




Article

# Robotic Sponge and Watercolor Painting Based on Image-Processing and Contour-Filling Algorithms

Lorenzo Scalera <sup>1,\*</sup> , Giona Canever <sup>2</sup>, Stefano Seriani <sup>2</sup> , Alessandro Gasparetto <sup>1</sup>  and Paolo Gallina <sup>2</sup>

<sup>1</sup> Polytechnic Department of Engineering and Architecture, University of Udine, 33100 Udine, Italy; alessandro.gasparetto@uniud.it

<sup>2</sup> Department of Engineering and Architecture, University of Trieste, 34127 Trieste, Italy; gionacanever96@gmail.com (G.C.); sseriani@units.it (S.S.); pgallina@units.it (P.G.)

\* Correspondence: lorenzo.scalera@uniud.it

**Abstract:** In this paper, the implementation of a robotic painting system using a sponge and the watercolor painting technique is presented. A collection of tools for calibration and sponge support operations was designed and built. A contour-filling algorithm was developed, which defines the sponge positions and orientations in order to color the contour of a generic image. Finally, the proposed robotic system was employed to realize a painting combining etching and watercolor techniques. To the best of our knowledge, this is the first example of robotic painting that uses the watercolor technique and a sponge as the painting media.

**Keywords:** robotic art; watercolor; sponge painting; etching; image processing



**Citation:** Scalera, L.; Canever, G.; Seriani, S.; Gasparetto, A.; Gallina, P. Robotic Sponge and Watercolor Painting Based on Image-Processing and Contour-Filling Algorithms. *Actuators* **2022**, *11*, 62. <https://doi.org/10.3390/act11020062>

Academic Editors: Marco Carricato and Edoardo Idà

Received: 29 December 2021

Accepted: 16 February 2022

Published: 19 February 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

At the present time, there is a growing interest in creating artworks using machines and robotic systems, and the introduction of machine creativity and intelligence in art fascinates both engineers and artists [1,2]. Robotic art is a niche sector of robotics, which includes different types of performance such as theater, music, and painting. Robotic art painting is a sector in which artists coexist with experts in robotics, computer vision, and image processing. Robotic art projects can be seen as a new art sector in which art and technology mix together and the exact representation of the world is not the main focus.

In recent years, technology has entered our lives more and more, and the use of robots in art projects can be seen as the natural progression of this continuous increase of the presence of technology in our daily lives. Robotic art should no longer be seen simply as an exercise to see how robots can be moved freely or a test of robots' abilities, but also as a novel art sector with its syntax and artistic characteristics. Currently, several drawing machines and robotic systems capable of producing artworks can be found in the present literature. The majority of these systems adopt image-processing and non-photorealistic-rendering techniques to introduce elements of creativity into the artistic process [3,4]. Furthermore, most robotic machines capable of creating artworks are focused on drawing and brush painting as artistic media. *eDavid* is a robotic system based on non-photorealistic rendering techniques, which produces impressive artworks with dynamically generated strokes [5]. *Paul the robot* is an installation capable of producing sketches of people guided by visual feedback with impressive results [6]. Further examples of artistically skilled machines are given by the compliant robot able to draw dynamic graffiti strokes shown in [7] and by the robot capable of creating full-color images with artistic paints in a human-like manner described in [8]. In [9], non-photorealistic rendering algorithms were applied for the realization of artworks with the watercolor technique using a robotic brush painting system. In [10], the authors presented a Cartesian robot for painting artworks by interactive segmentation of the areas to be painted and the interactive design of the orientation of the brush strokes.

Furthermore, in [11], the authors adopted a machine learning approach for realizing brushstrokes as human artists, whereas in [12], a drawing robot, which can automatically transfer a facial picture to a vivid portrait and then draw it on paper within two minutes on average, was presented. More recently, in [13], a humanoid painting robot was shown, which draws facial features extracted robustly using deep learning techniques. In [14], a fast robotic pencil drawing system based on image evolution by means of a genetic algorithm was illustrated. Finally, in [15], a system based on a collaborative robot that draws stylized human face sketches interactively in front of human users by using generative-adversarial-network-style transfer was presented.

