



US 20250265717A1

(19) **United States**(12) **Patent Application Publication**  
**JANISCH et al.**(10) **Pub. No.: US 2025/0265717 A1**(43) **Pub. Date: Aug. 21, 2025**(54) **METHODS AND SYSTEMS FOR  
GENERATING SEGMENTATION MASKS**(71) Applicant: **Siemens Aktiengesellschaft**, 80333  
München (DE)(72) Inventors: **LUCAS JANISCH**, München (DE);  
**BENJAMIN SAMUEL LUTZ**,  
Grossenseebach (DE)(73) Assignee: **Siemens Aktiengesellschaft**, 80333  
München (DE)(21) Appl. No.: **18/705,286**(22) PCT Filed: **Oct. 21, 2022**(86) PCT No.: **PCT/EP2022/079332**

§ 371 (c)(1),

(2) Date: **Apr. 26, 2024**(30) **Foreign Application Priority Data**

Oct. 29, 2021 (EP) ..... 21205589.1

**Publication Classification**(51) **Int. Cl.****G06T 7/11** (2017.01)**G06T 7/00** (2017.01)**G06T 7/187** (2017.01)(52) **U.S. Cl.**CPC ..... **G06T 7/11** (2017.01); **G06T 7/0004**  
(2013.01); **G06T 7/187** (2017.01); **G06T**  
**2207/10024** (2013.01); **G06T 2207/20081**  
(2013.01); **G06T 2207/20084** (2013.01); **G06T**  
**2207/30164** (2013.01)

(57)

**ABSTRACT**

A method for generating a segmentation mask of at least one image comprising generating a plurality of superpixels for the at least one image, automatically generating labels for the generated plurality of the superpixels, wherein the automatically generating labels comprises generating the labels for training a semantic segmentation model based on a plurality of segmentation masks by generating a label for each superpixel by identifying the most similar reference superpixel from a reference data set of reference superpixels, wherein each reference superpixel is associated with a class for supervised training of the semantic segmentation model, wherein the labeled superpixels form the segmentation mask of the at least one image, and computer-aided checking of the generated segmentation mask for correctness, wherein the checking for correctness comprises labeling unlabeled superpixels and correcting labels of incorrectly labeled superpixels by assigning the incorrectly labeled superpixels to the correct class.

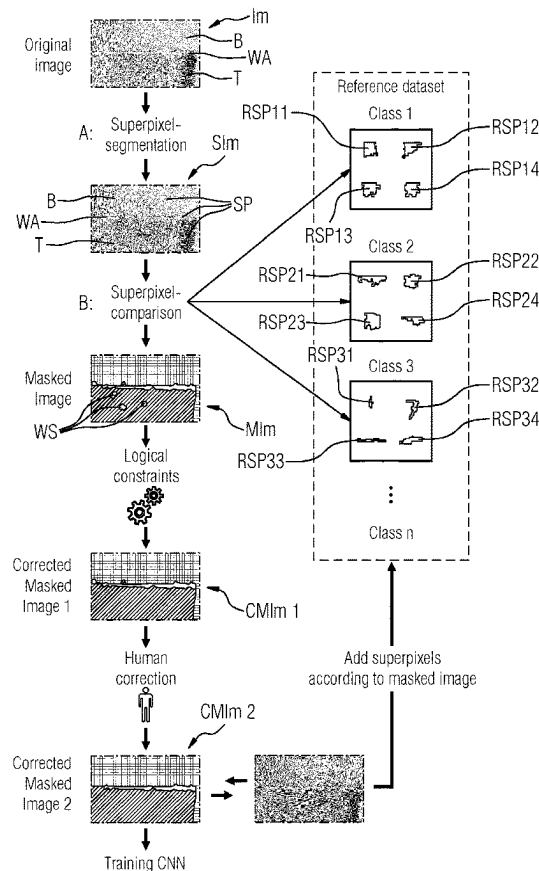
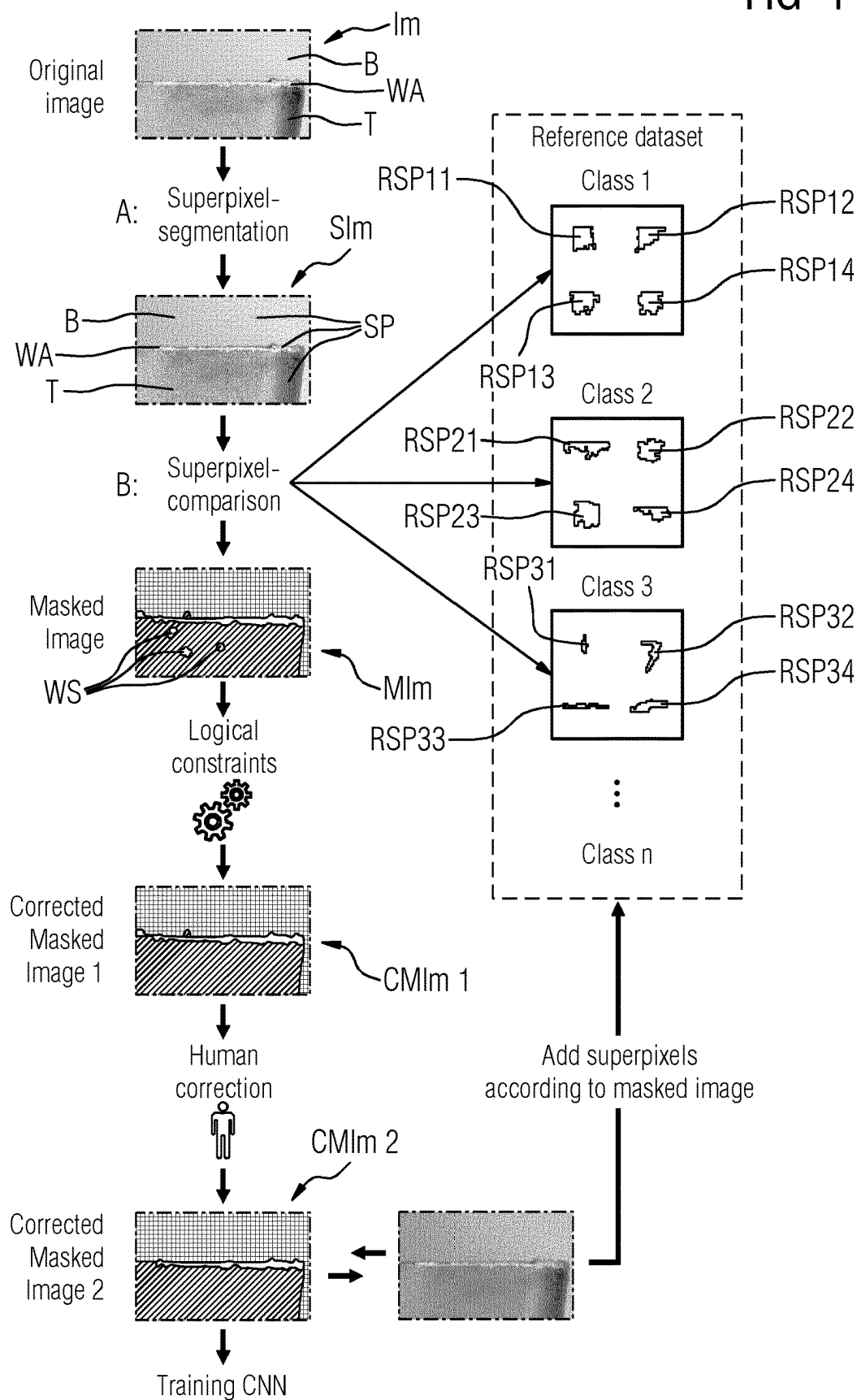


