

Single-View Based Recognition of Faces Rotated in Depth¹

Thomas Maurer*, Christoph von der Malsburg*⁺

*Ruhr-Universität Bochum
Institut für Neuroinformatik
D-44780 Bochum, Germany

E-mail: maurer@neuroinformatik.ruhr-uni-bochum.de

⁺University of Southern California
Dept. of Computer Science and Section for Neurobiology
Los Angeles, CA 90089, USA

Abstract

We present a method for recognizing objects (faces) on the basis of just one stored view, in spite of rotation in depth. The method is not based on the construction of a three-dimensional model for the object. Our recognition results represent a significant improvement over a previous system developed in our laboratory. We achieve this with the help of a simple assumption about the transformation of local feature vectors with rotation in depth.

1 Introduction

It is known from biology that objects are seen under two aspects [UM]: The 'Where' and the 'What' aspect. 'Where' means localization in space and perspective, 'What' means classification and identification. In technical systems the last task often turns out to be the more difficult one which needs the highest accuracy (e.g. it is less difficult to find a face in an image than to recognize it). Thus, visual features are needed which are *invariant* under perspective changes or can be *transformed*. Invariant features seem attractive, avoiding computational effort like mathematical transformations. But invariance always means loss of information because a feature detector is insensitive to the input change to which it is invariant. To compensate, many different detectors with different invariances have to be combined. This seems biologically plausible but is suited better for parallel than sequential hardware. Here, we have chosen to use localized visual features which can be transformed according to perspective changes in a homogenous style. We will describe our transformation method in detail and apply it to the recognition of faces rotated in depth.

The paper has three parts: The following section is a very short description of our face recognition system without feature transformation. The second section describes the mathematics of the feature transformation. In the last part we describe the incorporation of the feature transformation method into our face recognition system. The system learns the

transformation of a face from a gallery of faces in different poses. We present recognition results for rotated faces which are only known from a frontal view.

2 Description of the face recognition system

As basic visual feature we use a local image descriptor in the form of a vector called 'jet' [LVB]. Each component of a jet is a Gabor wavelet of specific frequency and orientation, extracted from the image at a definite point x . We are employing Gabor wavelets of 5 different frequencies and 8 different orientations for a total of 40 complex components. Such a jet describes the area surrounding x . Many jets taken at different positions form a graph which describes an object in the image. To compare jets and graphs, similarity functions are defined: The normalized dot product of two jets yields their similarity, the sum over jet similarities being the similarity of graphs of identical geometry. But distortions of the grid can also be taken into account to compute the graph similarity [LVB]. Features and similarities defined in this way are robust against changes in illumination and contrast.

In order to create an appropriate face graph for a gallery of 50-80 frontal faces of equal size, we place the nodes by hand, e.g., at the center of the eyes or at the tip of the nose. There are typically 40-50 nodes forming a face graph. Such a gallery contains the general face knowledge for frontal faces. For a new frontal face of the same size these nodes of the graph can be found automatically [WFK], even if this person is not in the gallery. After finding the face graph it can be compared with others. In order to achieve a similar scale for new images, they are preprocessed; this works similar to finding the graph nodes and is also described in [WFK].

The whole procedure can be repeated in the same manner for face images in other poses, say profile or half profile (faces rotated by approximately 45 degrees in depth). Faces of the same pose can be compared without problems, but comparing face images of different poses remains difficult. The graphs of all poses can be defined in such a way that they share many nodes, i.e. there are always eye nodes, nose

¹Supported by grants from the German Federal Ministry for Science and Technology (413-5839-01 IN 101 B9) and from the US Army Research Lab (01/93/K-0109).

nodes etc., and graph converters can be constructed which allow comparing graphs of different poses on the subset of common nodes. However, as long as the jets, the visual features extracted from a single 2D-view, are not transformed, results are less than satisfactory [WFK].

3 Feature Transformation

For simplicity we assume the local object shape near a single node to be flat. The orientation of this area, i.e. its normal vector, has somehow to be determined. Once this normal vector and the geometric transformation of the object (e.g. rotation angle of the face) are known, the jet at this node can be transformed analytically.