There are also examples of painting and drawing on 3D surfaces, such as in [16], where an impedance-controlled robot was used for drawing on arbitrary surfaces, and in [17], where a visual pre-scanning of the drawing surface using an RGB-D camera was proposed to improve the performance of the system. Moreover, in [18], a 3D drawing application using closed-loop planning and online picked pens was presented.

In addition to brush painting and drawing, other artistic media have been investigated in the context of robotic art. Interesting examples are the spray painting systems based on a serial manipulator [19] or on an aerial mobile robot [20], the robotic system based on a team of mobile robots equipped for painting [21], and the artistic robot capable of creating artworks using the palette-knife technique proposed in [22]. Other applications of robotic systems to art include light painting, performed with a robotic arm [23] or with an aerial robot [24], as well as the stylized water-based clay sculpting realized with a robot with six degrees of freedom in [25].

Elements of creativity in the creation process can be also acquired by means of the interaction between the painting machine and human artist. Examples are given by the human-machine co-creation of artistic paintings presented in [26] and by the Skywork-daVinci, a painting support system based on a human-in-the-loop mechanism [27]. Furthermore, in [28], a flexible force-vision-based interface was employed to draw with a robotic arm in telemanipulation, whereas the authors in [29] presented a system for drawing portraits with a remote manipulator via the 5G network. Further examples of human-machine interfaces for robotic drawing are given by the use of the eye-tracking technology, which is applied to allow a user paint using the eye gaze only, as in [30,31].

In this paper, we present a robotic sponge and watercolor painting system based on image-processing and contour-filling algorithms. To the best of our knowledge, this is the first example of robotic painting that uses the watercolor technique and a sponge as the painting media. In this work, we focused on filling continuous, irregular areas of a processed digital image with uniform color through the use of a sponge. More in detail, an input image was pre-processed before the execution of a custom-developed contour-filling algorithm, with the main focus being to find the best position of the sponge imprint inside the contour of a figure in order to color it.

In summary, the main contributions of this paper are the following: (a) the implementation of a robotic sponge and watercolor painting system; (b) the development of an algorithm for the image processing and the contour filling of an image to be painted using a brush with a pre-defined shape; (c) the experimental validation of the proposed method with the realization of two artworks also using a robotic etching technique.

This work fits into the context of figurative art, which, in contrast to abstract art, refers to artworks (paintings and sculptures) that are derived from real object sources. Within this expressive set, there are situations in which, in order to represent the subject, the activity of reproducing the contours is separated from the activity of filling in the backgrounds with colors or grey-scale tints. For example, it is well known that, in the case of comic books, the artist who is in charge of making the images of the panels (penciller) is different from the artist who is later in charge of coloring the backgrounds (inker or colorist). Another example is given by the engravings by Henry Fletcher (1710–1750). The artist realized the contours of the subject by means of etching; in a second step, he colored the flower petals [32].

In this work, we present two artistic subjects as test cases: the “Palazzo della Civiltà Italiana” and “Wings”. The former is a benchmark example to test the performance of the contour-filling algorithm. The latter was realized with a combination of etching and the watercolor technique. The etching was an intaglio printmaking process in which lines or areas are incised using acid into a metal plate in order to hold the ink [33]. In this case, the proposed algorithm was adopted to color an image for which the contours were first obtained with the etching technique.

The paper is organized as follows: Section 2 describes the image-processing and contour-filling algorithm. In Section 3, the watercolor technique, the experimental setup, and the software implementation are illustrated, followed by a description of the calibration process. The experimental results are shown in Section 4. Finally, the conclusions of the paper are given in Section 5.

