Head Pose Recognition with NNC-Trees

JIE JI,†1 NAOKI TOMINAGA,†1 YOSHIHIKO WATANABE,†1 KAZUHIKO HIRAKURI,†1 KEI SATO†1 and QIANGFU ZHAO†1

Pose recognition is important in many practical applications. For example, a driver assistance system can detect if the driver is tired, sleepy, or careless from the poses. A pet robot can detect certain behavior patterns of the human user. The main purpose of this study is to develop a driver assistance system that can protect the drivers from careless accidents. For this purpose, we propose a system for recognizing different poses of a human from the face images by using NNC-Tree. An NNC-Tree is a decision tree (DT) with each internal node containing a nearest neighbor classifier (NNC). We also developed a GUI for visualizing the prototypes in each NNC, as well as the whole tree. This interface makes it possible to understand, analyze, and reuse the learning results. This paper is a summary of what we have done so far.

1. Introduction

Head pose recognition is important in many practical applications. For example, a driver assistance system detects the driver’s conditions from a sequence of poses. A pet robot detects certain behavior patterns of the human user, and interacts with the human user in a friendlier manner.

In this study, we developed a system for recognizing different head poses based on human faces with NNC-Trees. The NNC-Tree is a decision tree (DT), with each internal node containing a nearest neighbor classifier (NNC). In the proposed system, the NNC in each internal node is trained by using the R4-rule, which is a very efficient algorithm for determining the proper number of prototypes through learning. Usually, NNCs obtained by the R4-rule are the smallest or nearly smallest.

There are mainly two reasons for us to use NNC-Trees for pose recognition (also true for face recognition, expression recognition, etc.) explained as follows.

(1) The NNC-Tree contains only a few prototypes which are a kind of abstracted representation of a large number of training patterns. Visualizing these prototypes, we can actually see what have been learned, and this is very helpful for understanding the learned results.

(2) The topological structure of the feature space is preserved naturally in the tree structure. That is, similar patterns are assigned to closely located nodes (e.g. nodes with the same parent or grand-parent) in the tree. This fact further improves the understandability of the learned results.

The rest of the paper is organized as follows. In the next section, we first review how to induce an
NCC-TREE using the \(R_4\)-rule. Experimental settings, methods, and results are provided in Section 3, Section 4, and Section 5. The last section draws some conclusions and proposes several future works.

2. Induction of NCC-Trees

2.1 NNC-Trees

An NNC-Tree\(^1\) is a DT in which each internal node contains an NNC. Fig. 1 shows the structure of an NCC-Tree. In general, we can implement an NNC-Tree as an \(N\)-ary DT \((N \geq 2)\). In this paper, however, we just consider binary NCC-Trees. NCC-Trees have three attractive features to pose recognition: they can classify unknown data more accurately than conventional DTs; their prototypes are also images which can be visualized easily; and the tree structures show the relationships among the face patterns in an understandable way.

The NCC used in each internal node classifies the input datum \(x\) into one of two groups. An NNC has a prototype set \(P\). For any \(x\), the NNC finds the nearest prototype \(p_i\) from \(P\) and categorizes \(x\) to the same group as \(p_i\).

Each terminal node of an NCC-Tree, on the other hand, contains a class label \(l_i\). Class 1–4 in Fig. 1 correspond to the class labels. The class label \(l_i\) is assigned to the input \(x\) if \(x\) is assigned to the terminal node which has \(l_i\) by the tree.

Any input datum \(x\) is recognized by an NNC-Tree as follows:

1. Set the root node as the current node.
2. If the current node is a terminal node, assign \(x\) with the class label of this node, then stop; otherwise, find the nearest neighbor of \(x\) from the prototypes of the NNC. Suppose that \(p^*\) is the nearest neighbor of \(x\), and the group label of \(p^*\) is \(i\) \((i = 1 \text{ or } 2)\), \(x\) is assigned to the \(i\)-th child of the current node.
3. Set the \(i\)-th child as the current node, then return to Step 2.

2.2 \(R_4\)-Rule

We use the \(R_4\)-rule\(^3\) to induce NNC-Trees. For each internal node, the \(R_4\)-rule can induce an NNC with a set of prototypes as small as possible.