One could argue that this assumption of flat areas is grossly violated at nodes near the tip of the nose or the eyes. However, many nodes in a face graph can be approximately described this way and even if the area is not locally flat the system tries to find an effective area with an effective normal vector to describe the transformation as accurately as possible. This will become clearer below.

Another difficulty is that with increasing rotation angle between two views of a face fewer nodes can be used for comparing the pictures because of reduced overlap of the graphs. This reduction in mutual information will make face recognition impossible beyond a certain rotation angle, as is borne out by psychophysical experiments [KBC]. The only exception may be the comparison of left and right profile views, which are similar due to facial symmetry.

We will now proceed as follows: We will determine the matrix connecting the image coordinates of a flat area on an object viewed from two different perspectives. Then we calculate a matrix connecting the corresponding jets. In the next chapter we will apply the feature transformation to faces and show how the necessary normal angles can be learned automatically.

3.1 Geometric transformation of a planar object

For introducing our choice of coordinate system and variables let us briefly repeat how a camera image transforms when a planar object is moved. We start with the simple example in figure 1.

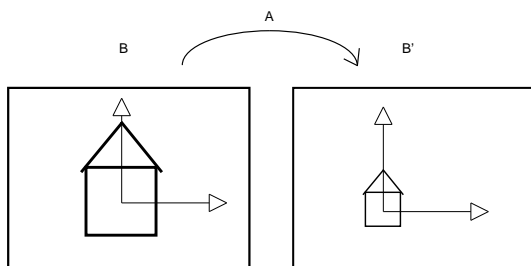


Figure 1: How do the image coordinates of the house transform from B to B' in a coordinate system whose origin is shifted to the corresponding object point?

For a planar object this is in general an affine

transformation including a shift vector. But we are only interested in how the surroundings of one point transform (where our jet will be located). So we put the origin of the camera coordinate system in every picture at this point and have

$$\mathbf{x}' = \mathbf{A} \mathbf{x} \quad (1)$$

with

$$\mathbf{A} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \quad (2)$$

in our example. In the case of a rotation in the image plane by angle α the matrix \mathbf{A} becomes

$$\mathbf{A} = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix}. \quad (3)$$

To scale and rotate simultaneously these matrices are multiplied.

Now we discuss the more interesting problem of rotation in depth.

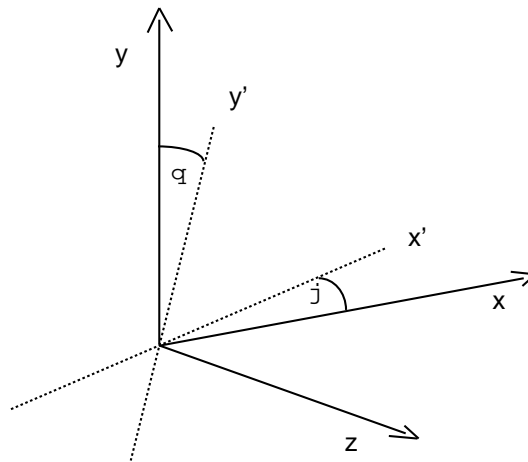


Figure 2: A flat object initially lies in the xy -plane, is rotated around the y -axis by angle ϕ and around the x' -axis by angle θ . If the camera is placed at a positive z -coordinate, how do image coordinates transform if the flat object is rotated from the unprimed to the primed orientation?

We start with the simple case shown in figure 2. \mathbf{A} is easily determined with the help of the transformed unit vectors:

$$\mathbf{A} = \begin{bmatrix} \cos \phi & \sin \theta \sin \phi \\ 0 & \cos \theta \end{bmatrix}. \quad (4)$$

If the planar object does not lie in the camera plane at the beginning but at (ϕ_1, θ_1) , then it can be imagined being transformed from the camera plane, resulting in

$$\mathbf{x}' = \underbrace{\mathbf{A}(\phi_2, \theta_2) \mathbf{A}^{-1}(\phi_1, \theta_1)}_{\mathbf{A}} \mathbf{x} \quad (5)$$

Rotation in the object plane can be added by multiplying with the ordinary rotation matrix eq. (3), and

similarly with scaling. Every 3D operation with the planar object, i.e. every combination of rotation in depth and in the plane and scaling, can now be translated into the corresponding matrix \mathbf{A} transforming the image coordinates of this planar object.

