The Knowledge Engineering Review, Vol. 34, e18, 1 of 13. © Cambridge University Press, 2019
doi:10.1017/S0269888919000146

A sketch drawing humanoid robot using image-based visual servoing

MENG-CHENG LAU[1], JOHN ANDERSON[1] and JACKY BALTES[2]

[1]*Autonomous Agents Laboratory, Department of Computer Science, University of Manitoba, Winnipeg MB R3T2N2, Canada;*
e-mails: mengcheng.lau@gmail.com, andersj@cs.umanitoba.ca
[2]*Educational Robotics Center, Department of Electrical Engineering, National Taiwan Normal University, 129, He-ping East Road, Section 1, Taipei, Taiwan 10610, Republic of China;*
e-mail: jacky.baltes@ntnu.edu.tw

Abstract

This paper presents our sketch drawing artist humanoid robot research. One of the limitations of the existing artist humanoid robot is the lack of feedback on the error that occurs during the drawing process. The contribution of this research is the development of a humanoid robot artist with drawing error correction capability. Based on our previous work with open-loop control pen-and-ink humanoid robot artist, we have implemented a closed-loop visual servoing approach to address this problem. Our experimental results show that this approach is sufficient to correct drawing errors that occur due to mechanical limitation of a robot.

1 Introduction

Recent research on humanoid robots has devoted significant effort to developing humanoid robots that can match human behavior on high-level tasks that require integration of sensing, physical motion and intelligence. The current state of the art supports diverse applications involving a wide range of industries, including entertainment, art, education, health care, household services, competition (Baltes *et al*., 2017), etc. One of the advantages of developing a humanoid robot artist compared to a robot that specially designed for sketch drawing (e.g. a 4 d.f. robotic arm) is that a humanoid robot has the capability to adapt itself to perform a wide variety of tasks such as cleaning dishes, folding cloths, sweeping floor, etc. A humanoid robot artist requires human-specific skills which are challenging tasks for humanoid robots. Because Sketching is one of the primitive drawing techniques, many researchers (e.g., Srikaew *et al*., 1998; Olsson *et al*., 2002; Calinon *et al*., 2005; Lin *et al*., 2009; Kudoh *et al*., 2009; Lau & Baltes, 2010; Sasaki *et al*., 2016; Singh *et al*., 2017; Atoofi, 2018) have tried to develop a robust humanoid robot that could produce pen-and-ink sketches. However, there are some limitations such as the drawing task is usually slow due to complicated motion control and complexity of the input images. It requires high-quality and expensive force and torque feedback sensors such as force sensing resistors, strain gauges or load cells for drawing accuracy. And, currently, the drawing is limited to line segment, curves such as a circle will be interpreted as multiple line segments. One of the motivations of our research is to adopt affordable robotics technologies to create human-like artwork with the capability of observational drawing through visual feedback. The goal of this research is for a humanoid robot to mimic the drawing process of human by reproducing its way of drawing based on limited feedback control.

This paper presents our work on the development of a sketch drawing artist humanoid robot. The remainder of this paper is divided as follows: Section 2 provides a survey of previous research into artist robots. Section 3 describes how we develop the closed-loop visual servoing technique to correct the sketch error. Section 4 explains the experimental setup and results on various pre-defined errors. Finally, in Section 5, we draw conclusions and discuss future applications of this research.

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

2 Background

The development of artist robots has received great interest since 1990. *AARON*, one of the oldest continuously developed programs in computing history (beginning in the mid-70's), attempted to create paintings that produced human-like artworks (Cohen, 1994). Gommel *et al*. (2004) implemented an industrial robotic arm, KUKA, to draw in Cartesian space. Srikaew *et al*. (1998) created a dual-armed humanoid robot, *ISAC*, that used the control of the end-effector with force sensors in the process of drawing. Ruchanurucks and Kudoh also implemented a multi-fingered robot called *Dot-chan* which was equipped with position- and force-sensing (Ruchanurucks *et al*., 2007; Kudoh *et al*., 2009) to produce color brush drawing. Katsura and his group have developed a calligraphy robot based on their Motion Copy System (Tsunashima & Katsura, 2010) which learn the input calligraphy motion (e.g. position and pressure) to reproduce the brush strokes (Kennedy & Osuga, 2012). Tresset and Fol Leymarie created *AIKON/Skediomata* a 3-d.f. low-cost robotic platform dedicated to drawing sketches (Tresset & Leymarie, 2005, 2006; Tresset *et al*., 2011). Calinon *et al*. from École Polytechnique Fédérale de Lausanne (EPFL) used the small humanoid robot HOAP2 for their human portrait drawing task (Calinon *et al*., 2005). *Pica* was introduced by Lin *et al*. for fast human portrait drawing but its drawing results were very simple and lacked contours (Lin *et al*., 2009). Another interesting approach was *Drawbot*, a wheeled-robot with a pen-holder mounted on an arm that allowed the pen to be lifted off the drawing surface (Brown *et al*., 2005). A similar approach is used in a polargraph art robot, *Makelangelo* developed by a robotic company called Marginally Clever (2013). It used a wall-drawing approach which draws on a piece of paper on the wall with a pen that adjusts its *x*-*y* position based on two strings controlled by two stepper motors. Lu *et al*. (2009) developed a robotic manipulator called *IRAS* based on visual feedback that determines stroke positions and applies local gradient interpolation to guide stroke orientation in its pen-and-ink artwork. However, this was only in a fixed plotter-like setting. In order to improve the drawing accuracy, some researches try to implement machine learning approaches such as deep neural network (Sasaki *et al*., 2016; Singh *et al*., 2017; Atoofi, 2018). An in-depth review of these works shows that none of them used visual servoing in humanoid robots as feedback for drawing observation, and none provides drawing correction during the drawing process. This is an important ability in pen and ink sketching. In this paper, we implemented a closed-loop image-based visual servoing (IBVS) control techniques to address the drawing deviation problem of our robot.