## 2. Theoretical Framework

As explained in the Introduction, the purpose of the research was to identify an appropriate strategy to evenly fill an area of the canvas using a sponge as the painting tool. The problem involved a complex solution since the layout of the sponge, unlike that of a brush, can have a non-circular shape; in fact, often, the sponge has a rectangular design. We call the algorithm that fulfills this task the *contour-filling algorithm*. The input of the algorithm was an RGB image. The contour-filling algorithm was composed of four steps:

- *Image preparation*: The image was analyzed, and a color reduction was performed. After that, the contours of the uniformly colored areas were carried out;
- *Area division*: This passage can be skipped according to the artist that uses the software tool; if selected, it performs a Voronoi partition of the area to be painted. This possibility was introduced for two reasons: (1) in order to break up the excessive regularity of a large background, which is usually not aesthetically pleasing; (2) to stop the painting process in case the partial results are not as expected;
- *Image erosion*: The obtained contours were eroded in such a way to prevent the sponge from painting beyond the area bounded by the edges;
- *Contour-filling algorithm*: This is the heart of the algorithm, where sponge positions and orientations (poses) are defined.

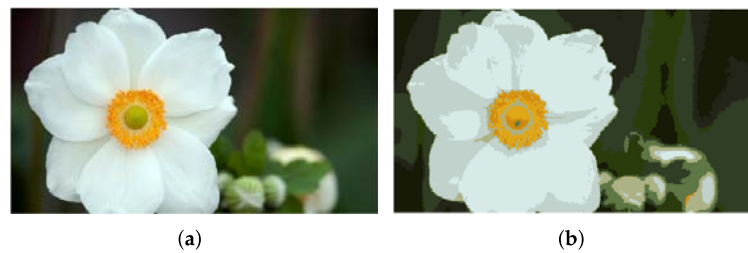
### 2.1. Image Preparation

In this first step, the image to be reproduced was selected and saved in a proper format: a matrix with dimensions  $height \times length \times 3$ . In this work, RGB images were considered, as the one of Figure 1a. The number of colors of the image was reduced according to the user-defined parameter  $n_{colors}$ , since the number of colors that the robot can manage was limited. Furthermore, the mixing of multiple colors to obtain particular shades has not yet been implemented. The resulting image was an RGB image, with pixel values  $px$  in the range  $0 \leq px \leq n_{colors}$ , as in Figure 1b.

During the whole process, only one color at a time was analyzed. The considered color image was converted to a black and white image (addressed as  $L$  in the following), and the obtained areas were labeled and converted into MATLAB `polyshape` objects for computational reasons. The area of possible internal holes was considered negligible with respect to the larger one, and for this reason, it was removed using a simple filter.

### 2.2. Area Division

The image can now be divided into smaller areas. This division was implemented to break the whole robotic task into several sub-processes in order to carry out eventual on-the-fly adjustments during the painting process. There are several ways to subdivide an area. We discarded the simplest ones, which consisted of dividing the area into regular subareas (such as squares), because graphic regularity, when perceived by the observer's eye, is seen as aesthetically poor.



**Figure 1.** Original image (a); image after reducing its number of colors (b).

The image was therefore divided by applying a Voronoi diagram to the considered shape to paint. In the present work, the Euclidean distance and a 2D space were considered. In usual terms, the Voronoi diagram resulted in a series of regions  $R_k$  in which every point inside was closer to  $P_k$ , the center of the considered region  $k$ , than to the generic  $P_j$ . The points  $P_j$  that define the Voronoi cells were random ones that fell inside the area to paint.

The area division was performed by superimposing the Voronoi cells lines on the original image. In this way, the original image area was divided into smaller subareas. Formally,  $V$  is the image matrix that represents the lines that define the Voronoi cells ( $V_{ij} = 0$  at the pixel belonging to the lines;  $V_{ij} = 1$  elsewhere).  $L$  is the image matrix to be divided ( $L_{ij} = 1$  at pixels of the area to paint;  $L_{ij} = 0$  elsewhere). The final image  $F$  is defined by the logical *and* operator:  $F = \text{and}(V, L)$ . An example of Voronoi area division is shown in Figure 2, where, for the sake of clarity, each separated area is represented with a different color.