The \(R_4\)-rule consists of four basic operations: recognition, remembrance, reduction and review. One learning cycle contains one of the following two sets: recognition, remembrance and review; or recognition, reduction and review.

In the operation recognition, the fitness of each prototype and the recognition rate \(r\) of the current NNC are calculated. The fitness of a prototype shows how important this prototype is for recognition. The following steps explain how to calculate the fitness:

1. All the fitness values are initialized to zero in each learning cycle.
2. When \(x\) (a training datum in the training data set \(T\)) cannot be classified correctly after removing a prototype \(p\) from the prototype set \(P\), update \(\text{fitness}(p)\) (the fitness of \(p\)) as follows:
   \[\text{fitness}(p) = \text{fitness}(p) + 1.\]
3. When \(x\) can be classified correctly only after removing \(p\) from \(P\), update \(\text{fitness}(p)\) as follows:
   \[\text{fitness}(p) = \text{fitness}(p) - 1.\]

The operation after recognition is selected from remembrance and reduction based on \(r\). When \(r\) is less than a desired recognition rate \(r_0\), the operation remembrance is selected.

In the operation remembrance, a new prototype is inserted into \(P\). The prototype to be inserted is one of the training data which cannot be recognized correctly with the current prototype set \(P\). In the operation reduction, the prototype with the smallest fitness is removed from \(P\). In the operation review, \(P\) is updated only if the NNC with \(P\) cannot recognize a training datum \(x\) correctly. The following shows how to update \(P\):

\[p_0 = p_0 + \alpha(x - p_0)\] \[p_i = p_i - \alpha(x - p_i)\]

where \(p_0\) is the prototype nearest to \(x\) and is in the
same class $x$ belongs to; $p_1$ is the prototype nearest to $x$ and is in a different class than $x$; and $\alpha$ is the convergence ratio, which becomes smaller as the update cycle of the operation $\text{review}$ goes on.

3. Experimental Settings

To evaluate the performance of the system, we conducted experiments using real data.

3.1 Databases

We used two databases in the experiments: our original one and one based on the Yale Face Database B.

3.1.1 Database 1

We collected data by ourselves. During data collection, we asked subjects to sit in a fixed chair, and asked them to look at seven directions, corresponding roughly to the following seven poses:

- Pose 0: Look forward
- Pose 1: Look at the meters
- Pose 2: Look at the back mirror
- Pose 3: Look at the right side mirror
- Pose 4: Look at the left side mirror
- Pose 5: Look at outside through the right window
- Pose 6: Look at outside through the left window

Fig. 2 and Fig. 3 show the positions and meaning of teacher signals.

We took pictures of ten subjects for Database 1. For each subject and each pose, we took five slightly different pictures. All together we have $10 \times 7 \times 5 = 350$ pictures. The reason why we consider only ten subjects is that the number of drivers of a family car is usually limited. Fig. 4 shows some examples from Database 1 (note that when we face to the subject, the direction is converse). Fig. 5 shows pictures of the same subject at the same pose.

3.1.2 Database 2

All the pictures in Database 1 have the same brightness level. Therefore, we also conducted experiments for the head pose recognition under different brightness conditions using an database based on the Yale Database B.

For Database 2, we selected pictures of 5 subjects from Yale Database B. For each subject, we selected 7 poses as shown in Fig. 7. For each subject and each pose, there are 15 pictures with different brightness conditions (see Fig. 6). Altogether, we have $5 \times 7 \times 15 = 525$ pictures.

3.2 Data Normalization

First, we segmented a certain area of the original picture and changed the segmented picture into grayscale. Here, we just segmented the same area of the picture, no matter this was a whole face or not. This was because, in the real environment, the
camera and driver's position are relatively fixed. To segment the face precisely and automatically is one of our future works. Fig. 8 shows two possible segmentation instances.

After face segmentation, we performed four kinds of normalizations: Brightness adjustment, Contrast adjustment, Resolution normalization and Matrix normalization.