It should be mentioned that the above determination of \mathbf{A} is not the only possibility. In contrast, there are as many procedures as choices of variables to parameterize 3D rotations. The optimal procedure for determining \mathbf{A} depends on the situation, i.e. the rotation axes and measured angles, but it is always a purely geometrical problem. The resulting \mathbf{A} for transforming the image is—of course—unique.

3.2 Linear jet transformation

Looking again at figure 1, we now turn to the question: how does the jet located at the origin of the coordinate system transform from the left image to the right? We will not restrict ourselves to scaling here but solve the general problem with arbitrary \mathbf{A} .

The jet component $j_{\mathbf{k}}$ with wave vector \mathbf{k} was defined as

$$j_{\mathbf{k}}(\mathbf{x}) \Big|_{\mathbf{x}=0} = \int d\mathbf{x}' I(\mathbf{x}') \psi_{\mathbf{k}}(\mathbf{x} - \mathbf{x}') \quad (6)$$

With $I(\mathbf{x})$ grey level intensity at \mathbf{x} and $\psi_{\mathbf{k}}(\mathbf{x})$ the Gabor wavelet centered at the origin

$$\psi_{\mathbf{k}}(\mathbf{x}) = \frac{k^2}{\sigma^2} e^{-\frac{k^2}{2\sigma^2} x^2} \left\{ e^{i \mathbf{k} \cdot \mathbf{x}} - e^{-\frac{\sigma^2}{2}} \right\}. \quad (7)$$

The integral can be taken to infinity because of the finite support of $\psi_{\mathbf{k}}(\mathbf{x})$. The corresponding jet component in the other image containing the transformed planar object is

$$j'_{\mathbf{k}} = \int d\mathbf{x} I(\mathbf{A}^{-1}\mathbf{x}) \psi_{\mathbf{k}}(\mathbf{x}). \quad (8)$$

Now we get by substitution

$$j'_{\mathbf{k}} = \int d\mathbf{x} I(\mathbf{x}) \psi_{\mathbf{k}}(\mathbf{A} \mathbf{x}) \det(\mathbf{A}). \quad (9)$$

We make the Ansatz

$$\psi_{\mathbf{k}}(\mathbf{A} \mathbf{x}) \det(\mathbf{A}) \approx \sum_{\mathbf{k}'} c_{\mathbf{k}\mathbf{k}'}(\mathbf{A}) \psi_{\mathbf{k}'}(\mathbf{x}), \quad (10)$$

although this can only be approximated. One can assume that the accuracy of this Ansatz increases with sampling density in \mathbf{k} -space. When the $c_{\mathbf{k}\mathbf{k}'}(\mathbf{A})$ are determined, the jet can be transformed according to

$$\mathbf{j}' = \mathbf{C}(\mathbf{A}) \mathbf{j}. \quad (11)$$

We now compute the $c_{\mathbf{k}\mathbf{k}'}(\mathbf{A})$. By multiplying eq.(10) with $\psi_{\mathbf{k}''}(\mathbf{x})$ and integrating we have

$$\begin{aligned} \sum_{\mathbf{k}'} c_{\mathbf{k}\mathbf{k}'}(\mathbf{A}) \int d\mathbf{x} \psi_{\mathbf{k}''}(\mathbf{x}) \psi_{\mathbf{k}'}(\mathbf{x}) \\ \stackrel{!}{=} \int d\mathbf{x} \psi_{\mathbf{k}''}(\mathbf{x}) \psi_{\mathbf{k}}(\mathbf{A} \mathbf{x}) \det(\mathbf{A}) \\ = \int d\mathbf{x} \psi_{\mathbf{k}''}(\mathbf{A}^{-1}\mathbf{x}) \psi_{\mathbf{k}}(\mathbf{x}) \end{aligned}$$