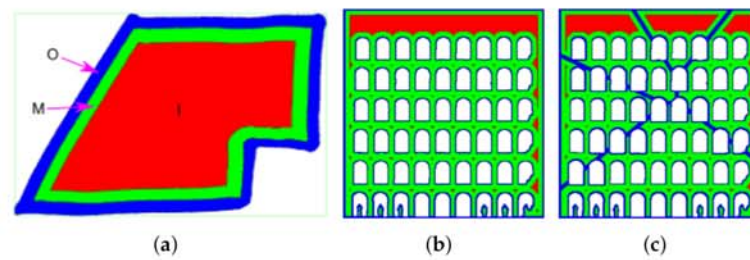
**Figure 2.** Voronoi area division of the bigger contour of the image of Figure 1b.

### 2.2.1. Image Erosion

As described below, the algorithm for filling an area aimed at generating a sequence of positions and orientations of the sponge (poses) in such a way that the envelope of all sponge imprints covered the entire area to be painted without exiting excessively from the area edges. To do this, some points were selected on the image where the center of the sponge would be sequentially positioned during the painting process. To facilitate the point selection, the area to be painted was divided into subsets of pixels through an erosion process applied to the initial area, as shown in Figure 3. An example of a typical binary erosion can be found in [34].

The original contours (referred to as  $C_j$ ) of the colored areas inside the image  $F$  were considered. The divided areas were eroded, and three regions were created:

- An *innermost area I* (red in Figure 3a);
- An *outermost area O* (blue in Figure 3a);
- A *median area M* between the innermost and the outermost ones (green in Figure 3a).



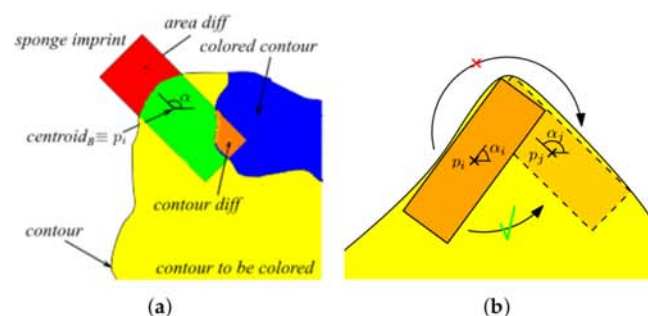
**Figure 3.** Example of image erosion. (a) Example of the eroded contour of a region and the definition of the obtained areas. (b) Eroded image obtained without dividing the area. (c) Eroded image obtained by dividing the area.

A lower point density was set in the innermost area than in the median area. Indeed, in the innermost area, all points were valid positions for the sponge, since its imprint always fell inside the contour  $C_j$ , whatever the sponge orientation was. On the other side, in the median area, a higher density of points was necessary to better follow the small details of the contour by properly defining the sponge orientation. Finally, in the outermost area, no points were placed because the sponge imprint in this area would surely fall out of the contour to paint. The acceptable distance of going beyond the contour was a user-defined percentage of the sponge area (e.g., 5%). This parameter also influenced the amount of erosion computed by the algorithm.

### 2.2.2. Contour-Filling Algorithm

The contour-filling algorithm was the core of the proposed strategy for robotic sponge and watercolor painting. In the following, it is established how the sponge positions were selected and how they were considered valid or not.

There are two possible approaches the user can choose to define the center sponge points  $p_i$ : random or based on a grid. For each generated point  $p_i$ , the algorithm evaluates if it falls inside the contour to paint. Let us define  $\alpha$  as the angular orientation of the sponge with respect to the horizontal axis (see Figure 4a). Inside the innermost area  $I$ , for every value of  $\alpha$ , the sponge imprint completely belongs to the area to be painted ( $\forall p_i \in I$  and  $\forall \alpha \Rightarrow B \subseteq I$ , where  $B$  is the sponge imprint); therefore, a random value  $\alpha$  was assigned to the points belonging to  $I$ .



**Figure 4.** Variables introduced in the algorithm (a); sponge rotation in the dragging technique (b).

No points were placed in the outermost area  $O$  since  $\forall p_i \in O$  and  $\forall \alpha \Rightarrow B \not\subseteq I$ . However, there could be a rare situation in which the sponge imprint corresponding to a particular point would fall within the area to be painted, for a particular value of  $\alpha$ . This could happen because image erosion is a coarse way to define the positions of the points. However, this strategy has the merit of speeding up the algorithm.

