LEARNING BASED ALPHA MATTING USING SUPPORT VECTOR REGRESSION

Zhanpeng Zhang[1,2] *Qingsong Zhu*[1,*] *Yaoqin Xie*[1]

[1]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[2]Sun Yat-Sen University, Guangzhou, China

ABSTRACT

Alpha matting refers to the problem of estimating the opacity mask of the foreground in an image. Many recent algorithms solve it with color samples or some local assumptions, causing artifacts when they fail to collect appropriate samples or the assumptions do not hold. In this paper, we treat alpha matting as a supervised learning problem and propose a new matting approach. Given the input image and a trimap (labeling some foreground/background pixels), we segment the unlabeled region into pieces and learn the relations between pixel features and alpha values for these pieces. We use support vector regression (SVR) in the learning process. To obtain better learning results, we design a training samples selection method and use adaptive parameters for SVR. Qualitative and quantitative evaluations on a matting benchmark show that our approach outperforms many recent algorithms in terms of accuracy.

Index Terms— alpha matting, support vector regression, foreground extraction, image segmentation

1. INTRODUCTION

Alpha matting aims to softly extract the foreground from an image. It plays an important role in many image and video editing operations and has been extensively studied. Formally, in alpha matting, an input image I is modeled as a linear combination of a foreground image F and background image B:

$$I = \alpha F + (1 - \alpha)B \quad (1)$$

where α is the opacity mask of the foreground (typically called alpha matte, like Figure 1d). The alpha value (opacity) of every pixel ranges from 0 to 1. Equation (1) is underconstrained since α, F and B on the right-hand side are all unknown. To solve this ill-posed problem, many recent approaches require additional constrains from user input. Trimaps and scribbles are the most common methods (like Figure 1b).

Considerable progress has been made in image alpha matting. Many existing algorithms use sampling-based or propagation-based techniques to estimate the alpha values. Sampling-based techniques analyze nearby labeled pixels [1] or pick out color samples from the labeled regions to estimate the alpha values [2-5]. Propagation-based techniques obtain the alpha matte by propagating constrains from the labeled regions to the unlabeled regions with local assumptions (e.g., local smoothness) [6, 7].

However, there are still some difficulties in alpha matting. For sampling-based techniques, foreground and background color samples are sometimes difficult to determine or even not in the search space (e.g., in translucent objects). Also, some local assumptions may not always hold and propagation-based techniques may fail in some cases (like thin structures or gaps). To avoid these problems, learning based matting [8] estimates a general model between image colors and alpha values with semi-supervised learning techniques. Local and global learning approaches are designed respectively and produce accurate results in many cases. However, the local learning approach is propagation-based style with a matting Laplacian matrix based on learning techniques. Errors occurred in the learning process may accumulate quickly. Global learning approach works well in slim unlabeled regions, but in coarse trimaps the results deteriorate [8].

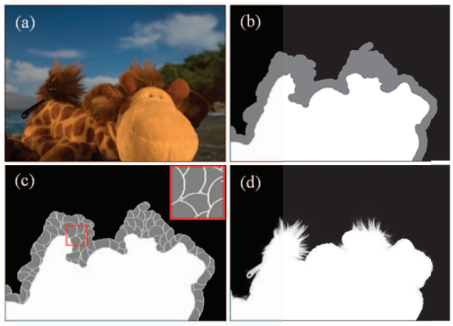

Figure 1. (a) Input image (b) Trimap (c) Unlabeled region segmentation (d) Alpha matte produced by our approach

To overcome these difficulties, we treat alpha matting as a supervised learning problem and propose a new alpha matting approach. Our goal is to learn the relations between pixel features and alpha values, instead of using color samples or local assumptions directly. Different from [8], we segment the unlabeled region into pieces and train a feature-α model for each piece using support vector regression. The segmentation algorithm and the order of training for these pieces are designed in a way that can reduce the accumulative errors. Training samples are selected from trimap or previously estimated pixels by similarity measurement. Adaptive parameters for SVR are adopted to improve the accuracy of the models. Experiments on the matting benchmark provided by Rhemann *et al.* [9] verify that the proposed approach produces more accurate results than many recent algorithms.