To keep notation simple we define

$$\langle \mathbf{k} | \mathbf{k}' \rangle \doteq \int d\mathbf{x} \psi_{\mathbf{k}}(\mathbf{x}) \psi_{\mathbf{k}'}(\mathbf{x}) \quad (12)$$

$$\langle \mathbf{k}(\mathbf{A}^{-1}) | \mathbf{k}' \rangle \doteq \int d\mathbf{x} \psi_{\mathbf{k}}(\mathbf{A}^{-1}\mathbf{x}) \psi_{\mathbf{k}'}(\mathbf{x})$$

and we write only $c_{\mathbf{k}\mathbf{k}'}$, dropping the (\mathbf{A}) . This gives

$$\sum_{\mathbf{k}'} \langle \mathbf{k}'' | \mathbf{k}' \rangle c_{\mathbf{k}\mathbf{k}'} = \langle \mathbf{k}''(\mathbf{A}^{-1}) | \mathbf{k} \rangle \quad \forall \mathbf{k}''. \quad (13)$$

This is a linear equation system determining the row vector $c_{\mathbf{k}}^T$:

$$\begin{bmatrix} \langle \mathbf{k}_1 | \mathbf{k}_1 \rangle & \cdots & \langle \mathbf{k}_1 | \mathbf{k}_N \rangle \\ \langle \mathbf{k}_2 | \mathbf{k}_1 \rangle & \ddots & \vdots \\ \vdots & & \\ \langle \mathbf{k}_N | \mathbf{k}_1 \rangle & \cdots & \langle \mathbf{k}_N | \mathbf{k}_N \rangle \end{bmatrix} \begin{pmatrix} c_{\mathbf{k}_1} \\ c_{\mathbf{k}_2} \\ \vdots \\ c_{\mathbf{k}_N} \end{pmatrix} = \begin{pmatrix} \langle \mathbf{k}_1(\mathbf{A}^{-1}) | \mathbf{k} \rangle \\ \langle \mathbf{k}_2(\mathbf{A}^{-1}) | \mathbf{k} \rangle \\ \vdots \\ \langle \mathbf{k}_N(\mathbf{A}^{-1}) | \mathbf{k} \rangle \end{pmatrix}.$$

The calculation of the integrals $\langle \cdot | \cdot \rangle$ can be done analytically and will be described in the appendix. Solving this linear system for one \mathbf{k} leads to one row vector $c_{\mathbf{k}}^T$ of the jet transformation matrix \mathbf{C} . Repeating this for all \mathbf{k} of the sampling set would do the job. This is written compactly by combining the right hand side vectors for all \mathbf{k} to one matrix, which results in one matrix equation:

$$\mathbf{T} \begin{bmatrix} c_{11} & \cdots & c_{N1} \\ c_{12} & \ddots & \vdots \\ \vdots & & \\ c_{1N} & \cdots & c_{NN} \end{bmatrix} =$$

$$\begin{bmatrix} \langle \mathbf{k}_1(\mathbf{A}^{-1}) | \mathbf{k}_1 \rangle & \cdots & \langle \mathbf{k}_1(\mathbf{A}^{-1}) | \mathbf{k}_N \rangle \\ \langle \mathbf{k}_2(\mathbf{A}^{-1}) | \mathbf{k}_1 \rangle & \ddots & \vdots \\ \vdots & & \\ \langle \mathbf{k}_N(\mathbf{A}^{-1}) | \mathbf{k}_1 \rangle & \cdots & \langle \mathbf{k}_N(\mathbf{A}^{-1}) | \mathbf{k}_N \rangle \end{bmatrix}.$$

The overlap matrix on the left hand side we called \mathbf{T} and the matrix on the right hand side we call \mathbf{S} . Then the matrix \mathbf{C} is determined by

$$\mathbf{C}^T = \mathbf{T}^{-1} \mathbf{S} \Rightarrow \mathbf{C} = \mathbf{S}^T (\mathbf{T}^{-1})^T = \mathbf{S}^T \mathbf{T}^{-1}$$

Note that \mathbf{T} is independent of the geometric transformation \mathbf{A} and needs to be inverted only once.