Creating a artist humanoid robot with high-performance industrial robots (i.e. accurate and fast) usually involves a high hardware cost. It is a much more significant challenge to develop an efficient robotic system if the cost is also an important factor. Reducing the cost of the system requires optimization of all aspects to retain its flexibility, reliability and performance. During the design and development of our humanoid robot, namely, Betty, we only used affordable hardware and open-source software to address both cost (approximately USD 3000) and performance issues (Baltes *et al*., 2011). The upper body of Betty consists of a 12-revolute joint system with 12 degrees of freedom (d.f.). Its head has 4 d.f. which are pan, tilt, swing and one d.f. for the mouth. Each of its arms provides 4 d.f., shoulders allow lateral and frontal motions and elbow gives lateral motion and a wrist motion. These joints are constructed by four Dynamixel RX-64 servos in the head and four Dynamixel RX-64 servos for each of its arms as shown in Section 4. The visual feedback control is driven by an open-source computer vision library (OpenCV) which has various image processing techniques. These techniques compute a line-art output vector that can be mapped to the kinematics of Betty's arm. This research aims to develop a robust visual servoing sketch drawing system that enables Betty to autonomously draw the sketches by exploring various research areas.

Due to the hardware performance limitations, it is difficult to achieve high mechanical accuracy. One of the main motivations of our work is to develop a closed-loop approach which is significantly superior to open-loop control to overcome the lack of accuracy in our humanoid robot. This non-feedback or 'blinded' open-loop approach might not be adequate for a sophisticated implementation of a drawing robot because it does not provide sufficient feedback during the drawing process. When using an open-loop control in a robotic system, there is no online interaction between the robot and the environment where no data is collected from the robot's sensors during task execution which can be seen in

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

Franklin *et al*. (2001), Instruments (2011). At first, it extracts the information from an input image. From this, it generates a robot pose and motion sequence. This allows the control of a robot and image processing to be viewed as two independent tasks (Kragic & Christensen, 2002). One of the main disadvantages of this approach is that the robot will not be able to correct its state if there are any changes in its environment during the process (Instruments, 2011). For instance, in our earlier implementation of our sketch drawing humanoid robot, the portrait of a person is recognized and extracted, then its pixels position is computed relative to the camera (robot) coordinate system. This Cartesian space information is used to move Betty's arm to the desired position relative to the drawing pad. With the eye-to-hand camera and inverse kinematic model, the position of the pen is computed based on the visual information extracted from the camera frame. It then maps the pixels of the image to Betty's inverse kinematic model. Next, Betty executes the drawing task by performing open-loop movements which assumes that the environment remains static after the drawing has started (Baltes *et al*., 2011). However, without the real-time feedback of what has been drawn by Betty, it is impossible to make corrections to ensure the desired output sketch which is obtained at the end. Hence, we need a closed-loop control system to overcome this problem.

3 Image-based visual servoing

Visual servoing is a common closed-loop visual feedback control approach for object manipulation in robotic research, for example (Bourquardez *et al*., 2006; Vahrenkamp *et al*., 2008, 2009; Siradjuddin *et al*., 2010; Chu *et al*., 2011) and industry, for example (Hodges & Hall, 1996; Nomura & Naito, 2000). In contrast to open-loop control, a closed-loop control system uses vision as the feedback sensor to continuously track the environment, which provides estimation and updates for robot control and positioning. Therefore, the control sequence is generated based on a consistent visual feedback of the error between its current and desired poses of the end-effector. The terminology of *visual servoing* was introduced in 1979 by Hill and Park (1979) prior to the general term 'visual feedback' (Hutchinson *et al*., 1996). Generally, visual servoing techniques are classified into three types: IBVS, position-based visual servoing (PBVS) and advanced visual servoing (Chaumette & Hutchinson, 2006, 2007). The general idea of a visual servoing system is to estimate the differential changes between robot joint space and the image space with a coupled robot-image Jacobian (Kragic & Christensen, 2002).

PBVS processes the error signal from the image features in order to compute the control sequence of the robot based on the relative 3D pose between the camera and the end-effector (Wilson *et al*., 1996; De Luca *et al*., 2007). PBVS solves the 3D localization problem where 3D information of the scene, including a geometric model of the target, and the parameters of the particular camera model (camera calibration) is known. PBVS is considered to be a 3D vision sensor. Usually, there are two tasks in a PBVS system. First, as mentioned, it requires a model between camera and end-effector to estimate the pose of the target. Second, it has to estimate the desired velocity screw of the robot, which requires precise camera-robot calibration to achieve accurate positioning (Kragic & Christensen, 2002).

Image-based visual servo systems (IBVSs) were introduced by Weiss and Sanderson (Weiss *et al*., 1985; Weiss & Sanderson, 1987) to compute the pose of the robot by processing the error signal from the image features parameters, for example, lines and image areas directly as its feedback control signals (De Luca *et al*., 2007). IBVS involves the computation of the image Jacobian or the interaction matrix (Hutchinson *et al*., 1996; Kragic & Christensen, 2002). In contrast to PBVS, IBVS is considered as a 2D vision sensor. IBVS eliminates PBVS's requirement of perfect knowledge of the camera calibration matrix and its complex interpretation of image features to derive 3D world-space coordinates. Figure 1 shows the general design of an IBVS system.

In our sketch drawing humanoid robot research, a 2D image (from the drawing pad) is used directly to estimate the desired pose of Betty's arm. This is a typical task of IBVS, where the image distance error (drawing variation) between the current and desired image features in the output image plane is tracked. We therefore proposed to use IBVS to address this problem in our work. Generally, IBVS can be divided into two separate systems. First, a dynamic look-and-move system. This system performs the

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

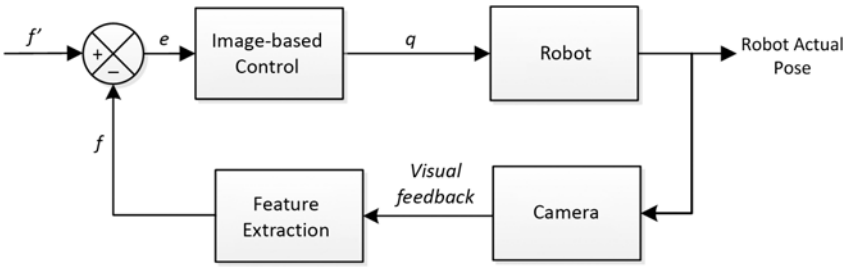


Figure 1. Block diagram of IBVS. f is the actual location of feature and f' is its desired location. The positioning task is described as location error, $e = f - f'$ and q represents the coordinate of the end-effector.

