid. This procedure is repeated 20 times. Then, the overall most likely fitting scale scale is chosen.

To train the *piece/background detector* images from the *square detector* with chess pieces in the center are saved as positive examples and pictures without as negative examples.

On this training set, a binary linear SVM using LIBLinear, which defines the piece/non-piece detector, is applied. A Support Vector Machine (SVM) is a method to find a line, the divides the

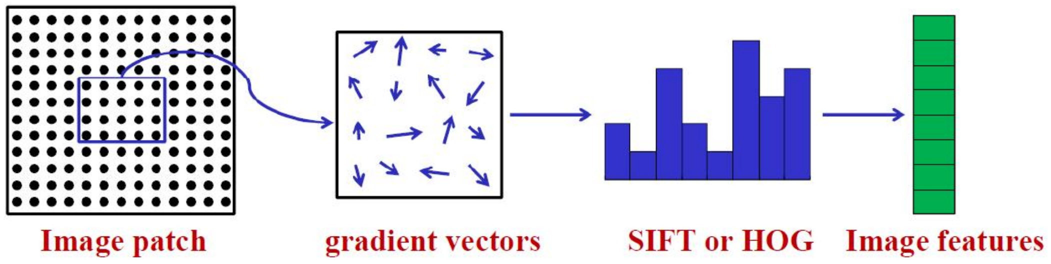


Figure 6: [10] Image processing for feature extraction.

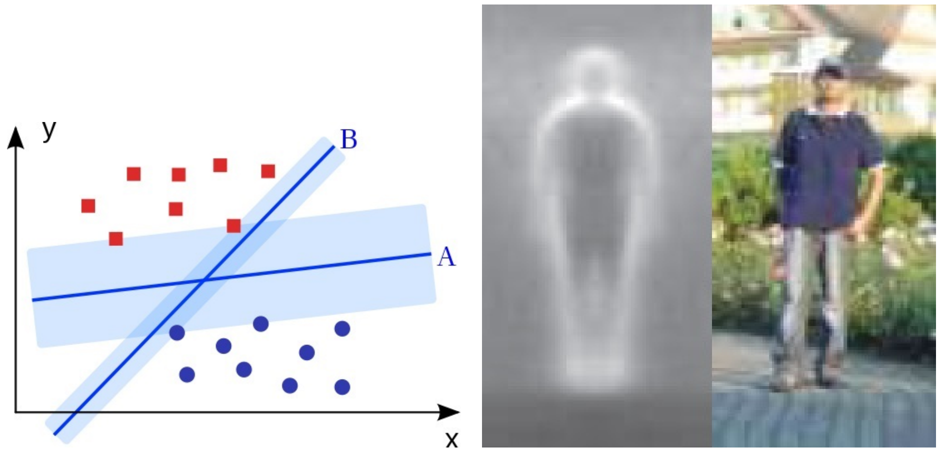


Figure 7: [13] (1) SVM example. [6] (2) Picture and visualized HOGs.

representatives of different classes in a vector room, if a representative of a class is considered as a vector. Therefore the SVM maximizes the minimum distance of all points to the line.

The function of the *color detector* is the separation of the pieces into the distinct classes black and white. This classifier is trained using the boxes created by the square detector (see Figure 5). This blue one is a training data point for white chess pieces.

At last the *chess piece classifier* is trained for each color to separate the piece types. Concatenated SIFT and kernel descriptors are used as chess piece classifiers.

Scale-invariant feature transform (SIFT) is an algorithm to find and describe local features in pictures. At first, the image is smoothed, for example with a Gaussian filter, to reduce the noise. After this, the picture is divided into local features, which are invariant to perspective distortions, transformation or lighting variations. These features are represented by histograms and stored in a vector. Kernel descriptors are another method to design rich features to capture various visual attributes. Thus obtained features can be captured with learned exampes to assign the chess pieces to the right type class.

3.3.3 Game Play and Manipulation

One move of Gambit takes about 22.5 seconds. Playing is only a cycle of recording the board state, checking for problems , performing the necessary computations and making a move. At the beginning of the game, the initial occupancy grid is built, the system computes and saves the models of the piece colors and creates a table of the heights of the different piece types. This information is used during the whole game. When an illegal move occurs, Gambit is able to ask for help and the game is paused.