4 Applying feature transformation to face recognition

4.1 Method

How can we now use the formulas derived above for comparing faces in different poses? First, we assume we have the normal vectors of a human face at the graph nodes (we will show below how to find them). Second, we need the rotation angle between the two poses (approximately), e.g., for comparing half profile against frontal view it is 45 degrees. We remark here that it is not a big problem to estimate the pose if it is unknown: By comparing a new face with a small set of faces in all poses the maximum of the average similarity will give the correct pose. From the normal angles and the rotation angle the orientation of the imagined plane at every graph node in both galleries is known and the matrix eq. (5) describes their geometric transformation. With this 2×2 matrix one can compute the jet transformation matrix \mathbf{C} for every node as described in the previous section. All face graphs can then be transformed by simply multiplying every jet with its transformation matrix.

Finally, we have to show how to determine the normal vectors: We take a set of 50-80 pairs of two poses, e.g. half profile and frontal images of the same persons. Now we can search for optimal normal vectors by optimizing the recognition performance on this training gallery. Optimizing recognition means that the similarity s between face graphs belonging to the same person should be high and similarity between face graphs belonging to different persons should be low. So the cost function C to be maximized would be

$$C = \frac{1}{N^2} \sum_i \sum_j (s_{ii} - s_{ij}) \quad (14)$$

where the two indices run over the two pose galleries. Unfortunately, optimizing all normal vectors simultaneously would need excessive computer time and, even worse, optimizing the normal angles of all nodes by maximizing one function C would probably lead to poor generalization. Fortunately, it is not difficult to avoid both problems: Because the face graph similarity is the sum of the single jet similarities, the search for the optimal normal angles can be done for each node independently. Thereby the recognition performance based on the observed node is maximized, i.e. s in eq. (14) now means jet similarity instead of graph similarity. Now there is a function C for every single node but only two free parameters — the two normal angles determining the normal vector — and 50–80 jet pairs as training data. With this procedure, good generalization results, and learning is fast: It takes less than an hour on a Sun Sparc 10.

In our actual procedure we have not used the cost function eq. (14) but a modified version. The problem with the above function is that s values stemming from badly placed nodes or that are outliers for other reasons have too much influence, resulting in effective normal vectors which minimize the error due to the outliers, even if these are the faces which can not be recognized at the end due to mistakes. So

we decreased the influence of outliers by using two sigmoid functions [Kr]:

$$C = \frac{1}{N} \sum_i \arctan \left(\alpha_0 \frac{1}{N} \sum_j \arctan(\alpha_1 (s_{ii} - s_{ij})) \right) \quad (15)$$

The two parameters α_0 and α_1 were determined only once as they depend mainly on the method and less on the particular pose pair. We found $\alpha_0 = 50$ and $\alpha_1 = 500$ to be a good choice. Results do not depend on their precise value. These values mean that the sigmoid is more important in the second sum than in the first.

4.2 Results

To test our method we used a database prepared by the US Army Research Laboratory (ARL). The database consists of galleries with frontal, quarter, half profile, and profile pose. One problem with this database is that only the frontal and profile gallery are homogenous while the half profile gallery consists of faces rotated in depth from 20 degrees to nearly 80 degrees. There were about 200 half left and about 150 half right pictures. We flipped the half left pictures to get 350 half right views in total. To create a reliable test set we went through the half profile gallery once and discarded faces with very high and very low rotation angles. In addition we had to discard half profile images without frontal counterpart. Out of the remaining 160 pairs we randomly selected 70 for training and the other 90 for testing. Rotation angle of these faces still varies significantly from the nominal 45 degrees.

After learning the normal vectors on the training set we got the following results on the test set, always comparing the half profile face to the frontal gallery.

	without rotation	rotating h to f	rotating f to h
correct recog. (rank 1)	36%	50%	53%
among best 5% (rank 1-4)	56%	73%	73%

In figure 3 a typical image pair is shown. The effective normal vectors as learned on the training set are also plotted: They are all of equal length and projected onto the image plane. These are the effective normal vectors for the whole gallery, not just for this special person. Some vectors seeming nonsensical here will make sense on the average (many people, especially women, having other hairstyles). The normal vectors in both pictures are the same, in the first picture the effective normal vectors are rotated by 45 degrees together with the face. Normal vectors can only be determined at nodes common to grids of both poses, which is the reason why the distribution of the vectors looks somewhat asymmetric. Black dots in the two images are normal vectors nearly perpendicular to the image plane. It is difficult to interpret single effective normal vectors, but