control of the robot in two stages: the vision system provides input to robot controller that then uses joint position feedback to control the end-effector. As pointed out by Hutchinson *et al.*, nearly all of the reported systems adopt this approach (1996). Second, direct visual servo systems. Here, the visual controller directly computes the input to the robot joints and robot controller is eliminated (no position feedback) (Kragic & Christensen, 2002). As discussed by Chaumette and Hutchinson, there is no strategy that provides perfect properties based on their stability issues (Chaumette & Hutchinson, 2006). However, the coarse estimations in IBVS will only imply perturbations in the trajectory performed by the robot to reach its desired pose (longer convergence time) and will have no effect on the accuracy of the pose reached. On the other hand, IBVS has several advantages over PBVS. In IBVS, a 3-D environment model is not required. Its targ*et al*ways stays in the field of view of the camera, and its performance is robust with respect to camera calibration errors and noise when a monocular camera is used (Bourquardez *et al.*, 2006; Chaumette & Hutchinson, 2006; De Luca *et al.*, 2007). Therefore, we use this image-based eye-to-hand visual servoing to control the relative pose between the drawing tool and the drawing pad with a camera mounted on Betty's head.

For the drawing task, a camera (right eye) is used as a global vision sensor (eye-to-hand configuration). It retrieves and tracks the position of the features (lines and shading) in the image relative to the Betty's inverse kinematic frame. Compared to PBVS, IBVS's great robustness with respect to calibration error can thus be expected. As discussed earlier, a visual servoing approach is classified based on the type of feedback it perceived. In our implementation, we extracted the line segments of the drawing pad's pen-and-ink sketch. Based on this information, it guides Betty during the drawing process to correct the drawn errors (e.g. line segments pose: two end-points and completeness). Hence, an IBVS approach is implemented.

IBVS involved manipulation of a set of image features (object) from an acquired image based on their coordinates by computation of the *image Jacobian* (Kragic & Christensen, 2002). In general, the Jacobian matrix (J) is defined as a matrix based on the motion of the image feature. J is used to set the desired pose of manipulator as seen in Equation (1) (Hutchinson *et al.*, 1996; Kragic & Christensen, 2002).

$$\dot{f} = J\dot{q} \tag{1}$$

Where $\dot{f}$ represents the motion of the image feature in consecutive images and $\dot{q}$ denotes the six dimension velocities, $[\dot{x}, \dot{y}, \dot{z}, \dot{w}_x, \dot{w}_y, \dot{w}_z]$: three translations and three rotations of the feature with respect to the camera frame (Chu *et al.*, 2011). Based on the unit focal length perspective projection model, the relationship between the robot arm and the image feature is shown as Equation (2) (Hutchinson *et al.*, 1996; Kragic & Christensen, 2002).

$$\begin{bmatrix} \dot{x}' \\ \dot{y}' \end{bmatrix} = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{x}{Z} & xy & (1+x^2) & -y \\ 0 & \frac{1}{Z} & -\frac{y}{Z} & -(1+y^2) & xy & x \end{bmatrix} \tag{2}$$

During the drawing task, the pen's tip is orthogonal to the surface of the drawing pad. Hence, no orientation adjustment is required during the servoing. The image features only need to be translated to

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

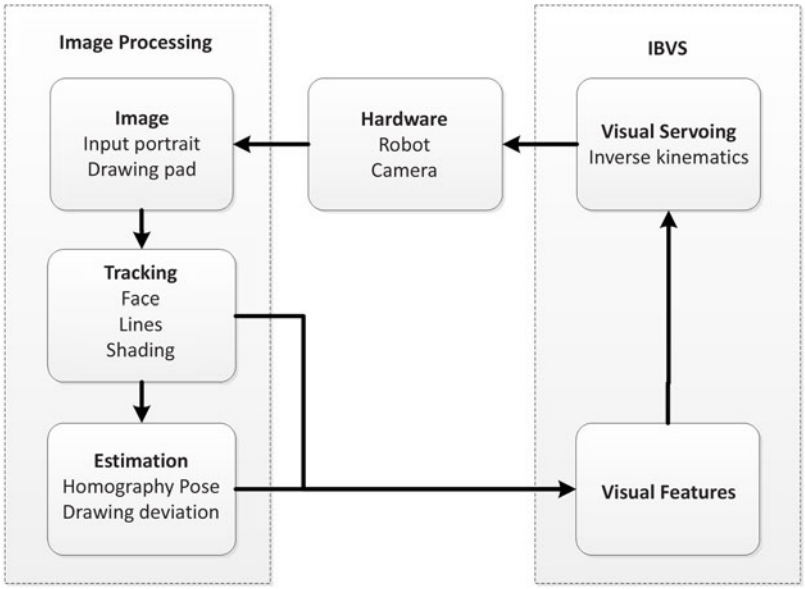


Figure 2. General framework of IBVS for drawing task.

the desired pose with the three translation $[\dot{x}, \dot{y}, \dot{z}]$ workspace. Therefore, the Jacobian can be simplified as Equation (3).

$$J = \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{x}{Z} \\ 0 & \frac{1}{Z} & -\frac{y}{Z} \end{bmatrix} \tag{3}$$

IBVS expresses the control error function directly in 2D image space (Kragic & Christensen, 2002). The aim of all vision-based control schemes is to minimize the error $e(t)$, which is typically defined as $e(t) = f - f'$ where its block diagram is shown in Figure 1. f is a vector of n visual features based on image measurements (e.g. the image coordinates of interest points, or the parameters of a set of image segments) and a set of parameters that represent partial knowledge about the system (e.g. camera intrinsic parameters or three-dimensional object models) (Siradjuddin *et al.*, 2010). The vector f' is the desired values of these features. In the drawing tasks, the simplest model for the controller is its drawing deviation. And, there must be a relationship between the time variation of each variable.