In the median region  $M$ , the sponge is rotated by an angle  $\alpha$  to define the best orientation that makes the sponge imprint fall as much as possible inside the contour to paint. This can be seen in Figure 4a, where the area to be minimized is the red one (*area diff*). To better follow the contour, a series of points were also placed on the line  $C$  that divides the outermost and the median regions. These points were chosen among the ones that define the contour vertices: if two consecutive vertices  $p_i$  and  $p_j$  are too far apart, a series of intermediate points  $p_k$  are automatically added.

To find the best angular rotation of the sponge (or the brush) in every defined point, the algorithm flowchart shown in Figure 5 is provided. The variables and parameters used in the flowchart are illustrated in Figure 4a. Their definitions are the following:

- *centroid<sub>B</sub>*: coordinates of the sponge centroid;
- *p<sub>i</sub>*: point in which the sponge is placed. There are 3 possibilities:
  - $p_i \in I$ ;
  - $p_i \in M$ ;
  - $p_i \in C$ ;
- $\alpha$ : angle of rotation of the sponge around its centroid;
- *contour to color*: contour yet to be colored;
- *ang incr*: angular increment used in the for cycle;
- *contour*: external contour to color;
- *area diff*: area of the sponge imprint that falls outside the contour to color; this is the variable to minimize;
- *% sponge out*: maximum user-defined percentage of the sponge size accepted to be outside of the contour to color;
- *contour diff*: area where the already painted contour and the rotated sponge overlap.

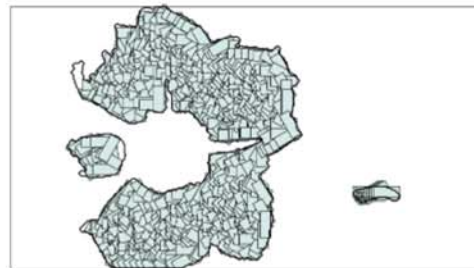
As can be seen in Figure 5, if  $p_i \in M \vee p_i \in C$ , the sponge is at first rotated clockwise and then counterclockwise. It was also evaluated if it fell sufficiently inside the image contour. If there was a valid position, the one that paints most of the yet-to-be-colored area was saved. In Figure 6, an example of a sequence of sponge positions resulting from the processing of Figure 2 with the proposed algorithm is provided.

Once the sequence of sponge positions is defined, two different painting technique can be selected:

- *Dabbing technique*: The sponge is moved between positions  $p_i$  on the canvas by raising the sponge in the passage between points. This technique does not require particular care since the sponge positions are already defined;
- *Dragging technique*: The sponge is moved between positions  $p_i$  on the canvas without being raised. This technique is more complex to simulate. Considering two defined points,  $p_i$  and  $p_j$ , if  $\|(p_j - p_i)\| > d_{max}$ , with  $d_{max}$  a user-defined maximum distance, a series of  $n$  intermediate points  $p_k$  that connect  $p_i$  and  $p_j$  are created. In these intermediate points  $p_k$ , the sponge imprint has to be verified; if it falls outside the area to paint, the sponge is raised. Furthermore, during the movement through the intermediate points  $p_k$ , the sponge is smoothly rotated between the rotation configuration of  $p_i$  and  $p_j$  (the sequence of consecutive poses is interpolated). If the starting angular position is  $\alpha_i$  and the ending one is  $\alpha_j$ , the intermediate points' angular positions are  $\alpha_k = (\alpha_j - \alpha_i)/n$ . It was also evaluated if the rotation was convenient to be clockwise or counterclockwise, with the goal of maximizing the coloring contribution to the area yet to be painted (Figure 4b). This process substantially corresponds to the addition of several poses to the original ones.



**Figure 5.** Flowchart of the algorithm for the definition of sponge angular rotation.



**Figure 6.** Sequence of sponge positions for the image of Figure 2.

### 3. Materials and Methods

In this section, we first briefly recall the watercolor painting technique. Then, the robotic system used in the experimental tests and the calibration operations needed for sponge painting are described.