During gameplay, Gambit repeats every move in spoken language. Gambit communicates the game state and problems, e.g. an illegal move performed by the opponent or dropping a piece by himself.

If Gambit captures an opponent's piece during a game, a sequence of manipulations is necessary. First, Gambit removes the opponent's piece, then it moves its piece onto the free position.

Gambit is able to play with nonstandard chess pieces, e.g. with asymmetric chess pieces, too. In this case or if a piece is not centered in a standard grap can fail. To execute local correction, Gambit utilizes the pictures of the palm camera. The pictures are used for the 2D pose and the roll angle of the manipulator, too. The features of the chess pieces, the orientation and the center, are computed using a binary image. Then, the local adjustments are computed with a loop.

The algorithm requires a picture of a single chess piece.

The algorithm starts with the computation of the longest principal axis, the center point and the orientation of the chess piece.

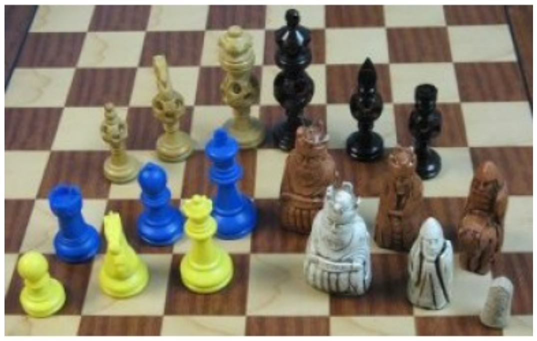


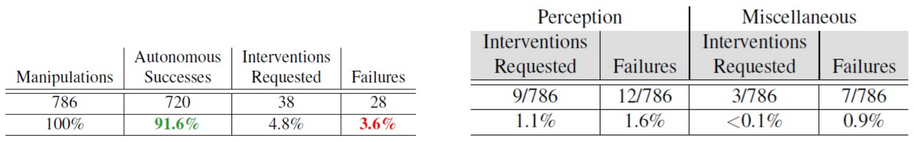
Figure 8: [1] Three chess sets

3.4 Evaluation

At the AAAI-10 Small-Scale Manipulation Challenge, Gambit played against the three other robots, and Gambit scored higher than its competitors in every match. This section analyzes Gambit's performance on three different chess pieces.

For each game, it was tracked how many interventions were ordered by Gambit and how many failures, without any request, had happened.

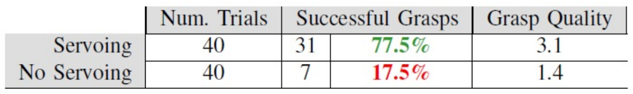
Three types of errors were differed. A manipulation error occurs if Gambit fails to grap or drop a piece or is not able to place a piece legally. Perception errors happen if Gambit is not able to find a piece on the chess board. The last error type is called a miscellaneous error. It occurs if the gripper of Gambit collides with a piece while placing or Gambit is not able to find an inverse kinematic solution for an explicit motion. The tests didn't reveal any logical errors like illegal moves or problems with the identification of the opponents move.



Manipulations	Autonomous Successes	Interventions Requested	Failures
786	720	38	28
100%	**91.6%**	4.8%	**3.6%**

Perception		Miscellaneous	
Interventions Requested	Failures	Interventions Requested	Failures
9/786	12/786	3/786	7/786
1.1%	1.6%	<0.1%	0.9%

Figure 9: [1] Left: Manipulation errors. Right: Perception errors and miscellaneous.



	Num. Trials	Successful Grasps		Grasp Quality
Servoing	40	31	**77.5%**	3.1
No Servoing	40	7	**17.5%**	1.4

Figure 10: [1] Manipulation failures.

To put it in a nutshell, the results are very good. In [1], visual servoing is suggested to improve the results because most of the mistakes were manipulation errors.