Based on Kragic *et al.*'s (Kragic & Christensen, 2002) and Chu *et al.*'s research (2011), the most common proportional control approach is applied to generate the control signal for the robot. J^{-1} is the pseudoinverse of the image Jacobian and K_p is the proportional gain. The visual servoing scheme continues until the evaluated error is less than the given threshold pixels limit on the image. Figure 2 illustrates the proposed framework of the visual processing pipeline for the IBVS system.

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = K_p J^{-1} \cdot e(t) \tag{4}$$

Let us assume the camera position is static and the height of the drawing pad is known. A number of feature lines are tracked and used to generate a vector of current measurements, f_1 and f_2. The vector of desired reference measurements is denoted f'_1 and f'_2. The error function for the drawing deviation is defined as a function of the distance between these measurements, $e = f - f'$. This error function is then updated in each frame and used together with the inverse kinematics to estimate the control input to Betty.

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

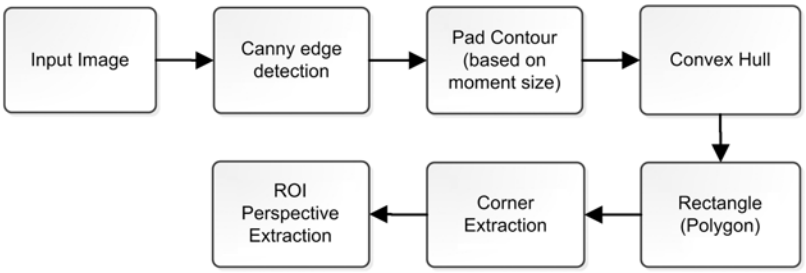


Figure 3. Flowchart for drawing pad detection.

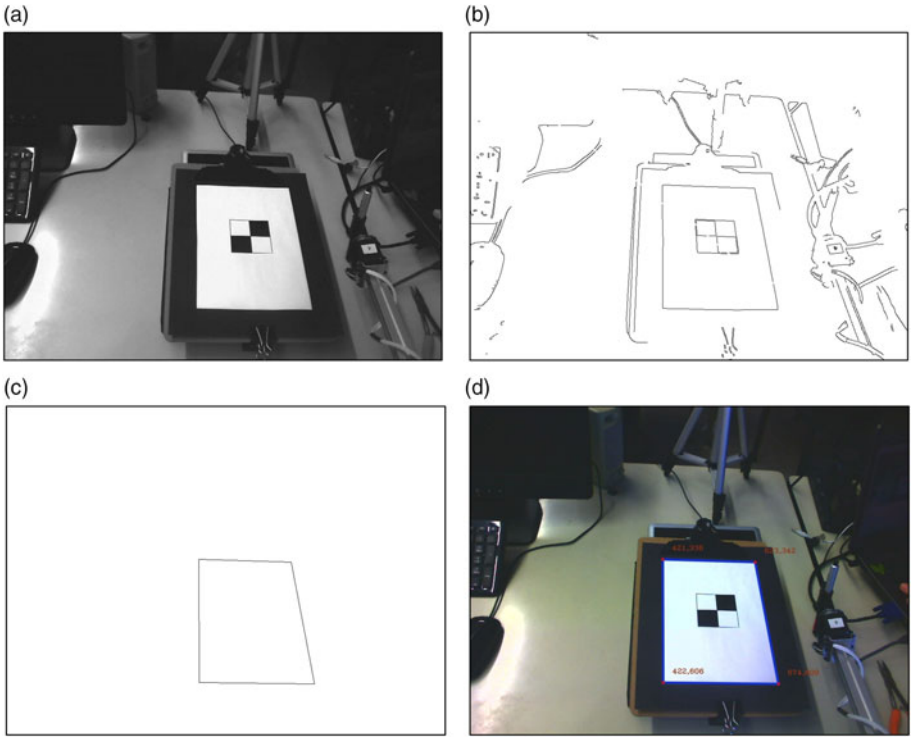


Figure 4. Image pipeline for drawing pad detection. (a) Input image; (b) Canny line detection output; (c) pad contour of drawing area is extracted; (d) four corners are estimated based on rectangular convex hull.

3.1 Drawing pad detection and sketch extraction

Drawing pad location is a crucial information in the IBVS implementation. Based on this information, it constructs the image plane on a Cartesian space to extract content on the drawing pad.

Figure 3 shows the overall process to detect the drawing pad location and to extract its content within the drawing area. First, we applied the Canny edge detector to the input image to obtain the line image as seen in Figure 4(b). With the wide range of edges in the image, drawing pad contour is extracted based on its moments size by using the *moments* function in OpenCV. It allows computation of all moments as seen in Equation (5) (Team, 2013b).

$$m_{ij} = \sum_{x,y} (array(x, y) \cdot x^j \cdot y^i) \tag{5}$$

We used a rectangular convex hull algorithm as the minimum bounding box of the rectangle drawing area to compute its four corners' coordinates in Figure 4(d). Last, based on the detected corners vertices, a perspective transformation is performed to map the deformed drawing pad (region of interest (ROI)) to its destination image. The mapping is done by *getPerspectiveTransform* and *warpPerspective* in OpenCV library (Team, 2013a). Equation 6 (Team, 2013a) shows the calculation of the 3×3 perspective