FIG 1



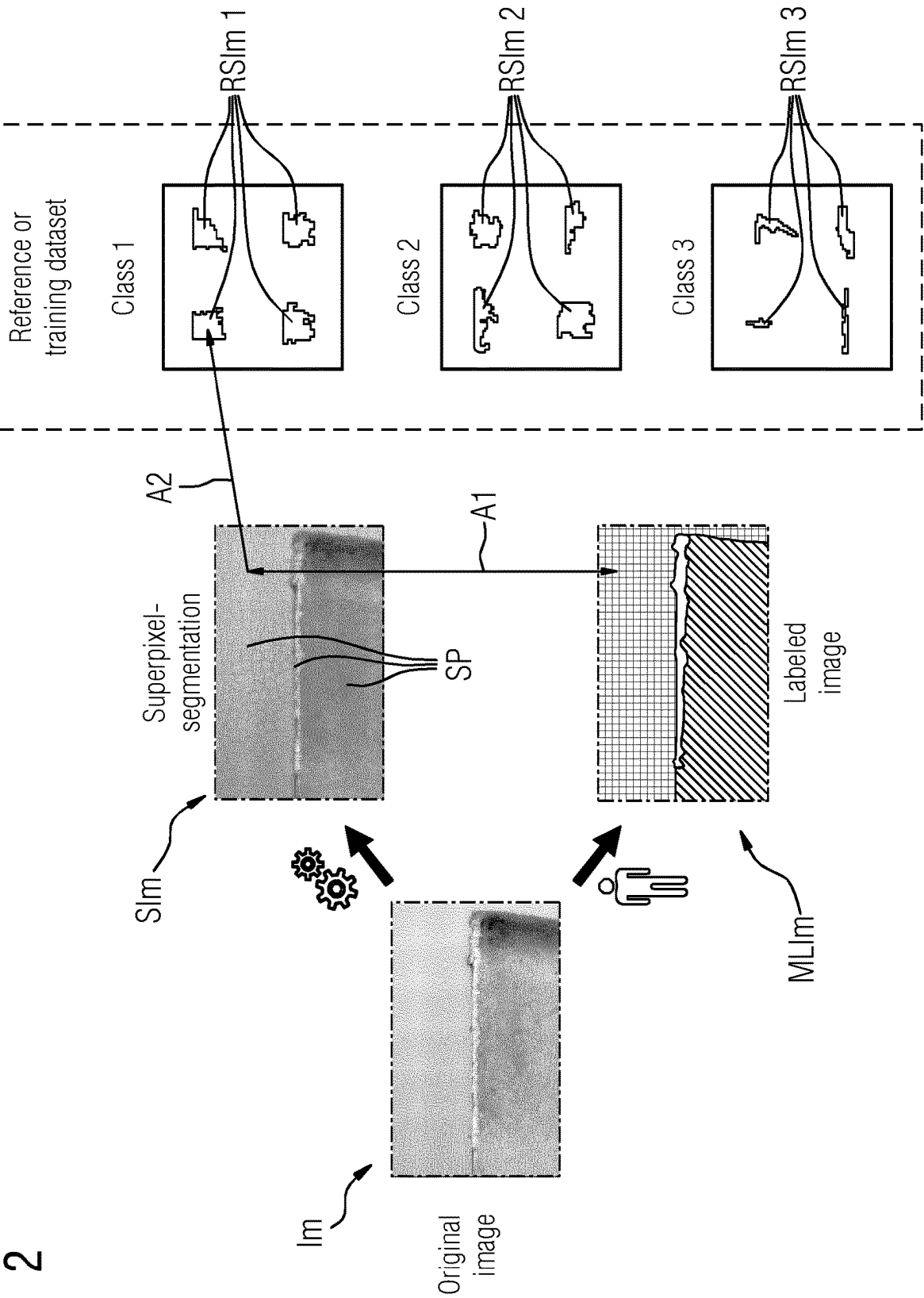


FIG 2

FIG 3

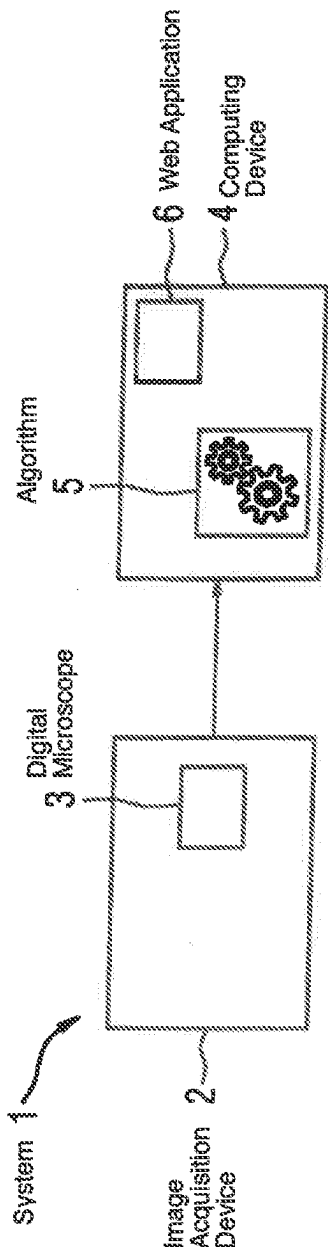


FIG 4

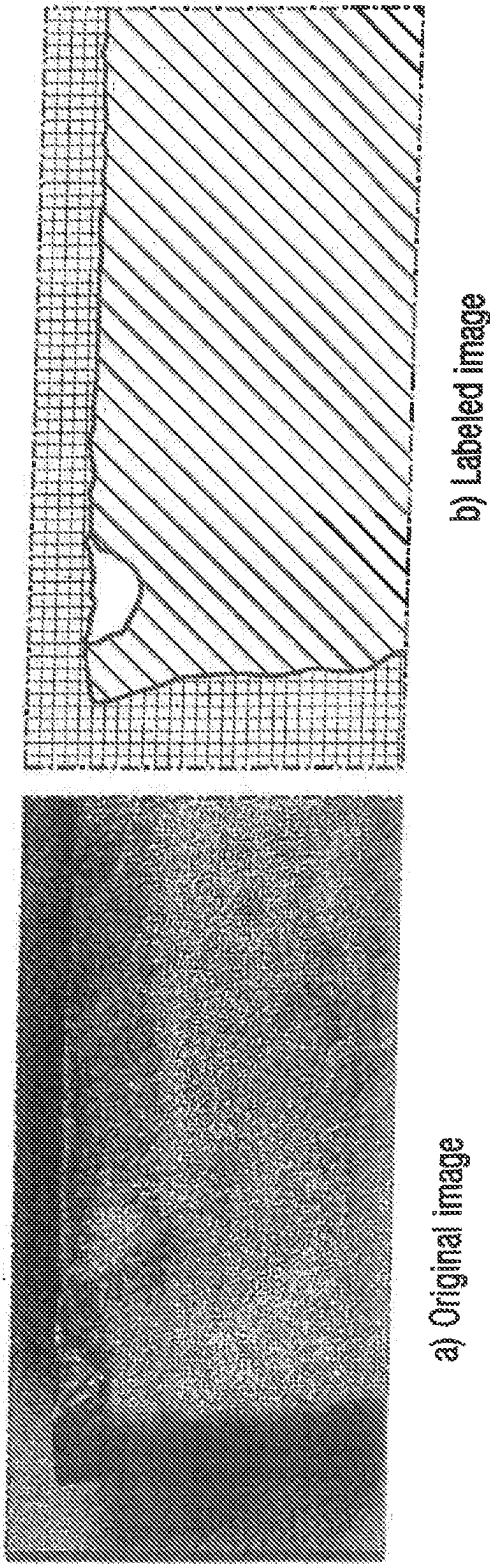


FIG 5

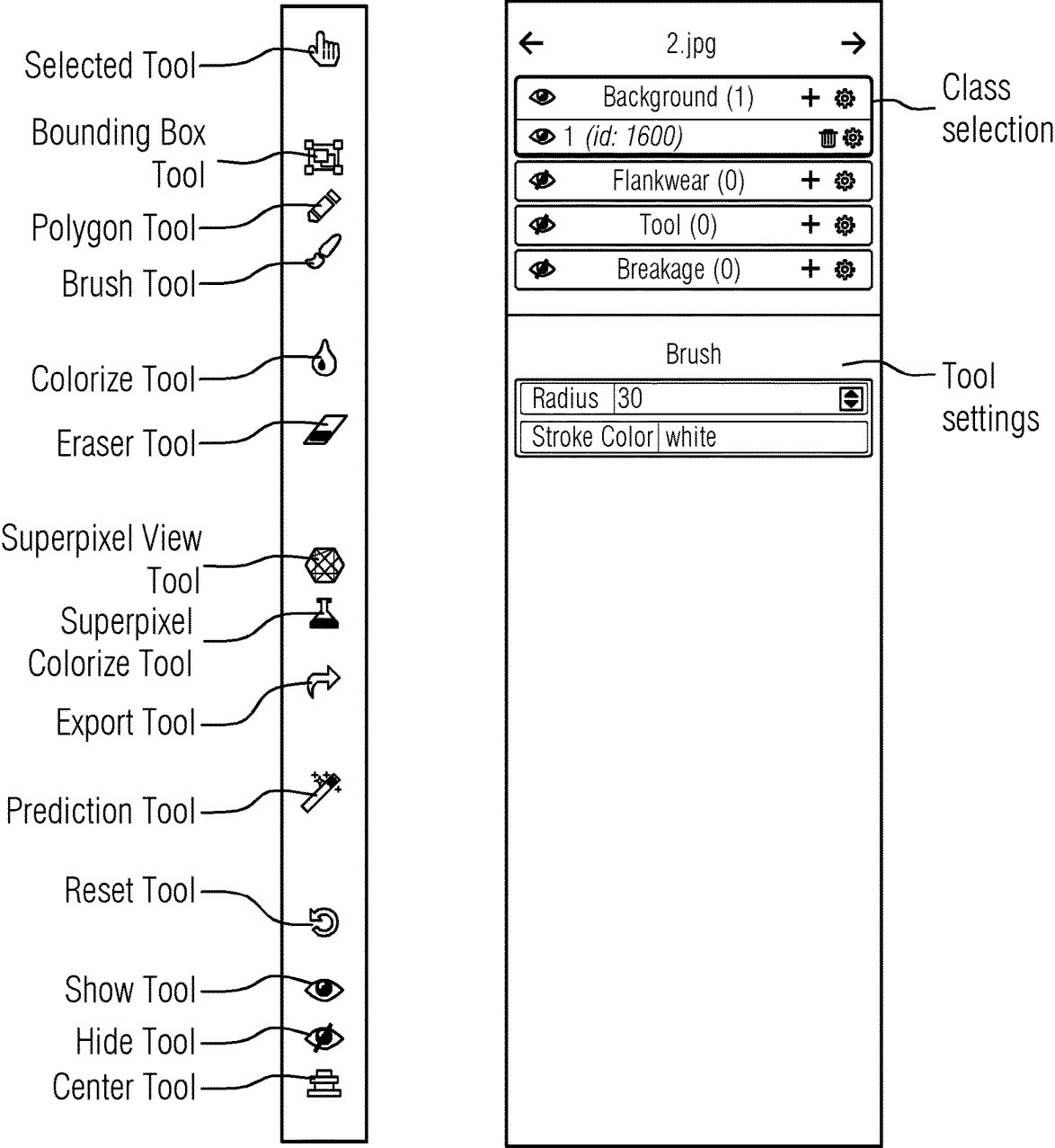
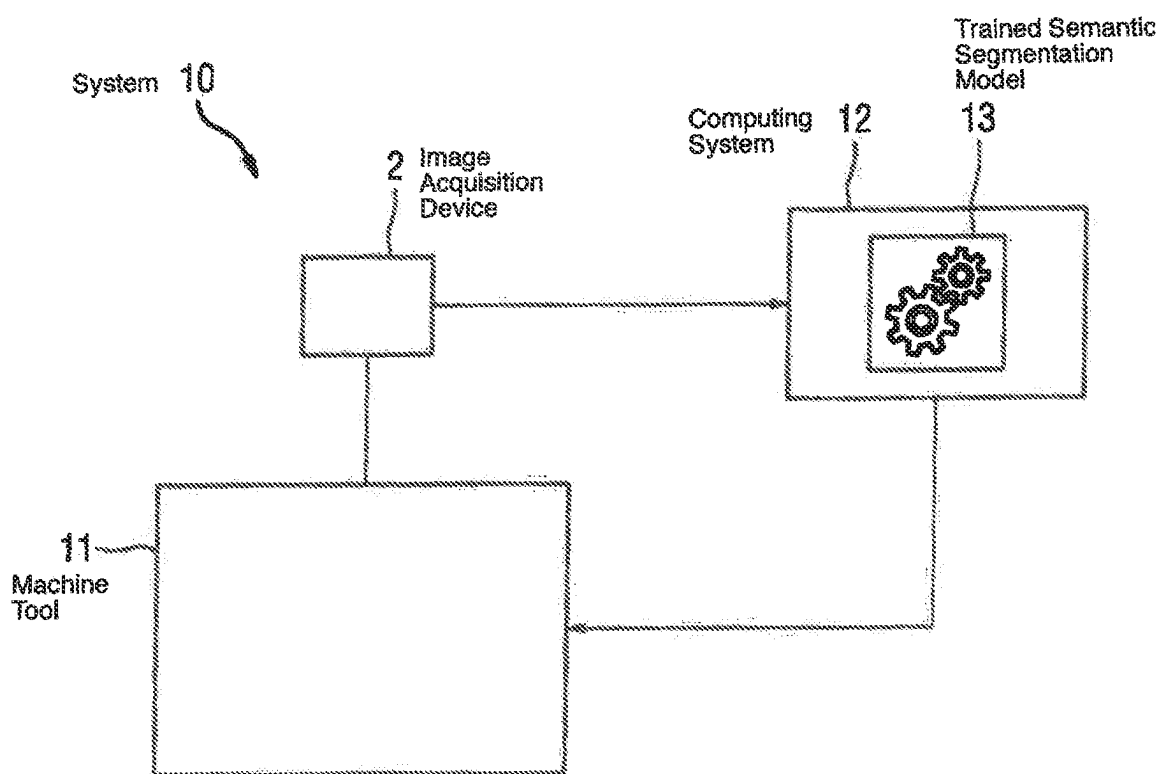


FIG 6



## METHODS AND SYSTEMS FOR GENERATING SEGMENTATION MASKS

[0001] The subject matter disclosed herein relates to systems and methods for generating segmentation masks based on provided images and training semantic segmentation models based on generated segmentation masks.