As you can see in the figure abowe visual sensoring is very important for the quality of grapping. To test visual servoing, chess pieces were placed in corners of a the board. This is the worst legal position during the game. To make it harder, pieces were placed around the test object.

Because a piece is placed by moving the gripper to the center of a chess square, the capturing of a piece affects the quality of the placement. For this reason, a valuation of the grasp was developed: from 4 points for a perfect grasp down to 1 point for a very poor one. Zero points were given for a illegal move or a lost grasp.

The last part of the testing was the piece recognition. Therefor the Lewis Chessmen set (see Fig. 8 in the left front) was used because the pieces are less stylized and have a lower contrast than the others. Figure 9 depicts the confusion matrix. The classifier achieved an accuracy of 93.22% in the tests in [1].

	K	Q	B	N	R	P	k	q	b	n	r	p
K	2											
Q		2										
B			4	1								
N				2								
R					4							
P						14						
k							2					
q								2				
b									2			
n										4	1	
r											2	
p									1		1	15

Figure 11: [1] Confusion matrix of the Lewis Chessmen set.

4 Future Work

One-armed robots are widely used. Today, nearly every production hall contains robots performing screwing, lifting or fitting in tasks. When searching google, you can even find one writing off the bible. These tasks can be fulfilled without a very difficult sensoring.

As you can see, there are many possibilities to utilize a robot without machine learning or perception of the environment. A possible application of a one-arm robot could be the kitchen to help with the housework. The power of the hand in combination with the recognition of new types of pieces could be very helpful to fill and dispel the dishwasher: You only have to show some kinds of cups, plates or forks and the robot is able to recognize many different kinds and can put them into the right position in the dishwasher.

For this task the classifier hierarchy for chess-piece recognition could be used, with some changes. At first the classifiers have to be trained once, after installation of the robot. The square detector has to be changed: there aren't squares at the worktop any more. Fields have to be defined beforehand. For training the chess-piece detector, images from the worktop and the empty dishwasher have to be served as negative and images of the plates as positive examples. There would be no need for a color detector. The chess piece classifier needs images of the various types of dishes with information how to grip a special type and were to place it in the dishwasher.

Of course this is only a rough sketch of this new application. The changes are a little more difficult and the fitness of SVM respectively SIFT would have to be reviewed. Nevertheless this example shows, that the classifier hierarchy for chess-piece recognition is chosen very well and it is possible to use the robot - with only minor changes - for various tasks.

All in all, the one-handed robot in combination with a poweful recognation algorithm is able to turn out as a very good help in the future.

References

[1] Matuszek et al.: Gambit: A Robust Chess-Playing Robotic System, ICRA 2011.

[2] Robert Willis (1821): An attempt to analyse the automation chess player of Mr. de Kempelen. Booth, London.

[3] Bernhard Koerber (2004): Der erste "'Schachcomputer"' Aus dem Leben des Hofrats Wolfgang von Kempelen, LOG IN nr.127, p. 73 to 74.

[4] http://chiara-robot.org/Challenge/

[5] http://lannyland.blogspot.ch/2009/03/chess-playing-robots-at-aaai-10.html

[6] N. Dalal and B. Triggs, "'Histograms of Oriented Gradients for Human Detection"', in CVPR, 2005.

[7] M. A. Fischler and R. C. Bolles, "'Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography"'. Comm. of the ACM, pp. 381395, 1981.

[8] L. Bo, X. Ren, and D. Fox, "'Kernel descriptors"', in NIPS, 2010.

[9] D. Lowe, ”‘Distinctive Image Features from Scale-Invariant Keypoints”’, IJCV, vol. 60, pp. 91110, 2004.

[10] Kernel Descriptors for Visual Recognition, http://books.nips.cc/papers/files/nips23/NIPS2010_0821_slide.pdf

[11] Jonathan A. Coens, ”‘Taking Tekkotsu Out Of The Plane”’, August 2010

[12] Neil Dantam, Pushkar Kolhe, and Mike Stilman, ”‘The Motion Grammar for Physical Human-Robot Games”’, May 2011

[13] http://de.wikipedia.org/wiki/Support_Vector_Machine