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

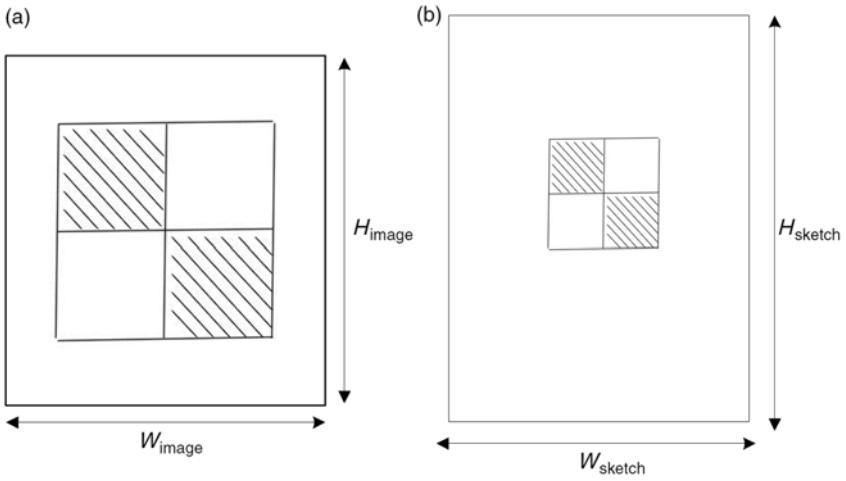


Figure 5. Image mapping from input image to sketch for drawing pad. (a) Input image and (b) mapped sketch based on drawing pad dimension.

transform matrix from four pairs of the corresponding points $i = 0, 1, 2, 3$. The destination image vertices are represented as $dst(i) = (x'_i, y'_i)$, and the source image vertices are referred as $src(i) = (x_i, y_i)$.

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = map_matrix \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{6}$$

Then, we used the *warpPerspective* function to perform a perspective transformation of a source image to a destination image using the specified 3×3 matrix, M to correct distortion caused by viewing angle of Betty right eye. Mapping of the drawing pad dimension is described in Equation (7) (Team, 2013a). Figure 4 shows an example of extracted drawing pad area.

$$dst(x, y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right) \tag{7}$$

3.2 Sketch lines matching

Sketch line matching is the process to match line segments based on geometrically possible pairs of generated sketch and actual drawn lines segments to compensate for drawing errors for the IBVS approach. This technique is adapted and extended from the edge drawing lines (EDLines) algorithm (Akinlar & Topal, 2011a, b). First, the ROI of an input image is defined and used the standard EDLines algorithm to detect line segments. These line segments are sorted by their y-coordinate. Next, line segments that lay near the edges of the sketch and short insignificant lines (based on the given threshold, e.g. line segments with fewer than 10 pixels) are removed from the line vectors. The final step is to merge all near-collinear line segments to minimize the number of significant line segments. Then, the geometry of the drawing pad is mapped to the generated sketch dimension in 2D space (x- and y-coordinates) as shown in Figure 5 with directional scaling from Figure 5(a) and (b). Its projective transformation matrix is shown in Equation 8, where $s_x = W_{\text{sketch}}/W_{\text{image}}$ and $s_y = H_{\text{sketch}}/H_{\text{image}}$.

$$\begin{bmatrix} x_{\text{image}} \\ y_{\text{image}} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{\text{sketch}} \\ y_{\text{sketch}} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x_{\text{sketch}} \\ s_y y_{\text{sketch}} \\ 1 \end{bmatrix} \tag{8}$$

Each of the line segments will be redrawn based on the correction of its position error. Errors of two end points of a line segment convey the errors of angle and length of the matching line segment to its desired baseline. It provides sufficient information to redraw the unmatched line segment. However, if the

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

Algorithm 1 Match line segments

Require: line segments from drawing pad, S_{pad} and line segments from sketch, S_{sketch}

```
Read and draw each S_sketch
for each line segments from S_sketch do
  if S_sketch and S_pad are approximately collinear then
    Compute the closest distance between S_sketch and S_pad
    if S_sketch and S_pad distance < threshold then
      Calculate two end points distances between S_sketch and S_pad, p1_dist and p2_dist
      if p1_dist and p2_dist < lower threshold then
        Flag the S_sketch as match
        Return line match
      else
        if p1_dist and p2_dist < within lower and upper thresholds
          Return the errors of end points, e_a^x, e_a^y, e_b^x and e_b^y
        else
          Return no line match
        end if
      end if
    end if
  end if
end for
```

matching is unsuccessful after several trials, the drawing process will move on to the next line segment to avoid infinite loop of single line segment matching. We use five-trial as the termination condition for each line segment.

The error function is defined simply as the difference between the current (or the closest line segment) and the desired line segment feature $e(t) = f - f'$. Based on their two end points where f is the pose of the current line segment ab and f' is its desired pose which is $a'b'$, hence, the current errors $e(t)$ of endpoints a and b in 2D space could be defined as the Manhanttan distance between them:

$$e_a(t) = dist(a, a') = [e_a^x, e_a^y] = [(x_a - x'_a), (y_a - y'_a)] \tag{9}$$

$$e_b(t) = dist(b, b') = [e_b^x, e_b^y] = [(x_b - x'_b), (y_b - y'_b)]$$

$$\begin{aligned} a_{IK}^x(t) &= a_{IK}^x(t-1) + K_p * e_a^x \\ a_{IK}^y(t) &= a_{IK}^y(t-1) + K_p * e_a^y \\ b_{IK}^x(t) &= b_{IK}^x(t-1) + K_p * e_b^x \\ b_{IK}^y(t) &= b_{IK}^y(t-1) + K_p * e_b^y \end{aligned} \tag{10}$$

Then, these errors are passed to the inverse kinematics routine to change the joint angles to the desired pose with a P controller. Equation (10) shows the proportional control function where K_p is a empirically tuned gain. The third dimension z could only be estimated based on no-touch events. The process of the line segments matching is shown in Algorithm 1. No transfer of errors from one stroke to the next. It avoids false corrections due to shifting of drawing pad during the drawing task. Figure 6 shows the closed-loop control diagram for the IBVS algorithm.

4 Experimental setup and results

In the experiments, we evaluated the feasibility of our IBVS control based implementation on their drawn outputs. Figure 7 shows the experimental setup of Betty. During the experiments, Betty used its right arm to draw. We used sinusoidal interpolation for its motion trajectory. We placed a Wacom Bamboo tablet

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

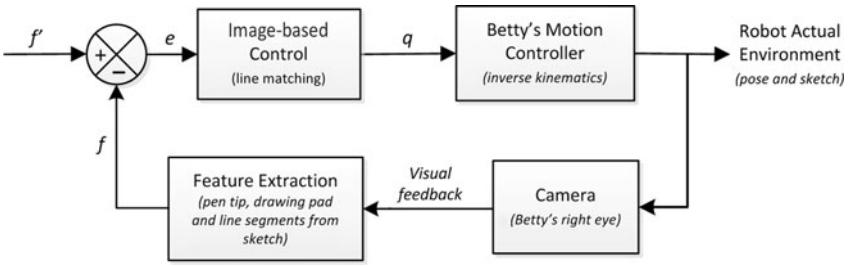


Figure 6. Closed-loop control diagram for IBVS.