[0002] Moreover, the subject matter disclosed herein relates to a system for monitoring a condition of a tool, particularly of a cutting insert of a machining tool, by using the above-mentioned trained semantic segmentation model.

[0003] Labeling objects in image scenes and using semantic segmentation to perform this task is known in the prior art (see, e.g., US patent application, published as US 2015/0206315 A1, or arXiv:1711.05998v1 [cs.CV], online on Nov. 16, 2017). These techniques, being highly automatic, are prone to errors, where some superpixels are not labeled or labeled incorrectly.

[0004] The images can be (machining) tool wear images, images taken during an additive manufacturing process and comprising defects, coating images, etc. Although the focus in the present disclosure is made on machining application, it will be appreciated by the skilled person that the methods and systems disclosed herein are also applicable to images related to the additive manufacturing and/or coating processes.

[0005] In machining, the quality of the workpiece, such as dimensional accuracy and surface roughness, depends mainly on the condition of the cutting tool, i.e., its wear. The term tool wear refers to the change in shape from the original shape of the tool's cutting edges and neighboring areas.

[0006] There are many types of causes responsible for tool wear. Depending on the circumstances, the following main mechanisms have been identified:

[0007] Abrasion is mainly caused by hard inclusions in the material, such as carbides and oxides.

[0008] Cracks and fractures occur in the area of the cutting edge due to mechanical or thermal overload.

[0009] Plastic deformation occurs when the cutting tool has too low resistance to deformation but sufficient toughness.

[0010] Adhesion results from molecular adhesion between tool and workpiece.

[0011] Diffusion occurs at high cutting speeds and mutual solubility of cutting material. The cutting material is weakened by chemical reactions, dissolves, and is removed.

[0012] Oxidation also occurs only at high cutting speeds. Contact with atmospheric oxygen oxidizes the cutting tool, weakening the microstructure.

[0013] Crater wear can occur on the rake face of the tool, where chip flow moves with frictional forces under high load and temperature. Crater wear unfavorably changes the angle at the cutting wedge. This is usually avoided or minimized by selecting cutting conditions and a cutting tool that does not have a diffusion affinity with the workpiece material.

[0014] A firmly adhering deposit of material particles can occur on the rake face of the cutting edge during the machining process. Such a formation is called a built-up edge (SUE). It causes a rough and unclean workpiece surface. Built-up edges occur mainly when the cutting speed is too low, the cooling lubrication is insufficient, or the rake face of the tool is rough.

[0015] Another form of tool wear is notch wear, which occurs at the depth-of-cut line when the tool rubs against the shoulder of the workpiece.

[0016] Chipping occurs when the cutting line breaks away from the tool cutting edge instead of abrading. A sudden load during machining or thermal gradients are two main reasons for this type of wear.

[0017] In practice, flank wear is generally used as the main criterion for cutting tool wear. This type of wear is caused by friction between the flank of the cutting tool and the machined workpiece surface and leads to the loss of the cutting edge. The flank wear primarily influences the geometrical accuracy and surface quality of the workpiece and is therefore usually decisive for the usage time of the tool.

[0018] For the actual tool life detection, the ISO Standard 3685 can be used. It specifies recommended procedures, the typical tool wear profile and its measurement. For a standardized identification of the wear, the cutting edge is divided into three areas:

[0019] Area C is the curved part of the cutting edge at the tool corner, marking the wear area's outer end.

[0020] Area B is the remaining straight portion of the cutting edge, which consists of a uniform wear zone.

[0021] Area N extends over the area of mutual contact between the tool and the workpiece over about 1 to 2 mm along the main cutting edge. The wear in this area is of the notch type.

[0022] The cutting tool is assumed to have reached its tool life when the permissible tolerance for the workpiece can no longer be maintained due to the increasing wear width on the tool cutting edge. According to ISO 3685, the measurement of the width of flank wear is the most commonly used parameter to evaluate cutting tool lifespan. It is measured within zone B in the cutting edge plane perpendicular to the major cutting edge. As long as the wear profile is uniform, the average value of is measured. If the flank wear is instead uneven, the maximum wear width should be considered as a measurement.

[0023] To minimize machining costs, one must not only find the most suitable cutting tool and work material combination for a given machining operation but also reliably predict the tool life. The process of cutting tool wear typically consists of three characteristic parts: the initial period, the uniform wear period, and finally, the accelerated wear period. The machining process needs to be, on the one hand, stopped at the right time to prevent undesired consequences of the tool wear, while on the other hand, the tool should be replaced as late as possible to increase efficiency and decrease tooling costs. Therefore, monitoring the condition of the cutting tool plays an essential role in achieving consistent quality and controlling the overall cost of production.

[0024] In general, tool wear can be evaluated using either direct or indirect methods. The indirect methods are achieved by correlating suitable sensor data with the cutting tool wear. In this case, the wear information is not obtained directly but estimated from the signal of the measurement feature. Research on indirect TCM has utilized various signals such as cutting forces, acoustic emissions, vibrations, energy consumption, and temperatures to infer the current tool condition. In contrast, direct observation systems, such as laser scanners or camera-based solutions, measure exact geometric changes of the tool. For this purpose, a vision sensor can be built into the machine tool.

**[0025]** In most real-world scenarios, the tool stops at a fixed defined position, where an image of the tool is captured. Thereby, the image acquisition is neither influenced by chips and coolant nor by machine parameters such as vibration signals from the machining process. For sufficient illumination of the tool area, lights, for example LED lights, can be used. Also compressed air can be used before each shot to remove dirt and dust from the tools flank. A disadvantage of this approach is the production stop required to record the cutting tool. Nevertheless, image-based observation methods have the advantage of using image data similar to the standardized method defined in ISO 3685 described earlier. This, for example, allows the skilled person with knowledge of typical wear patterns to assess the condition of the tool manually.

**[0026]** In order to facilitate and to automate direct wear analysis in the machining industry computer vision (CV) techniques can be used.

**[0027]** The terms artificial intelligence (AI), Machine Learning (ML), Deep Learning (DL), and Computer Vision must first be distinguished from each other. Artificial intelligence refers to a field of computer science devoted to the development of systems that perform tasks that previously required human intelligence. Therefore, AI is an umbrella term for any computer program that attempts to mimic human Intelligence and thus includes Machine Learning, which in turn includes Deep Learning. Machine learning covers all approaches that allow systems to learn from data and improve without being explicitly programmed. In short, Machine Learning is a technology for realizing AI. Deep Learning is a subcategory of Machine Learning. Here, neural networks are used to analyze data in a manner similar to a human neural system. Computer vision is an interdisciplinary field for the development of systems that include all tasks that can also be performed by the human visual system, which can be described as perceiving and understanding what is seen by extracting complex information.

**[0028]** A Machine Learning process is generally a computer program that is designed to learn from experience with respect to a set of tasks and a performance measure. Specifically, knowledge is developed from insights through training with data in the form of complex mathematical models. These models can then be applied to new data of the same kind as the training data in order to make predictions or decisions. According to the diverse data and tasks, a multitude of different models exists. In this context, a model can be described as a mathematical formula with adjustable parameters. By training with a learning procedure, these parameters are modified to subsequently achieve better results.

**[0029]** Different learning procedures can be divided into

**[0030]** Supervised learning,

**[0031]** Unsupervised learning, and

**[0032]** Reinforcement learning.

**[0033]** Supervised learning uses a training dataset for which the correct output data is already known. The training data is created by manually assigning results to the input data. This process is called labeling and generates so-called ground truth data. During the supervised learning process, these generated data samples are iteratively fed to the algorithm. At each iteration, the model generates a prediction with its current parameters, which is compared with the label. If there is a discrepancy between the prediction and the ground truth label, an error rate is calculated. During the

learning process, the model tries to adjust its parameters to minimize this error rate for more accurate future predictions.

**[0034]** In unsupervised learning, unlike supervised learning, the algorithm is not told what the desired output is for each input. Instead, the algorithm attempts to categorize the data based on its various properties. Models based on this type of algorithm can be used to discover unknown data patterns themselves.

**[0035]** Reinforcement learning is a Machine Learning technique in which an agent optimizes subsequent actions and reduces errors by processing feedback from interaction with the environment. At each time step, the agent receives some observations from the environment and must choose an action, which is then fed back to the environment. Finally, the agent receives a reward or punishment from the environment. The agent then selects a subsequent action together with the information from a new observation. The agent's behavior is controlled by a function that derives actions from observations of the environment. While the goal in unsupervised learning is to find similarities and differences between data points, the goal in self-reinforcing learning is to find an appropriate action model that maximizes the agent's total reward.

**[0036]** Simple Machine Learning algorithms, such as Support Vector Machines or Random Forests, work on a variety of important problems. However, they fail to solve the central challenges of AI, such as speech or object recognition. Artificial neural networks (ANN) provide a remedy. Artificial Neural Networks have many artificial neurons distributed over layers. The number of layers used corresponds to the depth of the model and can reach several hundred layers. Deep Learning is the term used when talking about such complex, multi-layered networks.

**[0037]** There are three types of layers that are present in every model. These include

**[0038]** the input layer (receives the input data),

**[0039]** the hidden layer (extracts the patterns within the data), and

**[0040]** the output layer (produces the result of data processing).

**[0041]** In an artificial neural network, information is transferred from one layer to another via connection channels between neurons. When training the model, the weights, i.e., the numerical values at the connection channels, are adjusted. The output of one neuron can be used as input for all neurons in the subsequent layer. If this is the case, then the neurons of the model are fully meshed. The number of decisions an ANN makes in the input layer based on the input data is the number of neurons in the input layer. Each neuron in the second layer makes another decision by weighing the results of the first decision layer. In this way, a neuron in the second layer can make a decision at a more complex and abstract level than the one in the first layer. The output layer in an artificial neural network is the final layer that produces the algorithm's results. The design of the input and output layer is often trivial. For example, if an ANN should recognize in a handwritten image, whether it contains the number "5", the network can be designed to encode the intensities of the image pixels into the input neurons. For example, if the input image is a 64×64 pixel grayscale image, then there will be 4096 input neurons, which have values for intensities between zero and one. The output



layer, on the other hand, contains only a single neuron whose output value leads to the detection of a “5” in the image if it is above a threshold.

**[0042]** Backpropagation is the repetitive process of adjusting the weights of an ANN so that it is trained to get better at solving a problem. This is done by passing the input values through the network, comparing the output with the expected result, and calculating an error rate via a cost function. This error rate is cascaded backwards through the network to modify the weights so that the error rate is reduced, and the next time the same or similar inputs are passed through the network, the prediction is closer to the ground truth. The weights are usually not updated after each iteration but after passing through a batch of data. The results of the training are saved and made reusable by storing the trained weights.