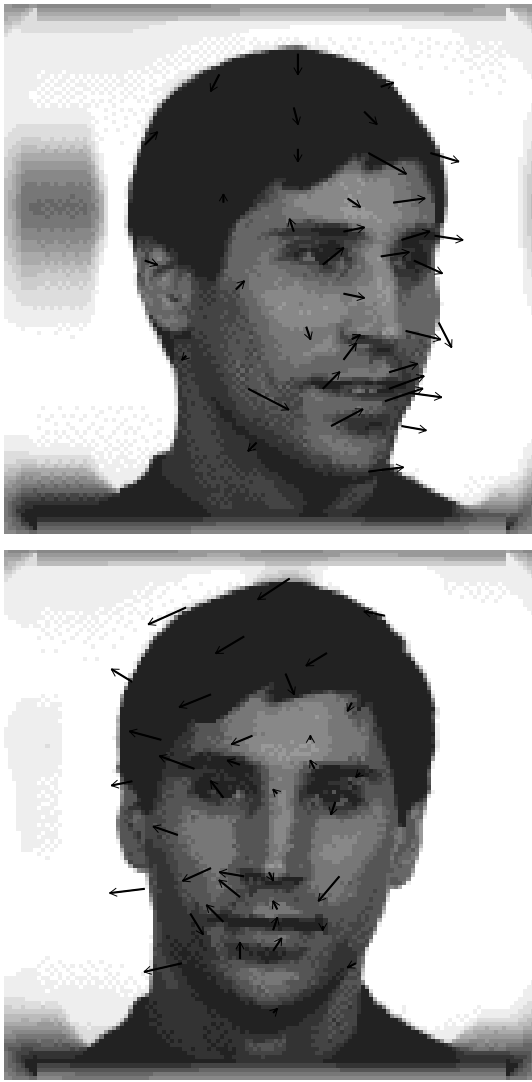


Figure 3: Typical half profile and frontal face pose. Effective normal vectors at the corresponding graph nodes are shown, in the first image they are rotated together with the face.

a more global trend can be observed: Normal vectors often seem to be pulled in the opposite rotation direction. This is analogous to maximizing the effective plane in the rotated image.

5 Conclusions

The method presented here is clearly able to improve recognition performance on face galleries rotated in depth. On the other hand recognition rates are still far away from being near 100%, in contrast to frontal face recognition. It is evidently much more difficult to recognize rotated faces than frontal ones. Until now we had only allowed one set of normal vectors for the whole gallery. This has obvious shortcomings: Faces are very different, for instance in hairstyle. It is possible to learn normal vectors for single faces of a training set whereby the rotation angle can also be

varied. For recognizing a test image the face from the training set with the highest similarity will be selected together with its learned normal angles and rotation angle. This will also allow us to handle with varying rotation angles. The computational effort will be much larger and some optimizations will be necessary for realizing this idea. Dealing with rotation in depth necessitates steps beyond image recognition towards threedimensional object representation. In the long run we will have to find or construct feature detectors (indicating, e.g., hair or skin type, gender, beard [WFK]) which are independent of pose. In the near future we will try to develop methods for automatically creating and dealing with galleries derived from face video recordings carefully controlled for lighting and pose. This material could be a starting point for developing a more general feature transformation scheme by weakening or dropping the flat plane assumption and which probably will have to be based on more powerful visual features.

References

- [WFK] L. Wiskott, J.-M. Fellous, N. Krüger, C. von der Malsburg, *Face Recognition and Gender Determination*, in *Proceedings of the International Workshop on Automatic Face- and Gesture-Recognition*, 1995.
- [LVB] M. Lades, J.C. Vorbrüggen, J. Buhmann, J. Lange, C. von der Malsburg, R.P. Würtz, W. Konen, *Distortion Invariant Object Recognition in the Dynamic Link Architecture*, *IEEE Trans. Comp.*, Vol.42, No.3, p.300-311, 1993.
- [KBC] P. Kalocsai, I. Biederman, E.E. Cooper, *To what extent can the recognition of unfamiliar faces be accounted for by a representation of the direct output of simple cells*, Poster presented at the *Annual Meeting of the Association for Research in Vision and Ophthalmology*, Sarasota, FL, May 1994.
- [Kr] N. Krüger, *Learning Weights in Discrimination Functions using a priori Constraints*, submitted to *17. Symposium der Deutschen Arbeitsgemeinschaft für Mustererkennung (DAGM)*, 1995.
- [UM] L.G. Ungerleider, M. Mishkin, *Two Cortical Visual Systems in Analysis of Visual Behavior*, edited by D.J. Ingle, M.A. Goodale and R.J.W. Mansfield, p.549-586, MIT Press, 1982.