Figure 7. Experimental setup using a Wacom Bamboo tablet (CTH640M), Betty, camera and pen.

(CTH670M) under the paper to measure position and pressure on the drawing pad. The measures of the tablet were only used to establish the reliability of the experiments but did not provide any feedback control to Betty. The drawing area of 217 mm by 137 mm is based on the specification provided by the manufacturer with the accuracy of ± 0.5 mm.

The efficiency of the IBVS control approach was tested based on two given x- and y- position errors of 10 and 20 mm offset from the correct line segment. Betty repeated 30 times for each stroke type of vertical, horizontal and diagonal lines based on these two offsets for a total of 90 trails in each initial offset error. Only one direction of strokes (downward and left-right) was examined in this experiment because it had no effect on visual information. For these trials, the average errors in the position for each type of stroke were measured. Next, we calculated the absolute differences of the current position of two endpoints, p_1 and p_2; and the desired position, p'_1 and p'_2 as mapped by the tablet where $e_{pi} = dist(p_i, p'_i)$, $i = 1, 2$ and $dist$ is the Euclidean distance between two end points of p_i and p'_i. We selected the proportional gain, $K_p = 0.4$ empirically during the experiment. Figure 8 shows the result if the drawing is moved during the experiment and Figure 9 shows how line segments were perceived from the camera.

Table 1 shows the number of required correcting strokes to the initial offset errors and how they converged to their desired distance from their baselines (correct line segments). The results clearly show that the 10 mm offset error was successfully corrected within two to three strokes for vertical and horizontal strokes. It reduced the error to approximately 5 mm from the desired baseline. However, diagonal strokes did not show good convergence on average. Based on our observation, it was caused by the sinusoidal drawing motion at the wrist that is likely to create curvy lines. Therefore, we used linear interpolation in

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

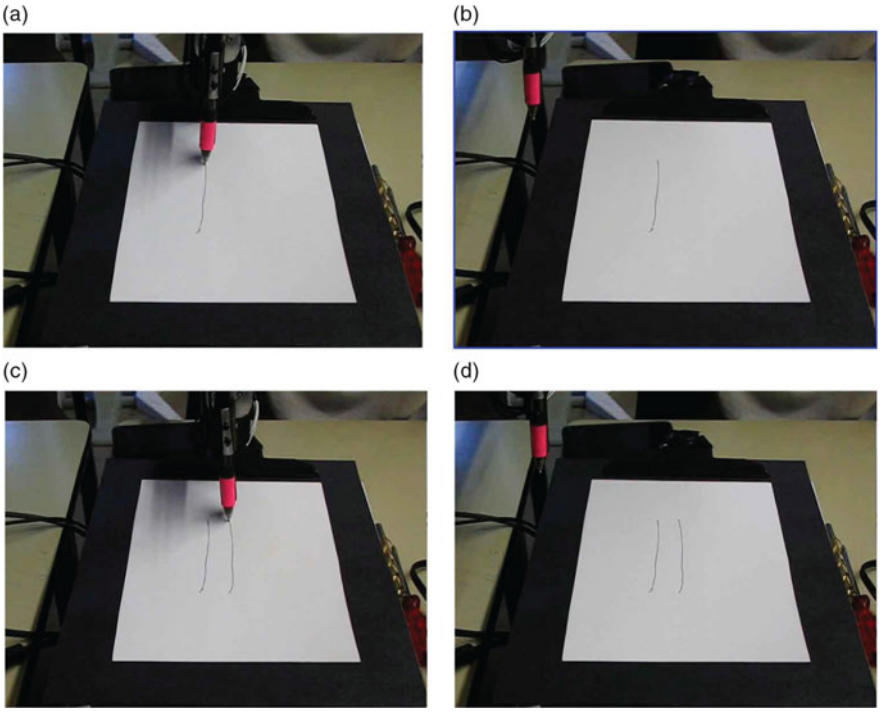


Figure 8. Example of vertical stroke error correction with IBVS control. (a) a faulty line is drawn with the given offset. (b) The hand is moved away from the drawing pad to avoid occlusion. (c and d) A correct line segment was drawn in second trial after error correction based on its line segment's feature.

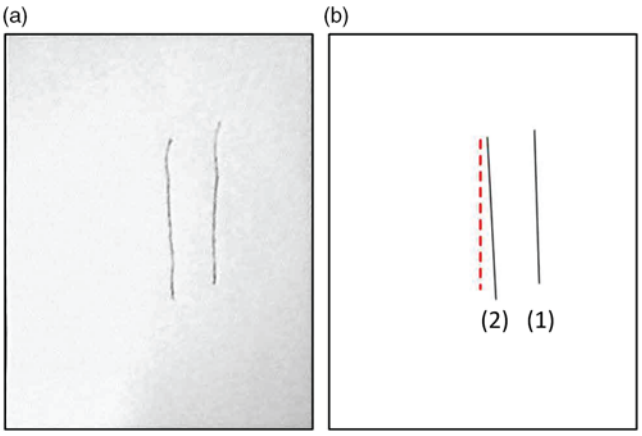


Figure 9. Line segments perceived from the camera. (a) Input ROI image as seen from camera, (b) line segments detected by eEDLines approach (1) is the first stroke, (2) is the corrected stroke and the dotted line is the desired line segment.

this matter but small vibration at the pen's tip caused by the non-smooth trajectory created jerking lines. This problem introduced significant noise to the existing line segment detection algorithm and affected the results. Similarly to 10 mm offset experiments, 20 mm offset error shows consistent convergence of vertical and horizontal strokes. However, due to the drawing motion trajectory; diagonal strokes remained harder to correct, often requiring 4-stroke and 5-stroke. In this experiment, we labeled five-stroke trails as correction failures. IBVS control approach had the failure rate of 13% and 17% in diagonal strokes compare to lower failure rate of nearly 3% in vertical and horizontal strokes for 10 mm and 20 mm offsets, respectively. Figure 10 shows the averages of trails needed to make the correction of each stroke type in both 10 and 20 mm offset conditions.