**[0043]** For the cost function mentioned above, various approaches exist for calculating and minimizing the error between the output of the network and the labeled data. An established method for the calculation is the quadratic cost function with a subsequent search for the minimum of the function by the gradient descent method. The training process is successful if the result  $C$  (cost function)=0 is true, which means that the network’s outputs correspond to the labels of the training data. The goal of the gradient descent procedure is to find this minimum of the cost function as a function of all weights and bias values.

**[0044]** The supervised learning procedure, i.e., weight adjustment, of Machine Learning models requires labeled training data to evaluate the outputs generated by the network. The data is usually divided into

**[0045]** training,

**[0046]** validation, and

**[0047]** test data.

**[0048]** The training dataset is used to train the weights of the network. The more training data a learning algorithm receives, the better it can adapt its model and reduce the error rate. The performance of a model also depends on the quality of the training data. If the algorithm is shown too many wrong examples, it will learn wrong answers.

**[0049]** During training, the model also predicts an output for each input from the validation dataset. This prediction is executed based on what it learns from the training dataset. The calculated cost function results from the validation dataset do not affect the adjustment of the weights in the model. One of the main reasons for using a validation dataset is to ensure that the model is neither over-fitting nor under-fitting to the training data. An over-fitted ANN is able to evaluate the data from the training dataset but is not capable of making accurate predictions for new data with which it has not been trained. Under-fitting, on the other hand, refers to models that do not describe data or reality accurately enough. Based on the validation results, the hyper-parameters of the models, such as the learning rate or the training duration, can be selected.

**[0050]** The test dataset is used to perform an unbiased evaluation of the final ANN.

**[0051]** The percentage distribution of data between the different domains varies greatly with the nature of the task. An established division of the data consists of 60% training data, 20% validation data and 20% test data. For ANNs with few hyperparameters to adjust, the validation dataset can often be kept very small to use the data for a larger training dataset.

**[0052]** Convolutional neural networks (CNNs) are a special type of multilayer neural network for processing data that has a known grid-like topology. Input examples can include image data that can be represented as a 2D grid of pixels. Convolutional meshes are therefore frequently used in computer vision. The main distinction between CNNs and ANNs is that CNNs are composed of convolutional layers instead of conventional artificial neurons.

**[0053]** Conventional neural networks comprise fully meshed or partially meshed neurons in several layers. These structures reach their limits when processing images, as there must be a large number of inputs corresponding to the number of pixels. The number of layers and the connections between them are usually extremely large and can therefore only be handled by computers with very high performance. Convolutional neural networks provide a solution to this.

**[0054]** The majority of developed CNNs comprise a combination of three different layers. These are

**[0055]** the convolutional layer,

**[0056]** the pooling layer, and

**[0057]** the fully connected layer.

**[0058]** To increase the non-linearity of the output, which is unlikely to have a linear relationship with the input, the convolutional layer can comprise an activation function (as its component). One example of an activation function is the Rectified Linear Unit (ReLU). ReLU is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

**[0059]** The pooling layer can condense and reduce the resolution of the recognized features. This aggregation can remove redundant information and reduce the amount of data, which in turn can reduce the required computing power and the sensitivity of subsequent convolutional layers to the location of the perceived features.

**[0060]** As with convolutional layers, pooling operators can consist of a fixed-size window that is pushed overall regions in the input and calculates an output step by step for the input data under consideration. Unlike the convolution layer, however, the pooling layer does not contain parameters. Instead, pooling operators are deterministic and usually compute either the maximum or the mean of the elements in the aggregation window. These operations are known as maximum pooling and average pooling.

**[0061]** The end of the architecture of a CNN is formed by one or more fully meshed (fully connected) layers. They join the repetitive sequence of convolutional and pooling layers and map the results to the classes or objects to be recognized.

**[0062]** Metric Learning is a popular concept in modern Machine Learning areas to learn a suitable similarity measure between two entities. The approach aims to automatically construct task-specific distance metrics from supervised data in a Machine Learning manner. Convolutional neural networks provide a new representation of input data and enable automatic feature ex-traction with the goal of achieving a higher level of abstraction in data transformation. In recent years, Deep Learning and Metric Learning have been merged to introduce the concept of Deep Metric Learning (DML). Often, a neural network learns to predict multiple classes. This poses a problem when new classes are to be removed or added from the data. In this case, the neural network must be updated and retrained with the entire dataset. Deep metric learning models, on the other hand, learn a similarity function so that they can determine

whether the two images are the same or very similar. This allows us to classify new classes of data without having to retrain the network.

**[0063]** One of the further advantages of DML models is their robustness to class imbalances. With only a few instances per class, it is sufficient for metric learning approaches to recognize the similarity of new pictures to these reference images. Another possible advantage is that the trained models can also be used with different object classes. For instance, a metric learning model that can recognize similarities in images of giraffes will most likely also recognize similarities in images of other animals correctly.

**[0064]** The disadvantages of Deep Metric Learning models are that they need more training time than normal networks because they require pairs of images to learn. Also, a metric learning model does not output the probability of a class but the distance to each instance in a reference dataset.

**[0065]** The primary approach of classical Metric Learning is to learn a metric to reduce the distances between samples of the same class and increase the distances between the samples of a different class. In most conventional metric learning methods, samples are represented by hand-crafted features. The goal is to learn a feature mapping to project samples from the original feature space to a new space with more meaningful discrimination using a similarity relationship between samples.

**[0066]** With this approach, a good metric can be obtained from the input data, but the handgenerated features are task-independent and can lead to information loss, which limits the learning performance of the metric. Given the success of CNNs in classification tasks, researchers are now extracting embeddings directly from these networks. The Deep Metric Learning models developed in this way far outperform classical metric learning methods. Moreover, DML models are more transferable to new tasks because fine-tuning the parameters for manual feature extraction is no longer required.

**[0067]** For the training of a DML model, a set of data points  $X$  and their corresponding labels  $Y$  are used. The goal is to train a CNN, here also referred to as feature extractor  $G_W$  where  $W$  are the Learned Weights. Together with a Distance metric  $D$  which is usually fixed beforehand  $D(G_W(x_1), G_W(x_2))$  produces small values if the labels  $y_1, y_2 \in Y$  of the samples  $x_1, x_2 \in X$  are equal, and larger values if they are not.

**[0068]** The most important factors when building a Deep Metric Learning model are the network structure, the loss function, and the sample selection. The Siamese Neural Network (SNN) can be used as a leading network structure. The SNN contains two identical subnetworks. "Identical" here means that they (subnetworks) have the same configuration with the same parameters and weights. The parameter update is mirrored over both subnetworks. As a metric learning approach, the Siamese network receives pair images, including positive or negative samples, to train a network model. The Distance  $D_W$  for a pair of input samples  $x_1$  and  $x_2$  can be calculated as:

$$D_W(x_1, x_2) = \|G_W(x_1) - G_W(x_2)\|_2$$

where  $G_W(x_1)$  and  $G_W(x_2)$  are new representations generated by the neural networks.  $D_W$  is used to calculate the distance between the two inputs in loss functions.  $L_{Contrastive}$  that can be used to calculate a loss in Siamese network model is,

$$L_{Contrastive} = ((1 - Y)(D_W)^2 + Y(\max(0, m - D_W)^2)/2$$

where  $Y$  is the label value. If a pair of inputs is from the same class, the value of  $Y$  is one, otherwise its value is zero. A margin  $m$  is useful to prevent the network from "cheating" by mapping all  $X$  to the same point, making the distances between all samples equal to zero.

**[0069]** Another network structure is the Triplet network which is inspired by the Siamese network. It contains three objects, namely a positive, a negative, and an anchor sample. Triplet loss first focuses on the similarity between the pair samples of the same and different classes using shared weights. The classification is performed by comparing the similarity of pair samples. Compared to Siamese networks, Triplet networks are more distinctive by using both intra-class and cross-class relationships.

$$L_{Triplet} = \max(0, \|G_W(X) - G_W(X^p)\|_2 - \|G_W(X) - G_W(X^n)\|_2 + \alpha$$

**[0070]** Although Deep Metric Learning is particularly concerned with the metric loss function, informative sample selection also plays an essential role in Deep Metric Learning models. Informative samples are one of the most important elements that increase the success of Deep Metric Learning. The sampling strategy can increase both the success of the network and the training speed of the network. The triplet network, for example, uses an anchor, a positive sample, and a negative sample to train a network.

**[0071]** Some simple triplets have no effect on updating a model because of their low discriminatory power. These triplets cause a waste of time and resources. To solve this problem, informative sample triples instead of random samples can train more precise models faster. Therefore, Negative Samples Mining is introduced, where on each training step, triplets  $X, x_p, x_n$  are sampled that satisfying

$$D_W(X, x_n) < D_W(X, x_p) + \alpha,$$

i.e., the samples that the network fails to discriminate or is not able to discriminate with high confidence.

**[0072]** Computer Vision (CV) is a technical field that deals with enabling computers to see, identify, and process images in the same way as human visual perception. Machine Learning methods and specifically CNNs find their application in computer vision and are responsible for great successes in the field. The main CV tasks that are addressed in this disclosure are image classification and image segmentation.

**[0073]** Image classification is a task that attempts to capture an image in its entirety. The goal is to assign a label from a fixed set of categories to an input image. Typically, image classification refers to images in which only one

object occurs and is analyzed. This is one of the core problems of computer vision, which has a variety of practical applications despite its simplicity. Some examples of image classification include:

[0074] Classifying handwritten digits

[0075] Labeling an X-ray image as cancer or not

[0076] Assigning a name to a photograph of a face

[0077] Moreover, other seemingly independent machine vision tasks, such as image segmentation, can be reduced to an image classification job.

[0078] Semantic image segmentation, also referred to as pixel-level classification, is the process of grouping parts of the image that belong to the same object class. Therefore, labels are assigned from a label set to each pixel. The pixels that belong to the same class take the same color value, resulting in a masked image. This clustering aims to identify meaningful segments that can be used to describe the content of the image. In semantic segmentation, multiple objects of the same class are treated as one entity.

[0079] In contrast, so-called instance segmentation treats multiple objects of the same class as distinct individual instances.

[0080] One approach for semantic image segmentation models is an encoder-decoder structure, where the encoder part is a convolutional network as in classification networks. Instead of aggregating the data into a single class after the convolution part like in a classification task, the decoder section is attached, which is a reverse convolutional neural network. It upsamples the feature representation size layer by layer to a full resolution segmentation map. Across the encoder-decoder layers, connections can be used to transfer features that help to reconstruct the mask images with higher accuracy.

[0081] Superpixel segmentation, sometimes also referred to as oversegmentation, is a method of clustering pixels in an image into meaningful regions. The superpixels can then be used in place of the original pixels to reduce complexity and improve performance in many computer image processing applications such as object localization, multi-class segmentation and related computer vision tasks. Superpixels can

[0082] define a unique subset of the image,

[0083] represent connected sets of pixels,

[0084] preserve boundaries of objects in the image,

[0085] have a compact shape, are regularly distributed and have smooth edges when there are no object boundaries,

[0086] be created efficiently and

[0087] be controllable in their amount.

[0088] Therefore, a segmentation of an image can be done with two or more different values for the amount of areas to be generated.