2. PROPOSED APPROACH

2.1. Unlabeled region segmentation

Usually nearby pixels have similar characteristics. Independently training a feature-α model for every pixel causes much redundant work. So nearby pixels can share a model. Unlabeled region segmentation algorithm aims to divide the unlabeled region into small pieces and decide the order of training models for these pieces. Because our approach uses previously estimated pixels as training samples, accumulative errors are unavoidable. An

appropriate order should make good use of labeled pixels in the trimap. So starting from the slim regions of the trimap can reduce the accumulative errors. The segmentation algorithm is as follows:

Algorithm 1 Unlabeled region segmentation

$\mathcal{F}, \mathcal{B}, \mathcal{U}$ denote the foreground, background and unlabeled region in the trimap respectively. r is a parameter (we set $r = 30$ pixels). Pieces (pixel sets) $\mathcal{P}_m (m > 0)$ are the segmentation results.

1: *Find the largest connected component* $\mathcal{F}_l$ *of* $\mathcal{F}$.
2: *For every pixel* $p \in \mathcal{U}$, *compute* $D_{\mathcal{F}_l}(p) = \min_{q \in \mathcal{F}_l} S(p,q)$, $D_{\mathcal{B}}(p) = \min_{q \in \mathcal{B}} S(p,q)$, *where* $S(\cdot)$ *denotes spatial distance.*
3: *Let* $\mathcal{U} = \{p_1, p_2, p_3 \ldots p_N\}$. *All pixels* p *in* $\mathcal{U}$ *are sorted by* $D(p) = D_{\mathcal{F}_l}(p) + D_{\mathcal{B}}(p)$, *in ascending order.*
4: *Set* $p \in \mathcal{P}_0$ *for all* $p \in \mathcal{U}$ *and* $m = 0$.
5: *for t = 1 to N*
6: *if* $p_t \notin \mathcal{P}_0$
7: *continue;*
8: *end*
9: $m = m + 1$;
10: *Find all pixels* $p \in \mathcal{P}_0$ *with* $S(p_t, p) < r$. *Set* $p \notin \mathcal{P}_0$, $p \in \mathcal{P}_m$ *and* $O(\mathcal{P}_m) = p_t$.
11: *end for*

Segmentation results $\mathcal{P}_m (m > 0)$ are circular or fan-shaped pieces as illustrated in Figure 1c. The subscript m represents the order to train models for these pieces. In line 1 of Algorithm 1, we find the largest connected component of $\mathcal{F}$ and ignore the rest. This is because in the trimap, users may add some small scribbles besides the largest one, but the foreground samples in these regions are usually not enough. $O(\cdot)$ in line 10 is defined for later use.

2.2. Model training for an unlabeled piece

2.2.1. Support Vector Regression

Support vector machine was originally introduced as a classifier based on the structural risk minimization principle. Its basic idea is to find the optimal hyperplane for data classification, so that the hyperplane has the largest distance to the nearest training samples of any class. Kernel trick is applied to make the data linearly separable by implicitly mapping them to a higher dimensional space. In [10], SVM classifier is used in image matting to provide discriminative information of foreground and background. Differently, we use ε-SVR [11], which is an extension of SVM for regression problems. Moreover, our approach is learning based and ε-SVR is applied to train feature-α models. Given a training set $\{(x_1, y_1), (x_2, y_2) \ldots (x_l, y_l)\}$ (x_i is input vector and y_i is output scalar), ε-SVR can be formulated as a convex optimization problem:

$$\min_{w,b,\xi,\xi^*} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$
$$\text{s.t.} \begin{cases} y_i - \langle w, x_i \rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \ge 0 \end{cases} \quad (2)$$

The regression function is:

$$y = \langle w, x \rangle + b \quad (3)$$

$\langle \cdot \rangle$ denotes inner product. The second term of the objective function in (2) represents the soft margin with C as the penalty parameter and ξ, ξ^* as the slack variables. w is the normal vector of the objective hyperplane while ε is the largest deviation without penalty. This convex optimization problem can be solved in its dual form obtained by utilizing Lagrange multipliers [11].