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

Table 1. Number of correcting strokes required based on the initial errors

	Number of correcting stroke							
	10 mm				20 mm			
Stroke type	1	2	3	4	1	2	3	4
Vertical	17	10	2	2	13	12	4	5
Horizontal	15	10	2	1	11	15	2	2
Diagonal	6	11	9	4	4	10	11	5

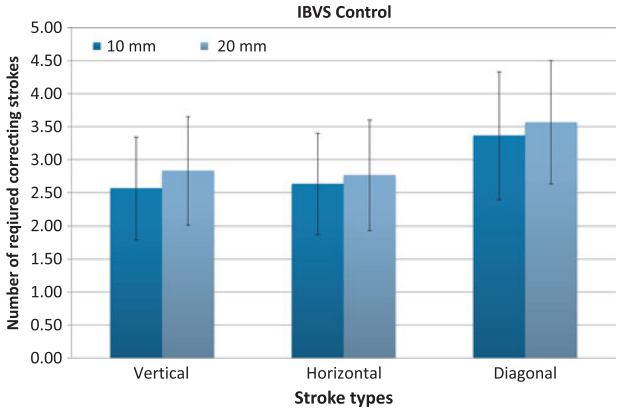


Figure 10. Number of strokes for different stroke types based on IBVS control. Error bars indicate the standard deviation.

5 Conclusion

In this work, we implemented a new approach for the drawing robot to visual servo the sketch drawing task, using an IBVS technique. The results show that this approach has successfully translated the drawing deviation into control actions within the drawing task. In our work, we used a Bamboo tablet and a series of experiments to provide the capability of drawing evaluation. This may lead to a better understanding of the various components, such as the correlation between pressure and drawing (pixels) deviation in different implementations.

As future work, we would like to improve the visual feedback of the drawing task by investigating better line segment detection algorithms. Such as, a line detection algorithm proposed by Liu *et al*. (2012) which is based on steerable filters for edge ridge detection and line fitting algorithm. Based on their experimental results, this method outperforms EDline in the evaluation of their work. Since there is a limitation in the area of diagonal line drawing motions, another extension to our work would be developing a better interpolation motion such as third-order spline fit to smooth the IK trajectory of the drawing motion which could allow the robot to draw smoother curves or circle. It would potentially prevent jerking and vibrations when a line is drawn as suggested in Calinon *et al*. work (2005).

Acknowledgements

This work was financially supported by the Chinese Language and Technology Center of National Taiwan Normal University (NTNU) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan, and Ministry of Science and Technology, Taiwan, under Grants No. MOST 108-2634-F-003-002, MOST 108-2634-F-003-003 and MOST 108-2634-F-003-004 (administered through Pervasive Artificial

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

Intelligence Research (PAIR) Labs), as well as MOST 107-2811-E-003-503. We are grateful to the National Center for High-performance Computing for computer time and facilities to conduct this research.

References

Akinlar, C. & Topal, C. 2011a. Edlines: a real-time line segment detector with a false detection control. *Pattern Recognition Letters* **32**(13), 1633–1642. http://www.sciencedirect.com/science/article/pii/S0167865511001772

Akinlar, C. & Topal, C. 2011b. Edlines: real-time line segment detection by edge drawing (ed). In *18th IEEE International Conference on Image Processing (ICIP)*, 2837–2840.

Atoofi P., Hamker F. H. & Nassour J. 2018. Learning of central pattern generator coordination in robot drawing. *Frontiers in Neurorobotics* **12**. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6064740/

Baltes, J., Cheng, C. T., Lau, M. C. & Anderson, J. E. 2011. Cost oriented automation approach to upper body humanoid robot. In *Proceedings of the 18th IFAC World Congress*, Milano, Italy.

Baltes, J., Tu, K.-Y., Sadeghnejad, S. & Anderson, J. 2017. Hurocup: competition for multi-event humanoid robot athletes. *The Knowledge Engineering Review* **32**, e1.

Bourquardez, O., Mahony, R., Hamel, T. & Chaumette, F. 2006. Stability and performance of image based visual servo control using first order spherical image moments. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4304–4309.

Brown, P., Bigge, B., Bird, J., Husbands, P., Perris, M. & Stokes, D. 2005. The drawbots. http://www.sussex.ac.uk/Users/philh/pubs/drawbots-muta-final-small.pdf [Accessed 25-July-2011].

Calinon, S., Epiney, J. & Billard, A. 2005. A humanoid robot drawing human portraits. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (HUMANOID 2005)*, IEEE-RAS, Tsukuba, Japan.

Chaumette, F. & Hutchinson, S. 2006. Visual servo control, part I: Basic approaches. *IEEE Robotics and Automation Magazine* **13**(4), 82–90.

Chaumette, F. & Hutchinson, S. 2007. Visual servo control, part II: Advanced approaches. *IEEE Robotics and Automation Magazine* **14**(1), 109–118.

Chu, H. K., Mills, J. K. & Cleghorn, W. L. 2011. Image-based visual servoing through micropart reflection for the microassembly process. *Journal of Micromechanics and Microengineering* **21**(6), 065016. http://stacks.iop.org/0960-1317/21/i=6/a=065016

Clever, M. 2013. Makelangelo 2. http://www.marginallyclever.com/blog/drawbot [Accessed 31-July-2013].

Cohen, H. 1994. The further exploits of aaron, paiter. http://crca.ucsd.edu/~hcohen/cohenpdf/furtherexploits.pdf [Accessed 13-June-2011].

De Luca, A., Oriolo, G. & Giordano, P. R. 2007. On-line estimation of feature depth for image-based visual servoing schemes. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2823–2828. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4209517

Franklin, G. F., Powell, D. J. & Emami-Naeini, A. 2001. *Feedback Control of Dynamic Systems*, 4th edition, Prentice Hall PTR.