[0089] Several superpixel generation algorithms exist. The Simple Linear Iterative Clustering (SLIC) algorithm is a simple and computationally efficient algorithm for superpixel segmentation. SLIC is also able to create superpixels with good boundary matches.

[0090] SLIC generates superpixels by clustering pixels based on their color similarity and proximity in the image plane. This can be done in a five-dimensional space  $[l_i, a_i, b_i, x_i, y_i]$ , where  $[l_i, a_i, b_i]$  is the pixel color vector in CIELAB color space and  $[x_i, y_i]$  is the pixel position. The CIELAB color space can be divided into three channels, where  $L^*$  represents the brightness,  $a^*$  the red-green axis and  $b^*$  the yellow-blue axis.

[0091] When the algorithm is used by default, the only parameter of the algorithm is  $k$  defining the desired number of superpixels. The clustering process begins with the initialization of  $k$  initial cluster centers  $C_i = [l_i, a_i, b_i, x_i, y_i]^T$ , which are sampled on a regular grid with a spacing of  $S$  pixels. To produce equal-sized superpixels, the spacing is defined

$$as = \sqrt{\frac{N}{k}},$$

where  $N$  is the number of pixels in the image. The centers can be moved to seed positions corresponding to the lowest gradient position, for example in a  $3 \times 3$  pixel neighborhood to avoid placing a superpixel on an edge and to reduce the probability of seeding a superpixel with a noisy pixel. Since the approximate area of a superpixel is  $S \times S$ , it can be assumed that the pixels belonging to this cluster center lie within an area of  $2S \times 2S$  around the superpixel center in the  $xy$ -plane. Each image pixel is assigned to the nearest cluster center in CIELAB space whose search area overlaps with that pixel. After all, pixels are assigned to the nearest cluster center, a new center is calculated as the mean  $[l, a, b, x, y]^T$  vector of all pixels belonging to the cluster. The process of assigning pixels and recalculating the cluster center is repeated until convergence. If the process is canceled before full convergence is achieved, the remaining disjoint pixels are assigned to nearby superpixels to force connectivity. A pseudo-code version of the described algorithm can be seen in following Algorithm 1.

---

```

1: Initialize cluster centers  $C_k = [l_k, a_k, b_k, x_k, y_k]^T$  by sampling pixels at regular grid
   steps  $S$ .
2: Move cluster centers to the lowest gradient position in a  $3 \times 3$  neighborhood.
3: Set label  $l(i) = -i$  for each pixel  $i$ .
4: Set distance  $d(i) = \infty$  for each pixel  $i$ .
5:   repeat
6:     for each cluster center  $C_k$  do
7:       for each pixel  $i$  in a  $2S \times 2S$  region around  $C_k$  do
8:         Compute the distance  $D$  between  $C_k$  and  $i$ .
9:         if  $D < d(i)$  then
10:          set  $d(i) = D$ 
11:          set  $l(i) = k$ 
12:   Compute new cluster centers.
13:   Compute residual error  $E$ .
14:   until  $E \leq \text{threshold}$ 

```

---

[0092] Superpixel Sampling Network (SSN) is another algorithm for generating superpixels, SSN combines a neural network, which for example can be based on a CNN, for extracting image features with a differentiable version of the superpixel algorithm SLIC. The differentiable SLIC is designed to compute pixel-superpixel association and to compute superpixel centers, wherein these two steps are performed as a closed loop. The method thus represents the first end-to-end trainable deep superpixel prediction technique.

[0093] Just as for the SLIC algorithm, the input is a representation of the image in the five-dimensional space  $[l, a, b, x, y]$ . In the primary part of the algorithm, the CNN extracts deep features  $F$  of size  $(n \times k)$  from the input image, and the superpixel centers are initialized. Where  $n$  is the number of pixels in the image and  $k$  is the number of channels the network determines. The  $m$  superpixel centers  $S^0 \in \mathbb{R}^{m \times 5}$  are then sampled evenly across the pixel grid. Just like described in the SLIC algorithm, the centers can be then moved to the position within a  $3 \times 3$  neighborhood with the lowest gradient. In the next step, the differentiable SLIC loop is repeated  $v$  times and the soft pixel-superpixel mappings  $Q_{pi}^t \in \mathbb{R}^{m \times 5}$  as well as the superpixel centers  $S$  are updated. The pixel-superpixel mappings in the original SLIC algorithm are hard associated, meaning that each pixel belongs to a particular superpixel in the SLIC loop. However, in the differentiable SLIC algorithm, these are turned into soft associations so that the algorithm can be trained from end to end. The soft associations can be specified by

$$Q_{pi}^t = e^{-\|F_p - S_i^{t-1}\|^2},$$

where the current SLIC iteration  $p$  refers to the pixel and  $i$  to the superpixel. The distance computations can be restricted to only few surrounding superpixels (for example nine surrounding superpixels). This is done due to the memory and computational power it takes to compute the distance for the complete set of superpixels. If memory and computational power is not an issue more surrounding superpixels can be considered.

[0094] The new superpixel centers are then derived using

$$S_i^t = \frac{1}{Z_i^t} \sum_{p=1}^n Q_{pi}^t F_p,$$

where  $Z_i^t = \sum_p Q_{pi}^t$  the normalization constant.

[0095] A detailed procedure of the superpixel segmentation by the SSN can be seen in the Pseudo Code—Algorithm 2.

[0096] Input: Image

$$I \\ n \times 5$$

[0097] Output: Pixel-Superpixel association  $Q_{n \times m}$   
 [0098] 1: Pixel features using a CNN,

$$F \\ n \times k = F(I)$$

[0099] Output: Pixel-Superpixel association  $Q_{n \times m}$   
 [0100] 1: Pixel features using a CNN,  
 [0101] 2: Initial superpixel centers with average features in regular grid cells,

$$S^0 \\ m \times k = J(F)$$

[0102] 3: for each iteration  $t$  in 1 to  $v$  do  
 [0103] 4: Compute association between each pixel  $p$  and the surrounding superpixel  $i$ ,

$$Q_{pi}^t = e^{-\|F_p - S_i^{t-1}\|^2},$$

[0104] 5: Compute new superpixel centers,

$$S_i^t = \frac{1}{Z_i^t} \sum_{p=1}^n Q_{pi}^t F_p; Z_i^t = \sum_p Q_{pi}^t.$$

[0105] Machine Learning models can be evaluated and compared. For example, Intersection over Union (IoU) can be used to evaluate object segmentation and object detection algorithms. In both tasks, IoU is applied to compare the detected area with the ground truth area of an object in the labeled image, i.e., an image from the ground truth data.

[0106] When evaluating Machine Learning models for object detection and classification the predictions are usually assigned to one of the four following categories:

[0107] False Positive (FP): An object is detected although none is present.

[0108] False Negative (FN): An existing object is not detected correctly.

[0109] True Positive (TP): An existing object is detected correctly.

[0110] True Negative (TN): An object that is not present is also not detected.

[0111] A precision value is used to measure how accurate the model's predictions are, that is, the percentage of all predictions that are correct and can be calculated as  $TP/(TP+FP)$ . A re-call value is the number of correct detections made over the total number of ground truth objects and can be calculated as  $TP/(TP+FN)$ .

[0112] From this a precision-recall curve can be created, where precision is mapped on the ordinate and recall on the abscissa. To plot this curve, all detections of a single class are used and listed in descending order with respect to the detection probability given by the model. Since the first element has the highest probability, there is a high probability that it is a TP. For this case, the precision computes to one, while the recall is zero. Recall scores increase as when moving down in the prediction ranking. The accuracy value, on the other hand, shows a zigzag pattern as it decreases with false positives and increases again with true positives. A good algorithm should have high precision over all values of recalls.

[0113] To be able to compare different CNNs with the described metrics, a measure called Average Precision (AP) can be used. AP calculates the area under the precision-recall-curve.

[0114] When comparing classification systems, it may be appropriate to use mean Average Precision (mAP). This is calculated by computing the AP for each class and then averaging them over the number of all classes  $N$ .

[0115] Supervised learning techniques build predictive models by learning from a large number of training examples that are accompanied by labels indicating the ground-truth model output. Although current techniques have achieved great success, it is worth noting that for many tasks, the high cost of the labeling process makes it difficult to obtain large amounts of data. A good example is the MS-COCO dataset, including images of objects found in everyday life in their natural environments. Here 2.5 million category instances had to be once annotated with image-level class labels as well as with in-instance segmentation labels. When looking at image-level class labels, it took on average 20 seconds per object. For the instance segmentation, outlines had to be drawn, which took on average 80 seconds per object. An example for dense pixel-level annotations is the Cityscapes Dataset for Semantic Urban Scene Understanding. In that work, the annotation of an image took around 1.5 hours on average. Therefore, it is desirable that segmentation models operate with a lower supervision effort.

[0116] One possible approach to address the problem within the context of the supervised ML is to generate and annotate a dataset offline. To differentiate this type of labeling process from other techniques, it is referred to as pre-labeling. This form of labeling is expensive, time-consuming, and should be performed by experts. In the absence of a sufficiently large number of specialists, crawling images on the Internet is a relatively inexpensive and promising alternative. An exemplary application for this is the so-called DALL-E model, where images and their textual descriptions are collected from the Internet. However, this can lead to other problems, such as inconsistent and incorrect captions by multiple annotators. In addition, this method is not suitable for collecting segmentation masks.

[0117] Another approach is called Weakly Supervised Labeling. Weakly supervised methods reconstruct predictions starting from a weaker supervision input. There is a correlation between the supervision source and the model's output target. In some cases of the supervised learning, the supervision source can be equal to the desired model output. For example, this can be training with pixel labels to also predict pixel label masks. A training process can also abstract from pixel labels to bounding boxes or from bounding boxes to image labels. This is called strong supervision.

[0118] The idea of weakly supervised learning is that the training starts with a low level of supervision, e.g., image labels and moves to a higher level such as bounding boxes. For semantic segmentation, various sources of weak supervision are explored, including image labels, point clicks, bounding boxes, scribbles, and various combinations of these. Weakly supervised annotation can also include incomplete supervision, where only a subset of the training data is labeled. Another related discipline is working with imprecise supervision, where the training data is only roughly labeled.

[0119] Furthermore, interactive labeling can be used. Interactive labeling methods reduce human effort by building on weakly supervised methods to generate annotations and then having a human correct them. In the interactive image segmentation procedure, a target object is first roughly labeled using the same methods as in weak supervision. A Machine Learning model then extracts the object as a binary mask. The interactive aspect allows for correction of this prediction using either the original weak annotation method or related approaches. An example would be a weak annotation using a bounding box and correcting the predicted mask using clicks.

[0120] Another alternative to pre-labeling is the use of active learning, which allows for the continuous refinement of a previously learned model as new labeled data becomes available. Active learning assumes that there is a human expert who can be consulted to obtain ground-truth labels for selected unlabeled instances. Since user interactions are time-consuming and thus expensive, active learning aims to minimize the number of user interactions required by querying only the information that best improves the accuracy of the given model. To find the most potentially useful unlabeled instances for learning, there are several selection strategies that strongly depend on the application and are still researched in terms of their optimization.