Appendix

We want to compute the integral $\langle \mathbf{k}(\mathbf{M}) | \mathbf{k}' \rangle$ for an arbitrary 2x2 matrix

$$\mathbf{M} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

The normal overlap integral $\langle \mathbf{k} | \mathbf{k}' \rangle$ is the special case $\mathbf{M} = \mathbf{I}$.

With the definitions eq. (13) and eq. (7) we have

$$\langle \mathbf{k}(\mathbf{M}) | \mathbf{k}' \rangle = \int dx \int dy$$

$$\begin{aligned}
& \frac{k^2}{\sigma^2} e^{-\frac{k^2}{2\sigma^2}[(a_{11}x+a_{12}y)^2+(a_{21}x+a_{22}y)^2]} \\
& \left\{ e^{-i[k_x(a_{11}x+a_{12}y)+k_y(a_{21}x+a_{22}y)]} - e^{-\frac{\sigma^2}{2}} \right\} \\
& \frac{k'^2}{\sigma^2} e^{-\frac{k'^2}{2\sigma^2}(x^2+y^2)} \left\{ e^{i(k_x x+k_y y)} - e^{-\frac{\sigma^2}{2}} \right\}
\end{aligned} \tag{16}$$

Before we proceed we calculate for arbitrary reals a , b , c

$$\int_{-\infty}^{\infty} dx e^{-ax^2+bx+icx} = \sqrt{\frac{\pi}{a}} e^{\frac{b^2-c^2+i2bc}{4a}}.$$

Using this formula it is somewhat tedious but straightforward to solve eq. (16). With the abbreviations

$$\begin{aligned}
b_1 & \equiv a_{11}^2 + a_{21}^2 \\
b_2 & \equiv a_{12}^2 + a_{22}^2 \\
b_3 & \equiv a_{11}a_{12} + a_{21}a_{22} \\
b_4 & \equiv k_x a_{11} + k_y a_{21} \\
b_5 & \equiv k_x a_{12} + k_y a_{22} \\
t_1 & \equiv (k^2 b_1 + k'^2)(k^2 b_2 + k'^2) - k^4 b_3^2 \\
t_2 & \equiv k^2 b_2 + k'^2
\end{aligned}$$

we get the result

$$\begin{aligned}
\langle \mathbf{k}(\mathbf{M}) | \mathbf{k}' \rangle & = \frac{2\pi}{\sigma^2} \frac{k^2 k'^2}{\sqrt{t_1}} \\
& \left\{ e^{-\frac{\sigma^2}{2} \frac{(k'_y - b_5)^2}{t_2}} e^{-\frac{\sigma^2}{2} \frac{[t_2(k'_x - b_4) + k^2 b_3(k'_y - b_5)]^2}{t_1 t_2}} \right. \\
& - e^{-\frac{\sigma^2}{2}} e^{-\frac{\sigma^2}{2} \frac{b_5^2}{t_2}} e^{-\frac{\sigma^2}{2} \frac{[t_2 b_4 + k^2 b_3 b_5]^2}{t_1 t_2}} \\
& - e^{-\frac{\sigma^2}{2}} e^{-\frac{\sigma^2}{2} \frac{k_y'^2}{t_2}} e^{-\frac{\sigma^2}{2} \frac{[t_2 k'_x + k^2 b_3 k'_y]^2}{t_1 t_2}} \\
& \left. + e^{-\sigma^2} \right\}.
\end{aligned} \tag{17}$$

Note that the matrix elements are real. With the usual parameter setting $\sigma = 2\pi$ only the first term will be nonvanishing. But we decided to evaluate and implement this general formula valid for arbitrary σ , which gives us the ability to vary parameters as required.