#### 3.1. Watercolor Painting

In the watercolor painting technique, ground pigments are suspended in a solution of water, binder, and surfactant [35]. This technique exhibits textures and patterns that show the movements of the water on the canvas during the painting process. The main uses have been on detailed pen-and-ink or pencil illustrations. Great artists of the past such as J. M. W. Turner (1775–1851), John Constable (1776–1837), and David Cox (1783–1859) began to investigate this technique and the unpredictable interactions of colors with the canvas, which produce effects such as dry-brush, edge darkening, back runs, and glazing. In more recent years, authors, such as Curtis et al. [36], Lum et al. [37], and Van Laerhoven et al. [38], have investigated the properties of watercolor to create digital images by simulating the interactions between pigment and water.

Great importance in this technique is given to the specific paper and pigment used: the canvas should be made of linen or cotton rags pounded into small fibers, and the pigment should be formed by small separate particles ground in a milling process. In the tests performed in the present work, the Fabriano Rosaspina paper was adopted. This is

an engraving paper made of cotton fibers and characterized by a high density per square meter. Color pigments in tube format specifically made for watercolor were used as well.

This complex and unpredictable painting technique combined with the use of a sponge as the painting media is challenging. In fact, a trial and error process to fine-tune the system was necessary to find the perfect ratio between color and water and the best compression of the sponge during the releasing of color on the canvas.

### 3.2. Robotic Painting System

The painting system was built on a KUKA LBR iiwa 14 R820 robotic platform, shown in Figure 7a. This 7-axis robot arm offers a payload of 14 kg and a reach of 820 mm. The repeatability of the KUKA LBR iiwa 14 R820 robot is  $\pm 0.1$  mm [39]. The manipulator was equipped with a Media flange Touch electrical, a universal interface that enables the user to connect electrical components to the robot flange (Figure 7b). The flange position is described by its tool center point, which is user defined and can be personalized depending on the tool attached to the robot flange.

In order to equip the robot for sponge painting, three components were designed (Figure 8): a connector, a calibration tip, and a sponge support. These components were fabricated using an Ultimaker 2+ Extended 3D printer. The connector was fixed to the robot flange with four screws. The calibration tip and the sponge support can be easily interfaced with the connector thanks to a pair of magnets with a 4 mm diameter, which were inserted into the printed material. The sponge used for painting was a make-up sponge (20 × 4 cm) with a piece of fabric glued on its top (Figure 8e). The size of the sponge was chosen empirically. However, the algorithm works with any size and section of sponge, but the larger the sponge, the less details can be painted and the less time it takes for the robot to paint. The piece of fabric on the tool was used to improve the dosage of water: the sponge works as an absorber, and the piece of fabric reduces the quantity of water that is deposited on the canvas. The main problems are related to the great variability given by the sponge behavior, by the color that can be less or more diluted, but also by the canvas paper that, after some time, becomes soaked by the water contained in the pigment solution.

The KUKA LBR iiwa was programmed in the Java environment, and the Sunrise Workbench application by KUKA was used to develop the applications needed to control the robot and set the software parameters. The image-processing and contour-filling algorithms (described in Section 2) were developed in MATLAB. To facilitate the software interface, a user-friendly MATLAB app was developed, which allows the user to set the parameters for the non-photorealistic rendering techniques, as well as to calibrate the painting setup and send commands to the manipulator (Figure 9).