### 3.2.1 Brightness Adjustment

In brightness adjustment, the average grayscale of a picture is calculated and then the following equation is used to adjust the brightness level:

\[pix(i, j) = pix(i, j) - (avg - 127.5)\]

After this step, the brightness was adjusted. For example, Fig. 10 shows Fig. 9 after such step.

### 3.2.2 Contrast Adjustment

In contrast adjustment, the histogram to all color gradations is stretched by using the following two equations:

\[pix(i, j) = \begin{cases} 
127.5 & (pix(i, j) < 127.5) \\
127.5 - \frac{pix(i, j) - 127.5}{pix_{min}} & (pix(i, j) \geq 127.5) 
\end{cases}\]

(3)

\[pix(i, j) = \begin{cases} 
127.5 & (pix(i, j) < 127.5) \\
\frac{pix(i, j) - 127.5}{pix_{max} - 127.5} & (pix(i, j) \geq 127.5) 
\end{cases}\]

(4)

Fig. 11 shows the picture and the histogram after this step.

### 3.2.3 Resolution Normalization

Every picture was changed into 30×36 resolutions.

### 3.2.4 Matrix normalization

For each picture, every pixel was normalized to [0,1] and the whole picture was convert into a vector. The teacher signal was then added as the last element of the vector.

Finally, we got the vector which had the following format and used it as an input to our system:
4. Experimental Methods

We conducted three kinds of experiments as follows:

(1) In Experiment 1, we used Database 1. In order to reduce the contingency, we considered 5 cases. In each case, 1 of the 5 pictures for each subject and each pose is used for test, and the rest are used for training. The number of test data is $10 \times 7 = 70$. Because each run of the $R_4$-rule based NNC-Tree induction may produce a different result, we performed 10 runs for each case to make the results more reliable.

(2) In Experiment 2, we used Database 2. In order to investigate the relationships between head pose recognition and brightness condition, we considered 15 cases. In each case, 1 of the 15 pictures for each subject and each pose is used for test, and the rest are used for training. The number of test data is $5 \times 7 = 35$, and all test images are of the same brightness condition. We conducted 10 experiments for each case.

(3) In Experiment 3, we also used Database 2. In order to investigate the head pose recognition of a subject whose face is not in the training data set, we considered 5 cases. In each case the pictures of 1 of the 5 subjects are used for test, and the rest are used for training. There are $7 \times 15 = 105$ test images. Again, we conducted 10 experiments for each case.

5. Experimental Results

Tables 1–3 shows the experimental results. They contain the average recognition rate for each case and the total average of all cases.

5.1 Discussion

Table 1 shows the results for Experiment 1. For this experiment, the average recognition rate is more than 98% even though we only use four pictures of the same subject and the same pose for the training. There are, however, still some pictures difficult to be classified. Fig. 12 shows such an example. Although this picture belongs to Pose 2 (look at the back mirror), the system sometimes assigns this picture to Pose 0 (look forward). Such kind of mistakes may be reduced if we use several frames captured by a video camera.

Table 2 shows the results for Experiment 2. For Experiment 2, when the pictures have the different brightness conditions, the average recognition rate is more than 92%. In the bright condition, it is more than 98%. The recognition rates dropped compared to Experiment 1. We can say it is because of different brightness conditions.

Table 3 shows the results for Experiment 3. For Experiment 3, when the pictures have different brightness conditions and a subject of test examples is different from the subjects of training examples, the average recognition rate is about 60%. However, it depends on subjects. The worst one, a female subject, is about 35%. It is difficult to say that the system recognized the poses well.
6. Conclusion and Future Works

In this paper, we have proposed a method to use NNC-Trees for head pose recognition. The proposed method is interesting because we can understand from the resulted NNC-Tree and the prototypes the learned knowledge as well as the topological structure of the feature space.

From the experimental results we can see that the performance of the system is not very sensitive to the brightness conditions and small changes in each pose. However, the system seems to be sensitive to subjects. That is, if a subject is not included in the training set, the recognition rate will be very low.

There are also some other topics to study, and the future work mainly includes two steps: 1. collecting more data to test our method; 2. applying NNC-Trees to pose sequences recognition. Step 2 is important because, in some real applications, we cannot detect a subject’s action accurately only based on signal head poses.

参考文献