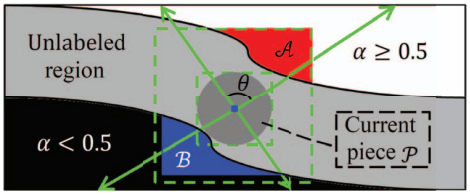

Figure 2. An illustration of candidate samples collection.

2.2.2. Features selection

Colors and color gradients are selected as the pixel features. Specifically, we use CIELAB color space with each component scaled in [0, 255]. For a pixel p, we define the features vector $x_p = \{l_p, a_p, b_p, gx_p^l, gx_p^a, gx_p^b, gy_p^l, gy_p^a, gy_p^b\}$. l, a, b denote the color components in CIELAB space. gx, gy represent the gradients of color components in horizontal and vertical directions. Gradients are selected in order to represent texture features.

2.2.3. Training samples selection

For an unlabeled piece, our goal is to obtain training samples which are similar to the unlabeled pixels and represent both local foreground and background characteristics. We first collect candidate samples from the trimap and previously estimated pixels, and then select the best candidate samples with specific criteria**.**

Candidate samples collection. Given the current unlabeled piece $\mathcal{P}$, two empty sample sets $\mathcal{A}$ and $\mathcal{B}$ (i.e., $|\mathcal{A}| = 0, |\mathcal{B}| = 0$), the collection method can be described as the following steps (also illustrated in Figure 2):

- *Step 1: Define the smallest rectangle* $\mathcal{R}$ *that can enclose* $\mathcal{P}$ *(like the small green rectangle in Figure 2).*
- *Step 2: For every labeled pixel i newly enclosed by* $\mathcal{R}$, *if* $\alpha_i \ge 0.5$ *and* $|\mathcal{A}| < \lambda|\mathcal{P}|$, *then* $i \in \mathcal{A}$; *else if* $\alpha_i < 0.5$ *and* $|\mathcal{B}| < \lambda|\mathcal{P}|$, *then* $i \in \mathcal{B}$.
- *Step 3: If* $|\mathcal{A}| \ge \lambda|\mathcal{P}|$ *and* $|\mathcal{B}| \ge \lambda|\mathcal{P}|$, *turn to step 4, else increase the width and height of* $\mathcal{R}$ *by one pixel and go back to step 2.*
- *Step 4: Shot 2n rays from* $O(\mathcal{P})$ *(like the blue pixel in Figure 2) with the length of* η. *The rays are separated by the equal angle* $\theta = \frac{\pi}{n}$. *Add labeled pixels on the rays to* $\mathcal{A}$ *if their alpha values* ≥ 0.5, *to* $\mathcal{B}$ *if their alpha values* < 0.5.

λ, n, η are parameters. We set λ=1.3, n = 6, η =300 in experiments. Pixels in $\mathcal{A} \cup \mathcal{B}$ are then treated as candidate training samples. Using an expanding rectangle to collect nearby samples can avoid spatial distance calculation between pixels, which is computation-intensive. We also collect samples on the rays because sometimes similar pixels do not fall in nearby labeled regions.

Samples selection. We define $H(p,q)$ to measure the similarity between an unlabeled pixel p in $\mathcal{P}$ and a candidate sample q:

$$H(p,q) = E(x_p, x_q) + \rho E\left((p_x, p_y), (q_x, q_y)\right) \quad (4)$$

where $E(\cdot)$ denotes the Euclidean distance between two vectors. ρ is a weighting factor (we set $\rho = 0.8$). Subscripts x and y denote horizontal and vertical coordinates respectively. Specially, before computing $H(\cdot)$, all features and coordinates of the pixels in $\mathcal{P} \cup \mathcal{A} \cup \mathcal{B}$ are normalized to [0,1]. We then select training samples in $\mathcal{A} \cup \mathcal{B}$ for pixels in $\mathcal{P}$ by the similarity measurement. For every pixel in $\mathcal{P}$, we select m most similar pixels and m most dissimilar pixels as the training samples (m is a small parameter and we set m = 3). This is because we want to obtain