Gommel, M., Haitz, M. & Zappe, J. 2004. Autoportrait project: portrait drawings with a robotic arm. http://www.robotlab.de/auto/portrait.htm [Accessed 20-May-2011].

Hill, J. & Park, W. 1979. Real time control of a robot with a mobile camera. In *Proceedings of 9th ISIR*, Washington, D.C., 409–417.

Hodges, S. E. & Hall, T. 1996. Looking for a cheaper robot: visual feedback for automated PCB manufacture. http://www.ifm.eng.cam.ac.uk/automation/publications/papers/seh-thesis.pdf

Hutchinson, S., Hager, G. & Corke, P. 1996. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation* **12**(5), 651–670.

Instruments, N. 2011. Control laws. http://www.ni.com/white-paper/8156/en/ [Accessed 04-April-2012].

Kennedy, D. & Osuga, R. 2012. Calligraphy robot uses a motion copy system to reproduce detailed brushwork. http://www.diginfo.tv/v/12-0181-r-en.php [Accessed 31-July-2013].

Kragic, D. & Christensen, H. 2002. *Survey on Visual Servoing for Manipulation*, Technical report, Centre for Autonomous Systems, Numerical Analysis and Computer Science.

Kudoh, S., Ogawara, K., Ruchanurucks, M. & Ikeuchi, K. 2009. Painting robot with multi-fingered hands and stereo vision. *Robotics and Autonomous Systems* **57**, 279–288.

Lau, M. C. & Baltes, J. 2010. The real-time embedded system for a humanoid: Betty. In *Proceedings of the 13th FIRA Robot World Congress*, Communications in Computer and Information Science **103**, 122–129, Springer-Verlag.

Lin, C. Y., Chuang, L. W. & Mac, T. T. 2009. Human portrait generation system for robot arm drawing. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Singapore. IEEE, 1757–1762.

Liu, Y., Mejias, L. & Li, Z. 2012. Fast power line detection and localization using steerable filter for active uav guidance. *ISPRS12* **XXXIX–B3**, 491–496.

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press

Lu, Y., Lam, J. H. M. & Yam, Y. 2009. Preliminary study on vision-based pen-and-ink drawing by a robotic manipulator. In *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Singapore. IEEE, 578–583.

Nomura, H. & Naito, T. 2000. Integrated visual servoing system to grasp industrial parts moving on conveyer by controlling 6DOF arm. In *2000 IEEE International Conference on Systems, Man, and Cybernetics* **3**, 1768–1775.

Olsson, T., Bengtsson, J., Johansson, R. & Malm, H. 2002. Force control and visual servoing using planar surface identification. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, Washington, DC. IEEE, 4211–4216.

Ruchanurucks, M., Kudoh, S., Ogawara, K., Shiratori, T. & Ikeuchi,K. 2007. Humanoid robot painter: Visual perception and high-level planning. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Roma, Italy. IEEE, 3028–3033.

Sasaki, K., Noda, K. & Ogata, T. 2016. Visual motor integration of robots drawing behavior using recurrent neural network. *Robotics and Autonomous Systems* **86**, 184–195. http://www.sciencedirect.com/science/article/pii/S0921889016305383

Singh, A. K., Baranwal, N. & Nandi, G. C. 2017. Development of a self reliant humanoid robot for sketch drawing. *Multimedia Tools and Applications* **76**(18), 18847–18870.

Siradjuddin, I., Behera, L., McGinnity, T. & Coleman, S. 2010. Image based visual servoing of a 7 dof robot manipulator using a distributed fuzzy proportional controller. In *2010 IEEE International Conference on Fuzzy Systems (FUZZ)*, 1–8.

Srikaew, A., Cambron, M. E., Northrup, S., Peters, R. A., II, Ii, R. A. P., Wilkes, D. M. & Kawamura, K. 1998. Humanoid drawing robot. In *Proceedings of the IASTED International Conference on Robotics and Manufacturing*.

Team, O. D. 2013a. Geometric image transformations. http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html# [Accessed 18-July-2013].

Team, O. D. 2013b. Structural analysis and shape descriptors. http://docs.opencv.org/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html [Accessed 20-June-2013].

Tresset, P. & Leymarie, F. F. 2005. Generative portrait sketching. In *Proceedings of the 11th International Conference on Virtual Sytems and MultiMedia (VSMM'05)*, VSMM, Ghent, Belgium, 739–748.

Tresset, P. & Leymarie, F. F. 2006. Aikon: The artistic/automatic ikonograph. In *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH'06. ACM. http://doi.acm.org/10.1145/1179622.1179664.

Tresset, P., Leymarie, F. F. & Khaorapapong, N. 2011. Skediomata: guinea pig and performer. In *Proceedings of the 17th International Symposium on Electronic Art*, Istanbul, Turkey. ISEA Press.

Tsunashima, N. & Katsura, S. 2010. Reproduction of human motion using motion-copying system based on coordinate modification. In *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, 1609–1614.

Vahrenkamp, N., Boge, C., Welke, K., Asfour, T., Walter, J. & Dillmann, R. 2009. Visual servoing for dual arm motions on a humanoid robot. In *9th IEEE-RAS International Conference on Humanoid Robots*, 2009. *Humanoids 2009*, 208–214.

Vahrenkamp, N., Wieland, S., Azad, P., Gonzalez, D., Asfour, T. & Dillmann, R. 2008. Visual servoing for humanoid grasping and manipulation tasks. In *8th IEEE-RAS International Conference on Humanoid Robots*, 2008. *Humanoids 2008*, 406–412.

Weiss, L. & Sanderson, A. 1987. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation* **3**(5), 404–417.

Weiss, L., Sanderson, A. & Neuman, C. 1985. Dynamic visual servo control of robots: An adaptive image-based approach. In *Proceedings of IEEE International Conference on Robotics and Automation*, **2**, 662–668.

Wilson, W., Williams Hulls, C. & Bell, G. 1996. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation* **12**(5), 684–696.

https://doi.org/10.1017/S0269888919000146 Published online by Cambridge University Press