[0121] Moreover, semi-supervised learning can be applied. It attempts to automatically exploit unlabeled data in addition to labeled data to improve a model's learning performance, while no human intervention is assumed. In supervised learning, one is presented with a set of data points consisting of some input and a corresponding output value. The goal, then, is to create a model that can estimate the output value for previously unseen inputs. In unsupervised learning, on the other hand, no specific output value is provided. Instead, one tries to infer some underlying structure from the inputs. The combination of these tasks is called semi-supervised learning. The basic idea is that higher accuracy can be achieved by combining the available amount of labeled data with large amounts of unlabeled data instead of using only one or the other.

[0122] Several applications of computer vision techniques to tool wear analysis are known in the art, see e.g. Lutz et al at "Evaluation of Deep Learning for Semantic Image Segmentation in Tool Condition Monitoring", in 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), P. 2008-2013, IEEE, Boca Raton, FL, USA; Bergs et al at "Digital image processing with deep learning for automated cutting tool wear detection", in Procedia Manufacturing, 2020, Vol. 48, P. 947-958; Treiss et al "An Uncertainty-based Human-in-the-loop System for Industrial Tool Wear Analysis", in arXiv:2007.07129 [cs], 2020.

[0123] All these methods used only a limited number of annotated images (100 in Lutz et al., 50 in Bergs et al., 213 in Treiss et al.) Such limited datasets are common in the industry, as the resources and knowledge required to obtain and label data are often limited. Under these circumstances, CNNs can occasionally produce suboptimal results because they typically require a significant amount of data to be trained. Increasing the number of annotated images requires a considerable effort. For example, using these state-of-the-art approaches labeling a single image (HD resolution) takes between 10 and 20 minutes. Thus, labeling a training dataset of 500 images would require 15 days of annotating effort.

[0124] Therefore, to obtain better predictive models a method is required to generate ground truth masks of the images, e.g., tool images with high accuracy.

[0125] The present disclosure focuses on further reducing human effort at an earlier stage of the machine learning development in condition monitoring applications.

[0126] In order to achieve this objective, the present invention provides a method for generating a segmentation mask of at least one image, for example a (machining) tool wear image, an image taken during an additive manufacturing process and comprising defects, a coating image, etc. The method comprises

[0127] A) Generating a plurality of superpixels (SP) for the at least one image, in particular for the whole image,

[0128] B) Automatically generating labels for the generated plurality of the superpixels (SP), wherein said automatically generating labels comprises generating said labels for training a semantic segmentation model based on a plurality of segmentation masks by: generating a label for each superpixel by identifying the most similar reference superpixel from a reference data set of reference superpixels, wherein each reference superpixel is associated with a class for supervised training of said semantic segmentation model, wherein the labeled superpixels form the segmentation mask of the at least one image;

[0129] C) Computer-aided checking of the generated segmentation mask (MIm, CMIm1) for correctness, wherein said checking for correctness comprises labeling unlabeled superpixels and correcting labels of incorrectly labeled superpixels by assigning the incorrectly labeled superpixels to the correct class.

[0130] The image can be an application-specific image, i.e., image with a content that is specific for a particular application, e.g., an image of a worn out or damaged tool, etc.

[0131] The classes can be application-specific classes, i.e., classes that are specific for a particular application: for example tool wear degree classes and/or tool wear types, defect classes related to the additive manufacturing or to the coating inspection.

[0132] The use of superpixels may increase the boundary adherence of the generated segmentation masks.

[0133] The provided method falls into class of annotation or labeling methods.

[0134] In an embodiment, the generating a plurality of the superpixels can be done by using a Superpixel Sampling Network.

[0135] In an embodiment, the Superpixel Sampling Network can be based on a combination of a SLIC algorithm and a deep neural network.

[0136] In an embodiment, different labels are generated for different superpixels, particularly for each superpixel a label is generated.

[0137] In an embodiment, automatically generating labels involves a calculation of a similarity of the superpixels with reference superpixels from a reference dataset.

[0138] In an embodiment, the similarity of a superpixel of the plurality of the superpixels with each reference superpixel from the reference dataset is calculated and this calculation is repeated for each superpixel of the plurality of the superpixels.

[0139] In an embodiment, calculating the similarity comprises setting a similarity threshold.

[0140] In an embodiment, when the calculated similarity is below the set similarity threshold, a notification is issued. In an embodiment, in the notification the user can be asked to intervene and to classify the particular superpixel by hand. In this case, the method falls into the category of active learning methods.

[0141] In an embodiment, the calculation of the similarity involves applying a Distance Metric Learning method to the superpixels and to the reference superpixels.

[0142] Tool wear detection and defect detection, in general, are usually faced with the problems of an imbalanced dataset with limited defect samples. When some classes are scarcely present in standard classification models, they are most likely perceived as noise or outliers, resulting in more misclassifications of this minority than the prevalent class. To address this problem, Deep Metric Learning is applied in defect detection tasks. The Deep Metric Learning algorithm is less susceptible to a class imbalance because even if there is only one image representing a defect type in a reference dataset, every image to be labeled that is more similar to this defect image will be classified correctly.

[0143] In an embodiment, the Distance Metric Learning method comprises extracting features from images, for example from superpixel images and/or from the reference superpixel images, with a feature extraction module, e.g. nonlinear CNN-based feature extraction module, wherein the feature extraction module embeds the extracted image features into a feature space so that the more related the images are to each other the closer are the encoding vectors in the feature space, and defining the similarity of two images by a distance of their vectors in the feature space, e.g. by an appropriate distance metric.

[0144] In an embodiment, the training of the DML model may comprise using a loss function, e.g. Soft-triple-loss or Proxy-anchor-loss function.

[0145] In other words, Deep Metric Learning proposes to train a CNN-based, nonlinear feature extraction module (or an encoder), that embeds the extracted image features (also called embeddings) that are semantically similar, onto nearby locations while pushing dissimilar image features apart using an appropriate distance metric. The similarity metric is effective to distinguish small differences between images.

[0146] In an embodiment, the reference dataset comprises one or more subsets, wherein each subset comprises reference superpixels associated to one, e.g., to a single one, class of the classes and different subsets comprise reference superpixels associated with different classes. Such classes—in addition to a background class—for tool, e.g. cutting insert wear, can be flank wear class, build-up-edge class, groove class, etc.

[0147] In an embodiment, the reference dataset comprises one or more subsets, wherein each subset comprises reference superpixels associated to one tool wear degree class of the tool wear degree classes and different subsets comprise reference superpixels associated with different degrees and/or types of tool wear

[0148] In other words, a given input image is first passed through a deep network that generates features for each pixel. These features are then passed to the SLIC's modification, which performs iterative clustering, resulting in the desired superpixels. In order to obtain superpixels that

represent the border of even the smallest existing tool wear area, the number of superpixels to reach that requirement is determined once iteratively as a function of the image resolution.

[0149] In an embodiment, different colors are associated with different subsets.

[0150] In an embodiment, the superpixels are labeled with colors associated to subsets, they are most similar with.

[0151] In an embodiment, the method further comprises a check of the segmentation mask for logical constraints.

[0152] In an embodiment, the method further comprises a computer-aided checking of the segmentation mask for correctness. This segmentation mask check can be performed by a user, e.g., by machining tool expert. In an embodiment, this includes checking and correcting misdetected superpixels, for example this can be performed by a human expert, who checks if some of the superpixels were not labeled or were labeled incorrectly and corrects the labeling by associated the misdetected superpixels to the correct class.

[0153] In an embodiment, a human can be asked to check the resulting mask for faulty annotations and to correct them. Moreover, the human can be actively asked to check specific superpixels for which the similarity with a superpixel of the reference set was below a threshold, when such threshold was set while calculating the similarity.

[0154] In order to achieve the above-mentioned objective, the present invention also provides a system for generating a segmentation mask of at least one image, the system comprising an image providing device for providing the at least one image and a computing device associated with the image providing device, wherein the computing device is configured to receive the at least one image and to carry out the steps of the above-described method.

[0155] The image providing device may comprise an image acquiring device and/or a database or alike to facilitate providing of the at least one image.

[0156] It will be appreciated by a skilled person that the computing device comprises a memory that stores machine-executable components and a processor that is operatively coupled to the memory, and is configured to execute the machine-executable components.

[0157] In order to achieve the above-mentioned objective, the present invention also provides a system for monitoring a condition of a tool of a machining tool, wherein the system comprises an image acquisition device, which is associated with a machine tool and is designed to acquire images of the tool, and a computing system associated with the image acquisition device and comprising a semantic segmentation model trained on a plurality of segmentation masks produced according to the above described method to perform analysis of the images of the tool.

[0158] In order to achieve the above-mentioned objective, the present invention also provides a computer-readable medium comprising a computer program with instructions for carrying out the above described method.

[0159] The above and other objects and advantages of the invention will be apparent upon consideration of the following detailed description of certain aspects indicating only a few possible ways which can be practiced. The description is taken in conjunction with the accompanying drawings, in which like reference characters refer to like parts throughout, and in which:

[0160] FIG. 1 shows a flow diagram of a method for generating a segmentation mask of at least one machining tool wear image,

[0161] FIG. 2 illustrates a pipeline of an embodiment for producing the reference or training dataset,

[0162] FIG. 3 shows a system for producing segmentation masks,

[0163] FIG. 4 illustrates an image and a manually labeled image,

[0164] FIG. 5 illustrates an annotation tool, and

[0165] FIG. 6 shows a system for monitoring a condition of cutting inserts in a machining tool.

[0166] The same reference signs in different figures may well correspond to the same technical features.

[0167] FIG. 1 illustrates a processing pipeline that leads to a segmentation mask MIm, CMIm1 or CMIm2 for a machining tool wear image Im. This pipeline corresponds to an embodiment of the method according to the invention.

[0168] The processing pipeline starts with the generation of superpixels 3P for the whole image Im (Step A). The image Im may comprise an image of a tool T (or at least a part of the tool), in particular of a tool T with a tool wear area WA, and a background B. The use of superpixels SP can increase the boundary adherence of the final segmentation masks MIm, CMIm1, CMIm2.

[0169] To generate superpixels SP algorithms like SLIC, Quick Shift, or Watershed can be used, as they have a low computational cost. These algorithms perform differently if it comes to the boundary adherence. For example, if a superpixel SP covers both the background B and the tool area T, the image can neither be correctly labeled nor serve as training data for the Deep Metric Learning model. While the SLIC algorithm outperforms the Watershed and Quick Shift algorithm in representing the tool T, background B, and wear area WA boundaries, some superpixels SP still may cover more than one class.

[0170] The generation of the superpixels SP can be also performed by using a Superpixel Sampling Network (SSN), which can be designed as a combination of a e.g. modified SLIC algorithm and a deep neural network. This method enables end-to-end training and the use of a deep network and can therefore be optimized by task-specific images and their semantic masks.

[0171] In an embodiment, the input image Im can be first passed through the deep network that generates features for each pixel. These features are then passed to the SLIC's modification, which performs iterative clustering, resulting in the desired superpixels SP—see superpixelized image Sim in FIG. 1. To obtain superpixels SP that represent the border of even the smallest existing tool wear area WA, the number of superpixels SP to reach that requirement can be determined once iteratively as a function of the image resolution.