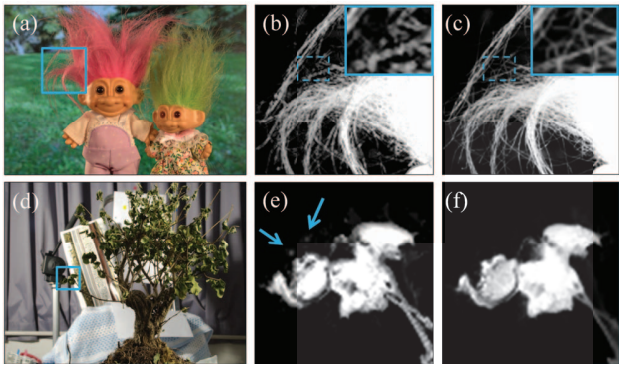

Figure 3. Results before and after post-processing.

samples similar to $\mathcal{P}$ while preserving the comprehensiveness. All the selected pixels for $\mathcal{P}$ are treated as the training samples after we remove the redundant. In addition, we assign a value $V(\cdot)$ to every pixel in $\mathcal{P}$, measuring the confidence of the alpha value estimated with these training samples. The confidence values are computed for later use and formulated as:

$$V(p) = exp\left\{-w_1 H_{min}(p) - w_2 D\big(O(\mathcal{P})\big)\right\} \quad (5)$$

where $H_{min}(p) = \min_{q \in \mathcal{A} \cup \mathcal{B}} H(p,q)$. w_1 and w_2 are parameters for normalization (e.g., 5 and 0.005 respectively).

2.2.4. ε-SVR training

We train a feature-α model using ε-SVR with the radial basis function (RBF) kernel:

$$k(x_i, x_j) = exp\left\{-\|x_i - x_j\|^2 / 2\sigma^2\right\} \quad (6)$$

where σ is a parameter (also called the kernel width) affecting the similarity measurement between two input vectors. Inappropriate selection of parameter σ, as well as C and ε in ε-SVR, will cause underfitting or overfitting problems as these parameters affect the model's generalization performance [12]. So we select these parameters adaptively. Let the training sample set be $\mathcal{T}$. $\mathcal{M}$ is a set containing the Euclidean distance between every two features vectors in $\mathcal{P} \cup \mathcal{T}$. σ^2 is set to the variance of $\mathcal{M}$. For the penalty parameter C, we follow the method in [12] and compute C by:

$$C = \max(|\bar{\alpha} + 3\tau|, |\bar{\alpha} - 3\tau|) \quad (7)$$

where $\bar{\alpha}$ and τ are the mean and standard deviation of the alpha values of $\mathcal{T}$. As the alpha values range from 0 to 1 and ε affects the precision of the model, we set

$$\varepsilon = 0.05 + \min\left(0.05, \tau\sqrt{\ln|\mathcal{T}|/|\mathcal{T}|}\right) \quad (8)$$

which is a combination of the method in [12] and our specific problem. Then we train the feature-α model using these parameters. After that, alpha values for $\mathcal{P}$ can be estimated by the regression function. All the other pieces are modeled through this method and we obtain the learning result.

2.3. Post-processing

Although nearby pixels share a feature-α model, roughness may still arise in smooth regions (like the hair in Figure 3b). This is partly because spatial information (e.g., coordinates) is not involved in the training process. So after learning the alpha values for all unlabeled pixels, we adopt the post-processing method like

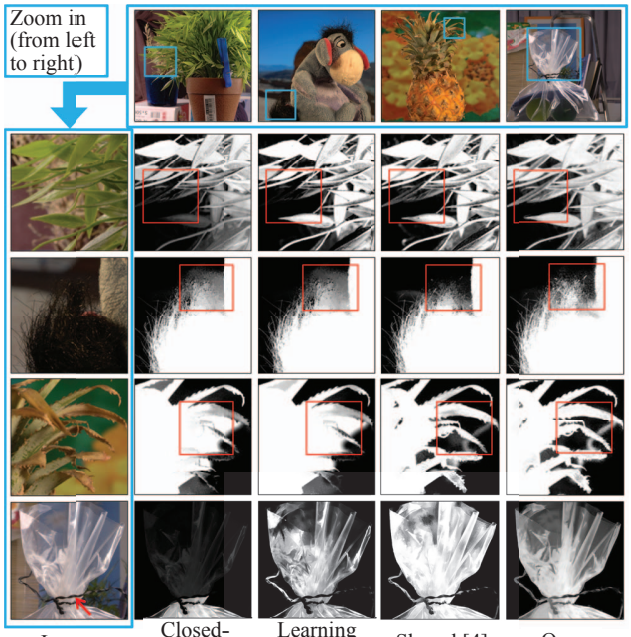

Figure 4. Qualitative comparisons among the results of different approaches. The red rectangles indicate the differences.