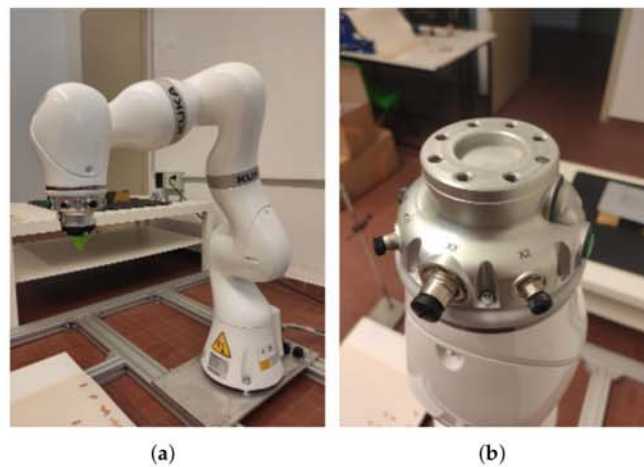
A client–server socket communication based on the TCP protocol was implemented to establish the communication between the MATLAB app and the Java program loaded on the robot, which interprets the MATLAB commands sent by the user. In this manner, the coordinates of the points resulting from the image processing can be fed to the robot controller and executed by the manipulator. The motion of the robot was planned using a trapezoidal speed profile for each couple of subsequent points. The painting operation was performed by limiting the joint speed of the manipulator to 30% of its maximum value, since abrupt movements of the sponge could damage the paper, especially during dragging.

### 3.3. Calibration

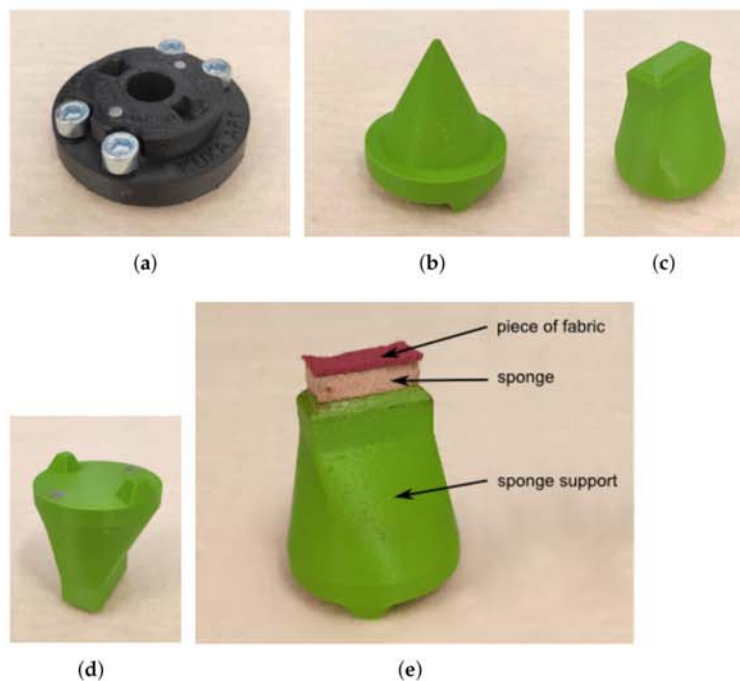
The robotic sponge painting required a calibration of the painting surface, as well as a calibration of the height of the sponge during color discharge. The canvas surface was calibrated using a procedure similar to the one implemented in [9,22]. More in detail, the robot equipped with the calibration tip was manually moved in contact with the canvas surface in five calibration points, as shown in Figure 10. The z position of the robot tool on each of these points was acquired using the MATLAB app. Actually, only three points could be sufficient, but a higher number was considered to improve accuracy. The interpolating plane was then generated starting from the five calibration coordinates and



used to define the corresponding  $z$  coordinate for each  $(x, y)$  planar position generated by the image-processing algorithm.



**Figure 7.** KUKA LBR iiwa 14 R820 manipulator (a); Media flange Touch electrical (b).



**Figure 8.** The 3D-printed parts for robotic sponge painting: connector (a); calibration tip (b); sponge support (c–e).

A calibration of the sponge height during color discharge was also needed to adjust the sponge compression during painting and, therefore, the quantity of color released. This operation was performed with a trial-and-error approach, since in this work, force control was not implemented on the robot. Figure 11 shows a scheme of the sponge discharge and some examples of sponge imprints for different heights. As can be seen from the figure, a compromise between a complete sponge imprint and a good amount of released water was needed. Indeed, if too much water were released on the canvas, the wet paper would produce ripples and undulations that would affect the quality of the results.