[0172] In an embodiment, the SSN can be pre-trained, e.g. on the BSDS500 dataset. When using this pre-trained model on tool images, superpixels that adhere to all semantic boundaries of different classes can be generated.

[0173] When using SSN (from scratch or pre-trained, e.g. PyTorch implementation), the segmentation results in a decrease of the superpixel boundary adherence.

[0174] In an embodiment, the number of superpixels N to get the best representation of the object boundaries was determined as  $N = h \times w / 10^4$  where h is the pixel height and w the pixel width of the image.

**[0175]** The next step (Step B) deals with automatically generating a label for every generated superpixel SP and thus a mask MIm of the image Im. The resulting segmentation mask or masked image MIm shows that the superpixels were labeled into three classes: background (chequered area), tool body or undamaged tool area (dashed area) and the wear area (unmarked (white) area) in MIm.

**[0176]** In an embodiment, the automatic labeling method takes a superpixel SP and calculates its similarity with every superpixel RSP11, RSP12, RSP13, RSP14, RSP21, RSP22, RSP23, RSP24, RSP31, RSP32, RSP33, RSP34, . . . from a reference dataset. This dataset can be organized into folders representing different classes Class 1, Class 2, Class 3, . . . Class n that might be present in the image Im. At least a part of the classes Class 1 (e.g. background class), Class 2 (e.g. tool body or unworn tool class), Class 3 (e.g. wear class), . . . , Class n (e.g. breakage) correspond to (tool) wear grade classes and/or wear types of the machining tool T—here Class 3 and Class n.

**[0177]** In an embodiment the similarity calculation is performed for every superpixel RSP11, RSP12, RSP13, RSP14, RSP21, RSP22, RSP23, RSP24, RSP31, RSP32, RSP33, RSP34 in the reference dataset. When the similarity calculation is done, the reference data set superpixel with the highest correlation is checked for the class it is labeled with. In an embodiment, different colors can be associated with different classes Class 1, Class 2, Class 3, . . . , Class n. In an embodiment, the superpixel SP in the image Im can be then labeled with the color of the class associated with this most similar reference dataset superpixel RSP11, RSP12, RSP13, RSP14, RSP21, RSP22, RSP23, RSP24, RSP31, RSP32, RSP33, RSP34. Using colors in the segmentation mask facilitates visualization, e.g. by means of a web application 6 (see below) and helps to distinguish different detected tool defects.

**[0178]** In an embodiment, a similarity threshold can be set, when the similarity is checked. If the similarity value of a given superpixel to the most resembling reference superpixel does not reach this threshold, it will not be labeled or colored. This may simplify the subsequent correction of wrongly labeled superpixels SP.

**[0179]** In an embodiment the procedure (similarity calculation) is repeated for every superpixel SP in the image Im. By this method, the segmentation task is turned into an image-level classification task.

**[0180]** In an embodiment, to calculate the similarity of the superpixels SP with the reference data superpixels RSP11, RSP12, RSP13, RSP14, RSP21, RSP22, RSP23, RSP24, RSP31, RSP32, RSP33, RSP34, Distance Metric Learning can be applied. Particularly, a Deep Metric Learning method can be used in which a feature . . . extraction module extracts features from the superpixel images, particularly from each superpixel's image. The feature extraction module can be based on a convolutional neural network, for example on a non-linear CNN. Such feature extraction modules are sometimes called encoder. The image features, also called embeddings, are then projected into an embedding space that maps related images onto nearby encoding vectors and unrelated ones far apart. The similarity of two (superpixel) images is then defined by the distance of their vectors in the embedding space. For example, superpixel images have shorter distances to superpixel images of the same class, while they have a larger distance to superpixel images of other classes.

**[0181]** In other words, applying the Deep Metric Learning may comprise training a CNN-based, nonlinear feature extraction module (or an encoder), that embeds the extracted image features (also called embeddings) that are semantically similar, onto nearby locations while pushing dissimilar image features apart using an appropriate distance metric. The similarity metric is effective to distinguish small differences between images.

**[0182]** To train a Deep Metric Learning model, superpixels of the tool images can be generated to serve as training data (see e.g. FIG. 2). For example, a series of images and masks of a plurality of different cutting tool flanks can be used. In an embodiment, no distinction is made between the two classes of flank wear and build-up edge (BUE), since they are very similar in appearance and are often unlikely to be distinguished by the metric learning model nor the human operator. In an embodiment, also other types of wear can be ignored in the training procedure, if they occur only rarely in the dataset and are difficult to distinguish from undamaged tool areas. This facilitates the convergence of the deep metric learning model.

**[0183]** In an embodiment, the training of the DML model may comprise using a loss function, e.g., Soft-triple-loss function or Proxy-anchor-loss function. The model converges at a much lower loss value when Soft-triple-loss is used. With Soft-triple-loss better training results can be achieved. To select the optimal weights the results on the validation dataset can be used.

**[0184]** Using the Deep Metric Learning algorithm increases insensitivity to class imbalance of the method, since the DML is not heavily influenced by the present class imbalance. When classes are scarcely present in standard classification models, they are most likely perceived as noise or outliers, resulting in more misclassifications of this minority than the prevalent class. DML helps with this. E.g. even if there is only one flank wear superpixel in the reference dataset, every flank wear superpixel in the image will be labeled correctly as long as it does not look more similar to any superpixel from another class. Another advantage of the deep metric learning approach is that it can compare not only the superpixels of the classes it is trained on but also of unseen ones. The same is true if the appearance of the classes varies, as the superpixels will be detected as long as the reference dataset covers these changes.

**[0185]** In an embodiment, a plurality of masked images MIm for the same original image Im can be produced while adding superpixels to the reference dataset after producing each of the masked images MIm. This can improve the quality of the reference dataset and therefore of the masked image MIm.

**[0186]** In an embodiment, when the segmentation mask MIm is generated, an algorithm can be used that checks the result MIm for logical constraints. These are set specifically for a given use case and are changed or disabled for other segmentation scenarios (other use cases). An example of a logical restriction is that there cannot be a superpixel of the wear class WS surrounded only by background superpixels since the wear can only occur on the tool or by superpixels from the bulk body of the tool T that has no wear (see the masked image MIm in FIG. 1). A single superpixel detected as wear in the background area B or in the unworn area of the tool T can be automatically relabeled with the background class or unworn tool class. As a result, a corrected masked image CMIm1 can be generated.



[0187] In an embodiment, the method integrates a check and correct of misdetected and/or unlabeled superpixels by a user. This can be useful, since there may be incorrect detections that logical constraints cannot resolve. When the operator is satisfied and has completed the labeling process, the final mask CMIm2 can be either used for the analysis of the wear itself or as training data for a fully automatic image segmentation model. Such subsequent correction of the segmentation mask may reduce the amount of (computational) time spent labeling an image.

[0188] In an embodiment, all superpixels of the marked/annotated image(s) MIm, CMIm1 or CMIm2 can be added to the reference dataset according to the class they were annotated with. The reference dataset thereby becomes more extensive with each annotated image, which then leads to a better segmentation mask prediction in the following image to be labeled.

[0189] The algorithm described with regard to FIG. 1 can be summarized as a pseudo-code.

---

Algorithm “Interactive image segmentation”

---

```

1: procedure Active Image Segmentation (Image, Reference Dataset, Threshold)
2:   for each superpixel in Reference Dataset do
3:     Generate Encoding, E ← MetricLearning
4:     Generate superpixels of Image, S ← superpixelSamplingNetwork
5:     for each superpixel in S do
6:       Generate Encoding, SupEn ← MetricLearning
7:       for each RefEn in E do
8:         Calculate Distance from RefEn and SupEn, d ←
SimilarityFunction
9:         if d ≤ threshold then
10:           threshold ← d
11:           Save class of RefEn, SPClasses ← RefEn-
Class
12:   Colorize superpixel in Image according to SPClasses → MaskedImage
13:   Automatic logical correction of MaskedImage
14:   Human correction of MaskedImage
15:   for each superpixel in S do
16:     save superpixel with its class to the Reference Dataset
17:   return MaskedImage

```

The generated reference or training data can be divided into approximately 90 % train and 10 % test data.

---

[0190] In order to increase the amount and variety of data for training without having to create more pictures, it is useful to generate modified copies of the already existing images. In this procedure the images can be randomly flipped, rotated and/or subjected to an affine transformation. Furthermore, the image values for brightness contrast and saturation can be randomly modified before the training cycle.

[0191] In an embodiment, each class in the reference dataset comprises ten superpixels or less. This amount of reference data can be easily provided by manually labeling an image or two.

[0192] FIG. 2 illustrates a pipeline of an embodiment for producing the reference or training dataset for the DML model that can be used in the processing pipeline of FIG. 1.

[0193] It will be appreciated by the skilled person that the super-pixels SP are generated (superpixelated image Sim is produced) with the same algorithm settings as in the later application of the labeling method, e.g. of the method of FIG. 1. To determine the class of the superpixels, the original image Im can be labeled manually producing a manually labeled image MLIm. Subsequently, the annotation class of the mask MLIm at the position of a superpixel SP in the

superpixelated image Sim is determined (this is illustrated by the arrow A1 in FIG. 2). The superpixel SP can then be saved in the class corresponding folder Class 1 with reference superpixel images RSim1, Class 2 with reference superpixel images RSim2, Class 3 with reference superpixel images RSim3, etc. This can be done for every superpixel SP in the image Sim (arrow A2 in FIG. 2). For saving the superpixels as separate images, their area can be extracted, and black pixels can be added to the border. The superpixel SP with the image label can then be used to train the DML model. In the same manner, as the training data can be generated, the superpixels SP can be stored in the reference dataset for later use of the method. As for the start of the labeling procedure, only a small amount of reference is needed.

[0194] In an embodiment, it suffices to have only a few (one or two) of manually labeled images for creating the reference dataset. In an embodiment, the reference dataset comprises classes with ten or less images per class.

[0195] FIG. 3 shows a system 1 for producing segmentation masks. The system may comprise an image acquisition device 2 for acquiring images of interest. For example, the image acquisition device 2 can acquire images of cutting tools used in a milling process, e.g. milling heads. The tools can have one or more, e.g. four cutting flanks, so that a plurality of tool flank images can be generated. To capture the images the image acquisition device 2 may comprise or be associated with a digital microscope 3. The microscope may have a definite resolution, to which the images can be cropped to, to remove unnecessary information from regions like the background.

[0196] In an embodiment, the pictures show the undamaged tool body and the background. Depending on the tool's condition, regions depicting different tool wear defects are present. The present defects can be for example grouped into flank wear, build-up-edge, and groove.

[0197] To generate labeled training data, some of the images (e.g. one or two images) of the milling heads tools can be annotated by an expert to provide an initial reference dataset. The initial dataset can be also provided otherwise, e.g. downloaded from a database. This can be performed by aid of a computing device 4, which is associated with the

image acquisition device 2. The resulting masks of the manual labeling process can be images completely colored according to present classes. FIG. 4 illustrates an image and a labeled image.