[3-5], combining the learning result $\tilde{\alpha}$ with the matting Laplacian matrix L of [6], which can be treated as a smoothness term. The final alpha matte is obtained by minimizing the function:

$$J(\alpha) = \alpha^T L \alpha + \varphi(\alpha - \tilde{\alpha})^T \Upsilon(\alpha - \tilde{\alpha}) + \omega(\alpha - \tilde{\alpha})^T \Gamma(\alpha - \tilde{\alpha}) \quad (9)$$

where φ and ω are parameters for weighting and normalization. φ is relatively large (e.g., 100) while ω is small (e.g., 0.1). α and $\tilde{\alpha}$ are treated as $N \times 1$vectors. N is the number of unlabeled pixels in the trimap. Υ and Γ are $N \times N$ diagonal matrices. Diagonal elements in Υ are one for labeled pixels in the trimap and zero for the others. Diagonal elements in Γ are the confidence values $V(\cdot)$ for unlabeled pixels in the trimap and zero for the others. In practice, the foreground in matting usually can be treated as a connected entirety. So, in the matte of $\tilde{\alpha}$, we select connected components with area less than 1% of the area of the largest one (like the regions indicated by the blue arrows in Figure 3e), and set $V(\cdot)$ of the pixels in these components to zero. That means, for these pixels, the alpha values are determined by the smoothness term. Figure 3 illustrates the effects produced by post-processing.

3. EXPERIMENTS AND EVALUATIONS

In the experiments, we compare the proposed approach with the state-of-the-art matting algorithms on the matting benchmark provided by Rhemann *et al.* [9]. The online benchmark has 8 natural test images and 3 different trimaps for each test image. Results of many recent image matting algorithms are available on website of the benchmark (*www.alphamatting.com*).

3.1. Qualitative evaluation

Figure 4 shows some cropped images in the benchmark and alpha mattes produced by closed-form matting [6], learning based matting [8], shared matting [4] and our approach. Closed-form matting

Metric	Closed-form	Learning based	Shared	**Ours**	**Ours***
Avg. MSE	1.26	1.18	0.90	0.82	**0.73**
Avg. SAD	15.49	15.51	13.65	14.03	**12.45**

Table 1. Average MSE and SAD error of different approaches on the benchmark of [9]. "Ours*" denotes our approach with the trimap expansion method in shared matting [4].

uses pure propagation-based technique. Shared matting uses a sampling-based method and optimizes the result with propagation-based technique. So these algorithms are representative and related to our approach. For the first and second images, the background is complex and colors of the leaves and fur are close to that of the background. All the four approaches cause artifacts. But ours performs better because of the learning technique and effective sampling method. In the second image, closed-from and learning based matting fail in the gaps of the foreground while shared matting and ours preserve the details. However, mistakes still arise in the regions where colors of the foreground and background overlap. For the last image, the foreground is a plastic bag. It is difficult to pick out definite foreground samples for the translucent object. Learning based and our matting approach can reveal the transparency while shared matting produces alpha values much too larger. Closed-form matting fails due to the rope (indicated by the red arrow in the image) violating the color line model [6].

3.2. Quantitative evaluation

For every approach, the benchmark uses ground truth alpha mattes to evaluate the 24 results (8 test images and 3 trimaps for each test image in the benchmark). Table 1 lists the average MSEs (mean squared error) and SADs (sum of absolute differences) of the 24 results of different approaches. The average MSE of our approach is smaller than other approaches, while the average SAD is just larger than that of share matting. However, share matting uses trimap expansion [4] as the preprocessing to reduce the unknown pixels. So, we also integrate the trimap expansion into our approach. The corresponding results are also listed in Table 1 and our approach performs best with respect to the two metrics.