[0198] The computing device 4 may further comprise an algorithm 5, e. g. designed as a software that comprises instructions that, when carried out by the computing device 4, produce a masked image MIm or a corrected masked image CMIm1, for example according to the processing pipeline of FIG. 1.

[0199] In an embodiment the computer device 4 may comprise a storage medium to store the masked images MIm or the corrected masked images CMIm1, CMIm2.

[0200] Furthermore, the computing device 4 may comprise a service, e. g. in form of a web application 6 that can be used by a user for example via a GUI and is designed to allow validation of the annotation method and to facilitate human correction of the masked image MIm or of the corrected masked image CMIm1.

[0201] The annotation view of an embodiment of the web application 6 is illustrated in FIG. 5. The annotation view can be designed to be used with a keyboard and a mouse and may include several shortcuts and gestures to improve the annotation efficiency.

[0202] The first step of generating an annotation is to select a class. This can be done in the right sidebar, as shown enlarged in FIG. 5, by clicking the plus button after a class. The following annotations are assigned to this class and appear in the corresponding color. In the same sidebar, classes can be added or deleted and their color code can be changed.

[0203] A variety of functions can be available for generating image annotations, which can be selected and executed in the left navigation bar. Classic annotation techniques are available here.

[0204] The Select Tool is the default tool in the annotation view and allows to choose and to modify annotations directly on the canvas. The users can carry out any of the following operations to modify annotations:

[0205] click and drag points to move them to their desired location,

[0206] hold the shift key and click to remove points,

[0207] click on any point on the contour of the shape to add more points.

[0208] The Bounding Box Tool allows the user to set two points that define the size and position of a rectangle. Subsequently, the box properties can be adjusted with the selection tool by clicking, dragging, and releasing.

[0209] The Polygon Tool allows users to click-drag release around objects to annotate them, while the system automatically assigns points along the boundary. After the polygon is generated, the select tool can be used to individually move each vertex independently to fine-tune the final shape.

[0210] Another function for manual labeling is provided by the Brush Tool. Here the user can first select a diameter in pixels to then click in the image to create a round polygon.

[0211] Furthermore, it is possible to create polygons with the selected diameter. As with the polygon tool, the resulting area can be adjusted with the selection tool.

[0212] When the user is satisfied with the polygon created by one of the above functions, it can complete the labeling for example by clicking on the Colorize Button. This will ink the polygon area in the appropriate class color.

[0213] If a user later disagrees with a previously colored area, the Eraser Button can be applied. To do this, the desired area to be deleted must be marked with a polygon using the previously described functions. Clicking the Eraser Button restores the original image information in the defined area.

[0214] The Superpixel Tool can be used to generate clustered areas for the whole image, e. g. by applying the Superpixel Sampling Network described above. The predicted superpixels can be then visualized by marking the boundaries, e. g. with thin yellow lines (correspond to gray lines in the superpixelized images Sim of FIGS. 1 and 2). The user can switch between the superpixel image and the original image at any time. Thereby the annotations made can be edited in both views. The superpixel view can be used on the one hand to adjust manual annotations, and on the other hand, the superpixel areas themselves can be colored automatically as a whole. For this purpose, first, the Superpixel Colorize Tool can be used. Subsequently, the desired class is selected to select any number of superpixels by clicking on them within their boundaries. When the selection is complete, the coloring can be triggered by pressing the Check Button in the left navigation bar.

[0215] At the bottom of the tool navigation bar, several helper functions can be implemented. The reset button restores the output state of the image and thereby deletes all annotations. With the usage of the mouse wheel, the image can be changed in size. Hereby the mouse position may be important as it zooms in and out from that location. In case the user gets lost in the zooming navigation, the Image Centering Tool can be used to show the image in its original size and position. Furthermore, there is a function to hide and show the annotations made. This allows the user to compare whether the mask matches the original image.

[0216] When the annotation of the image is finished, the superpixels can be added to the reference set by left mouse clicking the button. Thereby every superpixel can be added to a specific folder according to its annotated class. The additional reference data will then be used in the next image of the dataset when applying the prediction function.

[0217] For the case that faulty labeled superpixels were exported to the reference dataset, another app view can be implemented called Metric Learning. In this view the reference data for every existing dataset can be displayed. When accessing a specific reference dataset, the user is presented with a visualization of the folders organizing the superpixels images. In every folder, the superpixels images for a specific image and a certain class are located and displayed by a thumbnail. Here the user can either select the whole folder or single images to be deleted.

[0218] The generated segmentation masks MIm, CMIm1, and/or CMIm2 can be used to train a model, e. g. a semantic segmentation model. The masks MIm, CMIm1, and/or CMIm2 can also be used as additional reference data for the similarity detection step to label the next image. With every iteration, the automatically generated mask will therefore get better while reducing the human effort.

[0219] In an embodiment the computing device 4 can comprise a storage medium to store the masked images MIm or the corrected masked images CMIm1, CMIm2. Alternatively or additionally the computing device 4 may comprise an interface, for example a cloud interface, to store the images in a cloud, e.g. in a specific reference data database.

[0220] The generated segmentation masks MIm, CMIm1 or CMIm2 can be used to train a semantic segmentation

model. The semantic segmentation model can be based on a convolutional neural network. In an embodiment, the computing device 4 can comprise an algorithm to train the semantic segmentation model based on the segmentation masks.

[0221] FIG. 6 shows a system 10 for monitoring a condition of cutting inserts in a machining tool 11. The system can comprise the image acquisition device 2, which is associated with the machine tool 11 and is designed to acquire images of the cutting inserts used in the machine tool 11 for machining.

[0222] After an image of a cutting insert is acquired, it is sent to a computing system 12, for example to an edge computing device or to a service at a cloud platform or alike, that is designed to perform the analysis of the image. For this the computing system 12 comprises a trained semantic segmentation model 13 that can be based on a CNN and has been previously trained on the segmentation masks MIm, CMIm1, CMIm2 provided by methods described above.

[0223] In an embodiment, the computing system 12 can be provided with an interface allowing a user to choose the image to be analyzed and visualizing the results of the analysis. One possible result of the analysis can be a recommendation to an operator of the machine tool, e. g. a warning message indicating that the cutting tool is broken, and the operation of the machining tool shall be stopped to avoid damage of the machine or that the condition of the tool is good, and the operation may be continued.

[0224] In an embodiment, the analysis results comprise wear-related key performance indicators that can be stored and/or visualized, e. g. to the machine tool operator, with aid of the computing system 12.

[0225] Methods and systems disclosed herein reduce the human effort required to label tool images while maintaining or increasing the accuracy of the labels created. Therefore, the amount of time spent labeling an image is reduced. Using such images to train semantic segmentation model that is used for monitoring condition of tools or processes provides more accurate results.

[0226] An embodiment, in which the human can be asked to intervene when the similarity values of individual superpixels are below a threshold, falls into the category of active learning methods.

[0227] Regarding the influence of the size of the reference dataset, it is remarkable that already with a small amount of only ten superpixels per class, a high prediction quality is achieved by the DML model. Increasing the size of the reference dataset leads to further improvements.

[0228] Moreover, it has been observed that even if there is only one wear superpixel in the reference dataset, almost all wear superpixels in the image will be labeled correctly as long as they do not look more similar to any superpixel from another class. This shows that the methods disclosed herein show great insensitivity to class imbalance.

[0229] The reference signs used in the figures and in the claims are used for demonstrative purposes and shall not be considered as a limitation of corresponding claim features.

[0230] The above-described embodiments of the present disclosure are presented for purposes of illustration and not of limitation. In particular, the embodiments described with regard to figures are only few examples of the embodiments described in the introductory part. Technical features that are described with regard to systems can be applied to augment methods disclosed herein and vice versa.

1.-17. (canceled)

18. A method for generating a segmentation mask of at least one image, the method comprising:

generating a plurality of superpixels for the at least one image;

automatically generating labels for the generated plurality of the superpixels by generating said labels for training a semantic segmentation model based on a plurality of segmentation masks by generating a label for each superpixel by identifying a most similar reference superpixel from a reference data set of reference superpixels, wherein each reference superpixel is associated with a class for supervised training of said semantic segmentation model, wherein the labeled superpixels form the segmentation mask of the at least one image; and

computer-aided checking of the segmentation mask for correctness, wherein said checking for correctness comprises labeling unlabeled superpixels and correcting labels of incorrectly labeled superpixels by assigning the incorrectly labeled superpixels to the correct class.

19. The method of claim 18, wherein the generating a plurality of the superpixels is done by using a Superpixel Sampling Network.

20. The method of claim 19, wherein the Superpixel Sampling Network is based on a combination of a Simple Linear Iterative Clustering algorithm and a deep neural network.

21. The method of claim 18, wherein the automatically generating labels comprises calculating a similarity of the superpixels with the reference superpixels from the reference data set.

22. The method of claim 21, wherein the similarity of a superpixel of the plurality of the superpixels with each reference superpixel from the reference data set is calculated and this calculation is repeated for each superpixel of the plurality of the superpixels.

23. The method of claim 21, wherein the calculating the similarity comprises setting a similarity threshold.

24. The method of claim 23, wherein, when the calculated similarity is below the set similarity threshold, a notification is issued.

25. The method of claim 21 wherein the calculating the similarity comprises applying a Distance Metric Learning method to the superpixels and to the reference superpixels.

26. The method of claim 25, wherein the Distance Metric Learning method comprises extracting features from superpixel images with a feature extraction module, wherein the feature extraction module embeds the extracted superpixel image features into a feature space so that the more related the superpixel images are to each other, the closer encoding vectors are in the feature space, and defining a similarity of two superpixel images by a distance of their vectors in the feature space.

27. The method of claim 21, wherein the reference data set comprises one or more subsets, wherein each subset comprises reference superpixels associated to one class of the classes and different subsets comprise reference superpixels associated with different classes.

28. The method of claim 18, further comprising checking the segmentation mask for logical constraints.

29. A method for training a semantic segmentation model, the method comprising:

generating at least one segmentation mask of at least one image according to all of the method according to claim **18**;

training a semantic segmentation model on the at least one segmentation mask; and  
providing a trained semantic segmentation model.

**30.** A method for monitoring a condition of a tool of a machining tool comprising:

training a semantic segmentation model according to the method of claim **29**; and  
using the trained semantic segmentation model to monitor the condition of the tool.

**31.** A system for generating a segmentation mask of at least one image, the system comprising:

an imaging device for providing the at least one image; and

a computing device associated with the imaging device, wherein the computing device is configured to receive

the at least one image and to carry out the steps of the method according to claim **18**.

**32.** A computer-readable non-transitory storage medium, comprising a computer program having instructions for carrying out the method of claim **18**.

**33.** A system for monitoring a condition of a tool of a machining tool, the system comprising:

an imaging device associated with the machining tool designed to acquire images of the tool; and

a computing system associated with the imaging device, the computing system adapted to train a semantic segmentation model according to the method of claim **29** and to use the trained semantic segmentation model to perform analysis of the images of the tool to infer the condition of the tool.

\* \* \* \* \*