The benchmark can also rank the approaches with respect to 4 error metrics: SAD, MSE, gradient and connectivity. Detailed definitions for the ranks and the latter two metrics can be found in [9]. We list the overall ranks in Table 2. The results show that our approach performs best on three out of four metrics. Generally, approaches combining sampling and propagation based techniques produce better results than the pure propagation-based [6,7]. However, pure propagation-based approaches perform better on the fourth metric. This is mainly because the affinity matrices of the propagation-based approaches concentrate on the neighboring relations and preserve the connectivity of the foreground object.

Our approach is implemented in Matlab with LIBSVM (implemented in C++) [15] for ε-SVR. The algorithm runs on a 3.0 GHz CPU with 2GB RAM. It takes 75 seconds for an image (about 800×600) in [9] on average. The time spent in LIBSVM is about 5 seconds per image.

4. CONCLUSIONS

A supervised learning based alpha matting approach is presented in this paper. We learn the relations between pixel features and alpha values with ε-SVR. The relations vary in different regions of the image. Effective training samples selection method and adaptive parameters for SVR are used to improve the learning accuracy. We also smooth the learning result with a matting Laplacian matrix [6]. Experiments demonstrate the advantages of our approach in terms of accuracy. Future work will include improving the robustness in complex scenes and regions with color ambiguity.

Approach	SAD	MSE	Gradient	Connectivity
Ours*	**3.2**	**2.9**	**3.3**	5.4
Shared [4]	3.4	3.8	3.5	7.9
Segmentation-based [13]	4.3	4.4	3.5	7.2
Improved color [3]	4.8	4.8	3.8	5.8
Shared (Real Time) [4]	5.5	6.3	7.8	9.5
Learning based [8]	5.8	6	6.6	6.7
Closed-form [6]	5.9	6.1	6.6	4
Large kernel [14]	7.3	6.8	7.7	5
Robust [2]	8	7.6	6.8	9.5
Random Walk [7]	11.4	11.4	11.4	**2.3**

Table 2. Overall ranks of the top performing approaches on the benchmark of [9] with respect to four metrics. The best results are in red. "Ours*" has the same meaning as the one in Table 1.

5. REFERENCES

[1] Y.Y. Chuang, B. Curless, D.H. Salesin, and R. Szeliski, "A bayesian approach to digital matting," *CVPR*, pp. 264-271, 2001.

[2] J. Wang and M.F. Cohen, "Optimized color sampling for robust matting," *CVPR*, pp. 17-22, 2007.

[3] C. Rhemann, C. Rother, and M. Gelautz, "Improving color modeling for alpha matting," *BMVC*, pp. 1155-1164, 2008.

[4] E.S.L Gastal and M.M Oliveira, "Shared sampling for real-time alpha matting," *Computer Graphics Forum*, vol. 29, no. 2, pp. 575-584, 2010.

[5] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun, "A global sampling method for alpha matting," *CVPR*, pp. 2049-2056, 2011.

[6] A. Levin, D. Lischinski, and Y. Weiss, "A closed form solution to natural image matting," *CVPR*, pp. 61-68, 2006.

[7] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random walks for interactive alpha-matting," *VIIP*, pp. 423-429, 2005

[8] Y. Zheng and C. Kambhamettu, "Learning based digital matting," *ICCV*, pp. 889-896, 2009.

[9] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, "A perceptually motivated online benchmark for image matting," *CVPR*, pp. 1826-1833, 2009.

[10] T. Hosaka, T. Kobayashi, and N. Otsu, "Image matting based on local color discrimination by SVM," *Pattern Recognition Letters*, vol. 30, pp. 1253–1263, 2009.

[11] A.J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199-222, 2004.

[12] V. Cherkassky and Y. Ma, "Practical selection of SVM parameters and noise estimation for SVM regression," *Neural Networks*, vol. 17, 113-126, 2004.

[13] C. Rhemann, C. Rother, P. Kohli, and M. Gelautz, "A spatially varying PSF-based prior for alpha matting," *CVPR*, pp. 2149-2156, 2010.

[14] K. He, J. Sun, and X. Tang, "Fast matting using large kernel matting Laplacian matrices," *CVPR*, pp. 2165-2172, 2010.

[15] C.C. Chang and C.J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1-27, 2011.